# Velleman K8000 & Linux HOWTO

Heimo Schön *Heimo Schön <heimo.schoen@gmx.at>* Version 0.4 , 6. Feb. 2002

Setup a *Velleman K8000* on a *Linux* System

## 1 Overview

### 1.1 About this HOWTO

This document describes how to setup your *Velleman K8000 Board* on a *Linux* system, using the i2c Kerneldrivers

If you have any suggestions to this document, than feel free to contact me:

- Heimo Schön *<heimo.schoen@gmx.at>*
- *http://schoen.priv.at*

Have a lot of fun with your Velleman K8000 Board.

### 1.2 Links

Links to other sites with aditional information to K8000 running with Linux:

- *http://yahoogroups.com/group/k8000*
- *http://struyve.mine.nu:8080/pageloader.pl?block=k8000*
- *http://groups.yahoo.com/group/k8000/files/linux/libk8000/*

### 1.3 Thanks to

- *hihihi <hihihi@wanadoo.nl>* for his help in testing and correcting this mini-howto.

### 1.4 Version History

- 0.1 08 Dec 2001 first trial
- 0.2 28 Jan 2002 included a hint from Hihihi from NL to rmmod the lp-driver in the startscript
- 0.3 3 Feb 2002 A lot of corrections and new features from *hihihi <hihihi@wanadoo.nl>* included
- 0.4 6 Feb 2002 Minor bug fix from *hihihi <hihihi@wanadoo.nl>*

## 2 Introduction

In this chapter I want to discribe the history of this mini-mini-howto.

## 2.1 Target

In 1995 I was searching for an expansion board for my home-PC. It was a PC-AT with 386 CPU and at that time I was programming with clipper. I wanted to read some digital and analog input and I thought that I don't want a board in my PC. I was searching for an external solutions.

I found the velleman K8000 board in an electronic shop. There was a source code for the driver available. I compiled this code with borland-C compiler and linked it together with my Clipper application. This program is running since Summer 1995 an doing some watching of my oil-heating at home.

Today this old machine is the last DOS-based machine at home. All other PCs are running with Linux. I searched through the web a lot of hours and found only the i2c-velleman.c Kernel-driver in the kernel-tree. But I couldn't find any examples, how to connect from a small C-program to the K8000 board. So I decided, to try to hack it by myself. After the program(s) was(were) running I wrote this small HOWTO.

However, if you find I could help you learning howto connect from C to your Velleman K8000 board, feel free to send me a short message. I would be happy to hear from you - I am sure thare there are some other K8000 user out there :-))

In a discussion with *hihihi <hihihi@wanadoo.nl>* I found out that there also exists a velleman-library called libk8000. This HOWTO does not support this library-solution. Here I discribe the connection via kernel-drivers. At this point I want to thank *hihihi <hihihi@wanadoo.nl>* for his hints and for testing this HOWTO.

## 2.2 HW-Configuration

When you finished soldering your K8000, you should switch the two DIP-Switches to the off position.

## 2.3 SW-Configuration

You should prepare your system by running the following script (I stored it under the name "velle")

```
#! /bin/bash

echo free the Parport ...
/sbin/rmmod lp
/sbin/rmmod parport_pc
/sbin/rmmod vmppuser
/sbin/rmmod parport

echo i2c Treiber laden ...
/sbin/modprobe i2c-velleman
/sbin/modprobe i2c-dev

echo create the device-handle and free it for all users
cd /dev
rm velleman

# The following part was written by hihihi#wanadoo.nl
# Jan 31 2002
# A variation on the ./velle script of Heimo Schoen.
Filename=/proc/bus/i2c
line="not empty yet"
echo
echo "    Look for I2C devices in /proc/bus/i2c :"
```

```
echo
while [ "$line" != "" ]
do
  read line
  echo "     " $line
  if echo "$line" | grep -q "Velleman K8000"
  then
    K8000Entry=$line
  fi
done <"$Filename"

echo
echo "      Found a Velleman K8000 entry:" $K8000Entry
echo

K8000Nod=${K8000Entry#i2c-}
echo $K8000Nod
K8000Nod=`expr match "$K8000Nod" '\([0-9]*[0-9]\)'`
echo $K8000Nod

mknod velleman c 89 $K8000Nod
chmod a+rw velleman

echo Etwas auf die K8000 jagen ...
echo hallo > /dev/velleman
```

You can download this script *here*

# 3 the programs

## 3.1 check-i2c.c

This small C-program checks all addresses from 0 to 127 if there are reacheable i2c-chips connected to your parallel port. Before you start this program, you should compile and link it by typing make all and load all i2c kernel modules with the above described script named "velle"

```
/*
**   $Id: check-i2c.c,v 0.01
**
**   File:     check-i2c.c
**   Author:   Heimo Schoen / heimo.schoen@gmx.at
**
**   Abstract: Routine for testing the Kerneldriver
**             ../driver/char/i2cv-elleman.c
**
#
# Copyright (C) 2001,2002 by Heimo Schön  <heimo.schoen@gmx.at>
#
# 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, BUT NOT (!!!)
# any later version.
```

```
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
*/

# include <errno.h>
# include <stdio.h>
# include <signal.h>
# include <stdarg.h>
# include <time.h>

#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
  int fh   = 0;
  unsigned int port = 0;
  int res;
  char devicename[100] = "/dev/velleman";

  if (argc == 2)
    sprintf(devicename, "%s", argv[1]);

  fh  = open(devicename, O_RDWR);

  if (fh < 3) {

    printf("\n");
    printf("check-i2c had a problem opening %s\n", devicename);
    printf("\n");
    printf("  please execute script velle included in the package check-i2c\n");
    printf("\n");
    printf("  You should have a file /dev/velleman generated with the command\n");
    printf("     mknod /dev/velleman c 89 0\n");
    printf("\n");
    printf("  You also should have loaded i2c-velleman i2c-dev i2c-core and i2c-algo-bit\n");
    printf("  into your kernel. See script velle for details how to do this\n");
    printf("\n");
    printf("usage of check-i2c:\n");
    printf("\n");
    printf("    check-i2c [devicename]\n");
```

```
    printf("\n");
    printf("    if parameter devicename is not given, check-i2c will try to open\n");
    printf("    /dev/velleman.\n");
    printf("\n");
    printf("(c) 2001,2002 by Heimo Schön / heimo.schoen@gmx.at / http://schoen.priv.at\n");
    printf("\n");
    printf("      have a lot fun ...\n");
    printf("\n");

    exit (1);

  } else {

    printf("\n");
    printf("check-i2c will search now on %s for reachable i2c-devices and will show you the addresses:\n", devicename);
    printf("\n");

    // Loop across all available 7-Bit addresses
    for (port=0; port<=127; port++) {

      // Try to ioctl this device-address
      if (ioctl(fh, I2C_SLAVE, port) < 0)
        perror ("IOCTL Problem");

      /* Using SMBus commands */
      res = i2c_smbus_write_byte(fh, 0x00);
      if (res >= 0)
        printf("found i2c-chip at     i2c-address: 0x%02x (%d)      (real-address: 0x%02x)\n",
               port, port, port*2);

      //usleep(1000);

    } // loop port=0-127

    printf("\n");
    printf("NOTE:\n");
    printf("the i2c-drivers in the kernel use a 7-bit adress.\n");
    printf("E.g. if you want to write to a PCF8574A at the adress 0x70\n");
    printf("than you have to divide 0x70 by two. This is 0x38, which is the\n");
    printf("address for the kernel-drivers.\n");
    printf("With other words, the kernel-driver is shifting your adress (0x38)\n");
    printf("one byte left an adds 0 if you write ore 1 if you read from the\n");
    printf("address.\n");
    printf("I know, this is a little bit complecated ;-) But this is also the\n");
    printf("reason why I wrote this small program\n");
    printf("\n");
    printf("     have a lot of fun ...\n");
    printf("\n");

    exit (0);
  }
}
```

You can download this program with the Makefile *here*

## 3.2 velleman.c

velleman.c is doing some test with digital-output on your velleman K8000. This program is expecting that you have a K8000 connected to your /dev/lp0 and that all i2c drivers are loaded as described before. It switches all outputs on and off. There is a small help available (just read the source ;-)

```
/*
**   $Id: velleman.c,v 0.01
**
**   File:     velleman.c
**   Author:   Heimo Schoen / heimo.schoen@gmx.at
**
**   Abstract: Routine for testing the Kerneldriver
**             ../driver/char/i2cv-elleman.c
**
#
# Copyright (C) 2001,2002 by Heimo Schön  <heimo.schoen@gmx.at>
#
# 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, or (at your option)
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
*/

# include <errno.h>
# include <stdio.h>
# include <signal.h>
# include <stdarg.h>
# include <time.h>

# include <sys/types.h>
# include <sys/time.h>
# include <sys/socket.h>

#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
#include <unistd.h>

// ----------------------------------------------------------
int p_i2c_write(unsigned int  vf_fh,
```

```
                unsigned int  vf_port,
                unsigned char vf_data)
{
  int vp_res;

  if (ioctl(vf_fh,I2C_SLAVE,vf_port))
    perror ("IOCTL Problem");
  vp_res = i2c_smbus_write_byte(vf_fh, ~vf_data);
  if (vp_res < 0)
    perror("i2c_smbus_write_byte");
  return (vp_res);
}



// ----------------------------------------------------------
int p_write(unsigned int  vf_fh,
            unsigned int  vf_port,
            unsigned char vf_data)
{
  int vp_res;
  unsigned char vp_buf[10];

  if (ioctl(vf_fh,I2C_SLAVE,vf_port))
    perror ("IOCTL Problem");
  vp_buf[0] = ~vf_data;
  vp_res = write(vf_fh, vp_buf, 1);
  return (vp_res);
}



// ----------------------------------------------------------
int main(int argc, char *argv[])
{
  int fh  = 0;
  int res;
  int i;
  unsigned int port1 = 0x38;
  unsigned int port2 = 0x39;
  unsigned char buf[10];
  char devicename[100] = "/dev/velleman";
  unsigned char testmode = 0;

  if (argc >= 2) {
    if (strlen(argv[1]) > 1)
      sprintf(devicename, "%s", argv[1]);
    if (strlen(argv[1]) == 1)
      testmode = atoi(argv[1]);
  }

  if (argc >= 3) {
    if (strlen(argv[2]) > 1)
      sprintf(devicename, "%s", argv[2]);
    if (strlen(argv[2]) == 1)
```

```
    testmode = atoi(argv[2]);
}

fh  = open(devicename, O_RDWR);

if (fh < 3) {

  printf("\n");
  printf("velleman had a problem opening %s\n", devicename);
  printf("\n");
  printf("  please execute script velle included in the package velleman\n");
  printf("\n");
  printf("  You should have a file /dev/velleman generated with the command\n");
  printf("     mknod /dev/velleman c 89 0\n");
  printf("\n");
  printf("  You also should have loaded i2c-velleman i2c-dev i2c-core and i2c-algo-bit\n");
  printf("  into your kernel. See script velle for details how to do this\n");
  printf("\n");
  printf("usage of velleman:\n");
  printf("\n");
  printf("    velleman [devicename] [test]\n");
  printf("\n");
  printf("    if parameter devicename is not given, velleman will try to open\n");
  printf("    /dev/velleman.\n");
  printf("\n");
  printf("    parameter [test] has to be numeric and tells which testnumber you want\n");
  printf("    If Parameter [test] is not found, velleman will only check the PCF8574A if present\n");
  printf("\n");
  printf("example:  velleman /dev/i2c-0 2\n");
  printf("\n");
  printf("(c) 2001,2002 by Heimo Schön / heimo.schoen@gmx.at / http://schoen.priv.at\n");
  printf("\n");
  printf("       have a lot fun ...\n");
  printf("\n");

  exit (1);

} else {

  // checking if we can find the two PCF5874A

  if (ioctl(fh,I2C_SLAVE,port1) < 0)
    perror ("IOCTL Problem");
  res = i2c_smbus_write_byte(fh, 0x00);
  if (res >= 0)
    printf("found i2c-chip at     i2c-address: 0x%02x (%d)      (real-address: 0x%02x)\n",
           port1, port1, port1*2);

  if (ioctl(fh,I2C_SLAVE,port2) < 0)
    perror ("IOCTL Problem");
  res = i2c_smbus_write_byte(fh, 0x00);
  if (res >= 0)
    printf("found i2c-chip at     i2c-address: 0x%02x (%d)      (real-address: 0x%02x)\n",
           port2, port2, port2*2);
```

```
    sleep(1);

    if (testmode == 1) {
      printf("Starting testmode 1 : Counter    (stop test with CTRL-C)\n");
      while (1) {
        /* Using SMBus commands */
        for (i=0; i<=0xff; i++) {
          p_i2c_write(fh, port1, i);
          p_i2c_write(fh, port2, i);

          usleep(5);
        }
      }
    }

    if (testmode == 2) {
      printf("Starting testmode 2 : Knightrider    (stop test with CTRL-C)\n");
      while (1) {
        for (i=1; i<=15; i++) {

          if (i<=7) {
            p_write(fh, port1, 1<<i);
            p_write(fh, port2, 0);
          } else {
            p_write(fh, port2, 1<<(i-8));
            p_write(fh, port1, 0);
          }
          usleep(1);
        }
        for (i=14; i>=0; i--) {

          if (i<=7) {
            p_write(fh, port1, 1<<i);
            p_write(fh, port2, 0);
          } else {
            p_write(fh, port2, 1<<(i-8));
            p_write(fh, port1, 0);
          }
          usleep(1);
        }
      }
    }

/*     ioctl(fh,I2C_UDELAY,5); */
/*     ioctl(fh,I2C_MDELAY, 2); */
/*     ioctl(fh,I2C_RETRIES,3); */

    if (ioctl(fh,I2C_SLAVE,port1) /* ioctl(fh, I2C_SLAVE, port1) */ < 0)
      perror ("IOCTL Problem");

    errno = 0;
    res = ~ i2c_smbus_read_byte(fh);
    if (errno != 0)
      perror("i2c_smbus_read_byte");
    else
```

```
        printf("i2c_smbus_read_byte: 0x%02x\n",
               res);

      sleep(1);

      /* Using I2C Read, equivalent of i2c_smbus_read_byte(file) */
      res = read(fh,buf,1);
      if ( res == 1)
        printf("POSIX-Read: %d bytes buf[0]=0x%02x\n",
               res, ~ buf[0]);

    }


    return 0;
  }
```

You can download this program with the Makefile *here*

## 3.3 Makefile

Last but not least, the Makefile to the above two sources

```
#
# Makefile for the velleman testprogram
#
# Copyright (C) 2001,2002 Heimo Schön <heimo.schoen@telecom.at>
#
# 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, or (at your option)
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#

           CFLAGS        =  -pedantic -Wall -O -g

           LDOPTIONS     = -g -DYY_SKIP_YYWRAP
           RM            = rm -f
           CP            = cp -v
           STRIP         = strip --verbose

           VELLEMANSRCS = velleman.c
           VELLEMANOBJS = $(patsubst %.c,%.o,$(VELLEMANSRCS))
           CHECKSRCS    = check-i2c.c
```

```
        CHECKOBJS    = $(patsubst %.c,%.o,$(CHECKSRCS))

all:: velleman check-i2c

tags:
        etags *.[hc]

velleman:  $(VELLEMANOBJS) $(DEPLIBS)
        $(RM) $@
        $(CC) -o $@ $(VELLEMANOBJS) $(LDOPTIONS) -O0 -g -L/usr/lib/termcap -lreadline -lhistory -ltermcap

check-i2c:  $(CHECKOBJS) $(DEPLIBS)
        $(RM) $@
        $(CC) -o $@ $(CHECKOBJS) $(LDOPTIONS) -O0 -g -L/usr/lib/termcap -lreadline -lhistory -ltermcap

clean::
        $(RM) velleman check-i2c *.o *~

depend::
        @if [ -n "$(VELLEMANSRCS)" ] ; then set -x;\
          $(DEPEND) $(DEPENDFLAGS) $(VELLEMANSRCS);\
        fi
        @if [ -n "$(CHECKSRCS)" ] ; then set -x;\
          $(DEPEND) $(DEPENDFLAGS) $(CHECKSRCS);\
        fi
```

### 3.4 notes

If you dont want to use the kernel-drivers, you find a solution for a user-space program *here*

## 4 Allgemeines

### 4.1 Copyright