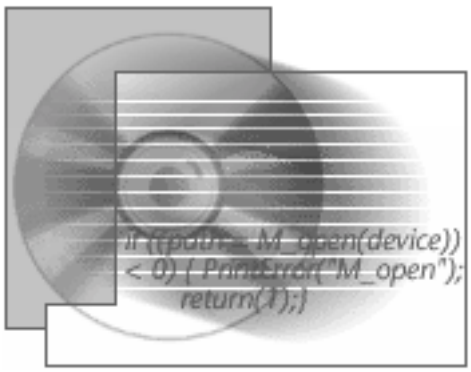
Embedded Solutions

VME4L – VME for Linux
Installation Instructions



User Manual

mikro elektronik
gmbh · nürnberg

About this Document

This manual deals with the installation of MEN's VME for Linux software package VME4L (Linux device driver for MEN PCI-to-VME bridge, article no. 13Z014-90).

History

Edition	Comments	Technical Content	Date of Issue
E1	First edition	T. Schnürer	2004-12-08

Conventions

This sign marks important notes or warnings concerning proper functionality of the product described in this document. You should read them in any case.

italics — Folder, file and function names are printed in *italics*.

bold — **Bold** type is used for emphasis.

`monospace` — A `monospaced` font type is used for listings, C function descriptions or wherever appropriate.

hyperlink — Hyperlinks are printed in blue color.

The globe will show you where hyperlinks lead directly to the Internet, so you can look for the latest information online.

IRQ# /IRQ — Signal names followed by "#" or preceded by a slash ("/") indicate that this signal is either active low or that it becomes active at a falling edge.

Copyright Information

MEN Mikro Elektronik reserves the right to make changes without further notice to any products herein. MEN makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MEN assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages.
"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts.
MEN does not convey any license under its patent rights nor the rights of others.
Unless agreed otherwise, MEN products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the MEN product could create a situation where personal injury or death may occur. Should Buyer purchase or use MEN products for any such unintended or unauthorized application, Buyer shall indemnify and hold MEN and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that MEN was negligent regarding the design or manufacture of the part.

Unless agreed otherwise, the products of MEN Mikro Elektronik are not suited for use in nuclear reactors and for application in medical appliances used for therapeutical purposes. Application of MEN's products in such plants is only possible after the user has precisely specified the operation environment and after MEN Mikro Elektronik has consequently adapted and released the product.

All brand or product names are trademarks or registered trademarks of their respective holders.

Information in this document has been carefully checked and is believed to be accurate as of the date of publication; however, no responsibility is assumed for inaccuracies. MEN Mikro Elektronik accepts no liability for consequential or incidental damages arising from the use of its products and reserves the right to make changes on the products herein without notice to improve reliability, function or design. MEN Mikro Elektronik does not assume any liability arising out of the application or use of the products described in this document.

Please recycle

Germany
MEN Mikro Elektronik GmbH
Neuwieder Straße 5-7
90411 Nuremberg
Phone +49-911-99 33 5-0
Fax +49-911-99 33 5-901
E-mail info@men.de
www.men.de

France
MEN Mikro Elektronik SA
18, rue René Cassin
ZA de la Châtelaine
74240 Gaillard
Phone +33 (0) 450-955-312
Fax +33 (0) 450-955-211
E-mail info@men-france.fr
www.men-france.fr

UK
MEN Micro Ltd
Whitehall, 75 School Lane
Hartford, Northwich
Cheshire UK, CW8 1PF
Phone +44 (0) 1477-549-185
Fax +44 (0) 1477-549-178
E-mail info@menmicro.co.uk
www.menmicro.co.uk

USA
MEN Micro, Inc.
PO Box 4160
Lago Vista, TX 78645-4160
Phone (512) 267-8883
Fax (512) 267-8803
E-mail sales@menmicro.com
www.menmicro.com

Contents

1 Installing VME4L 5
1.1 Preparing the Kernel Sources 5
1.2 Build Option 1: Using MEN's MDIS Build System 6
1.3 Build Option 2: Using the Linux Kernel Build System. 8
1.3.1 Compiling Test Programs 10
1.4 VME4L Major Number 11
2 RTAI Support for VME Interrupts 12

1 Installing VME4L

The two modules *vme4l-core* and *vme-pldz002* replace the old driver *vme-menpci2vme*. The VME4L driver can be built in two environments:

1 Using MEN's MDIS build system
(MDIS package for Linux 13M000-13 >= 3.0).

2 Using the Linux kernel build system.

Option 1 is the most convenient. At least, you should use it when you want to use MDIS drivers for devices on VMEbus. With this option, VME4L is built as a kernel module.

Option 2 can be used when you want to use only the userland API of VME4L, or if your kernel doesn't support loadable modules and you have to link VME4L statically into the kernel.

1.1 Preparing the Kernel Sources

No matter whether you use option 1 or 2, you should do the following:

☑ You must have the sources of the kernel that shall be running on your target with the matching *.config* script.
E.g. under SuSE, install package *kernel-sources* using Yast2, then do the following:

```
# cd /usr/src/linux
# cp /boot/vmlinuz.config .
# make oldconfig
```

☑ Remove the old *vme-menpci2vme* driver from your kernel, if present:
Check if *<kernel-dir>/drivers/char/vme-menpci2vme.c* exists. If so, modify the *drivers/char/Makefile*:

- From the "export-objs := ...." list, remove *vme-menpci2vme.o*.
- Remove line

```
obj-$(CONFIG_MEN_PCI2VME) += vme-menpci2vme.o
```

1.2 Build Option 1: Using MEN's MDIS Build System

☑ Download the MDIS system package for Linux, article no. 13m00013.zip (version >= 3.0) from MEN's website.
This package also contains this VME driver.

☑ Install the MDIS system package as described in MDIS4 under Linux user manual (article no. 21M000-17). All files will be installed in */opt/menlinux*.

☑ Use the MDIS configuration wizard to define your system. If you select a component with VME support (e.g. A12, A15, A500), the MDIS wizard automatically adds the required VME modules.

☑ If you are using ELinOS, add the following lines to the ELinOS project's *autonode.sh*:

```
# VME4L
major=230
echo node "/dev/vme4l_a16d16"          c $major 0 0666 0 0 ""
echo node "/dev/vme4l_a16d32"          c $major 2 0666 0 0 ""
echo node "/dev/vme4l_a24d16"          c $major 4 0666 0 0 ""
echo node "/dev/vme4l_a24d16_blt"      c $major 5 0666 0 0 ""
echo node "/dev/vme4l_a24d32"          c $major 6 0666 0 0 ""
echo node "/dev/vme4l_a24d32_blt"      c $major 7 0666 0 0 ""
echo node "/dev/vme4l_a32d32"          c $major 8 0666 0 0 ""
echo node "/dev/vme4l_a32d32_blt"      c $major 9 0666 0 0 ""
echo node "/dev/vme4l_a32d64_blt"      c $major 10 0666 0 0 ""
echo node "/dev/vme4l_slave0"          c $major 11 0666 0 0 ""
echo node "/dev/vme4l_slave1"          c $major 12 0666 0 0 ""
echo node "/dev/vme4l_slave2"          c $major 13 0666 0 0 ""
echo node "/dev/vme4l_slave3"          c $major 14 0666 0 0 ""
echo node "/dev/vme4l_slave4"          c $major 15 0666 0 0 ""
echo node "/dev/vme4l_slave5"          c $major 16 0666 0 0 ""
```

☑ If you are using a self-hosted system, create device nodes as follows:

```
major=230
mknod "/dev/vme4l_a16d16"          c $major 0
mknod "/dev/vme4l_a16d32"          c $major 2
mknod "/dev/vme4l_a24d16"          c $major 4
mknod "/dev/vme4l_a24d16_blt"      c $major 5
mknod "/dev/vme4l_a24d32"          c $major 6
mknod "/dev/vme4l_a24d32_blt"      c $major 7
mknod "/dev/vme4l_a32d32"          c $major 8
mknod "/dev/vme4l_a32d32_blt"      c $major 9
mknod "/dev/vme4l_a32d64_blt"      c $major 10
mknod "/dev/vme4l_slave0"          c $major 11
mknod "/dev/vme4l_slave1"          c $major 12
mknod "/dev/vme4l_slave2"          c $major 13
mknod "/dev/vme4l_slave3"          c $major 14
mknod "/dev/vme4l_slave4"          c $major 15
mknod "/dev/vme4l_slave5"          c $major 16
```

☑ Build your MDIS configuration as described in the MDIS4 under Linux user manual (21M000-17).

☑ Install the files on the target as described in the MDIS4 under Linux user manual (21M000-17) (not required for ELinOS).

☑ On the target, you should find the following files in */lib/modules/$KERNEL_VERSION/misc*:

- *men_vme4l-core.o*
- *men_pldz002.o*

☑ Load these modules on the target (on self-hosted systems, run *depmod* before):

```
target:# modprobe men_pldz002
```

This will load both modules. You should see an output on the console or in */var/log/messages* or */proc/kmsg* similar to this:

```
vme4l-core $Revision: 1.4 $,  (No bridge driver attached)
vme4l_z002_init_module
vme-menpldz002: found PLDZ002 bridge (rev x), irq y
VME4L bridge driver has registered: PLDZ002 VME bridge (rev 7), \
vme4l-pldz002 $Revision: 1.4 $
```

☑ If required, you can then load further MDIS/BBIS drivers, e.g.:

```
target:# modprobe men_bb_a201
target:# modprobe men_ll_m22
```

☑ You can now try the test programs. E.g. the following dumps standard VME space address E00000:

```
# vme4l_rwex 4 e00000 100 2 r
```

1.3 Build Option 2: Using the Linux Kernel Build System

☑ Install the VME4L package in an arbitrary directory (let's choose *~/vme4l*):

```
$ mkdir ~/vme4l
$ cd ~/vme4l
$ unzip <path-to-download-dir>/13z01490.zip
```

☑ Copy the driver sources into the kernel tree. You may have to do this as "root":

```
# cd ~/vme4l/DRIVERS/VME4LX/DRIVER_K24
# ./install-to-kernel <kernel-dir>
```

☑ Modify the *<kernel-dir>/drivers/char/Makefile*:

- Add file *vme4l-core.o* to "export-objs := ....".
- Add line

```
obj-$(CONFIG_MEN_VME4L) += vme4l-core.o
```

- Add line

```
obj-$(CONFIG_MEN_PLDZ002) += vme4l-pldz002.o
```

☑ Modify *<kernel-dir>/drivers/char/Config.in*:

☑ Search for "CONFIG_MEN_PCI2VME". If you found it, remove the following entry block:

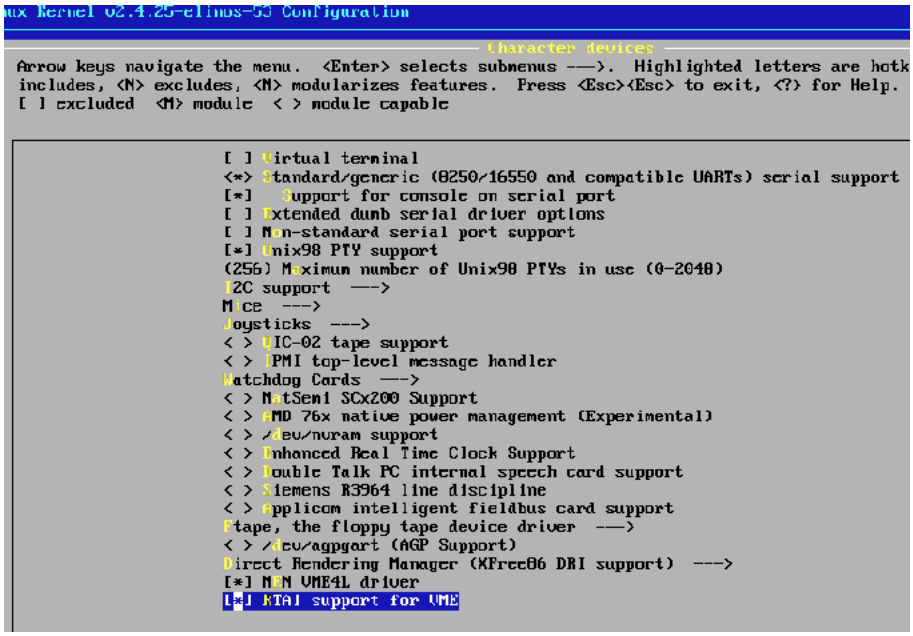
```
tristate 'MEN PCI to VME bridge support' CONFIG_MEN_PCI2VME
if [ "$CONFIG_MEN_PCI2VME" != "n" ]; then
    define_bool CONFIG_MEN_VME_KERNELIF y
fi
```

☑ Add the following before "endmenu":

```
tristate 'MEN VME4L driver' CONFIG_MEN_VME4L
if [ "$CONFIG_MEN_VME4L" != "n" ]; then
    define_bool CONFIG_MEN_VME_KERNELIF y
    dep_bool 'RTAI support for VME' CONFIG_MEN_VME_RTAI_KERNELIF
$CONFIG_MEN_VME_KERNELIF
fi
```

If you changed the content of the *Config.in* file above properly and then call the kernel build system through *make menuconfig*, it should give you the choices as in Figure 1, menuconfig with selections for VME, on page 9 below.

Figure 1. menuconfig *with selections for VME*



☑ Then, rebuild your kernel using your build facilities function (e.g. *elk* from ELinOS) or call the plain *make*:

```
# cd <kernel-dir>
# make
```

☑ If you have chosen static linking, reinstall your kernel and reboot.
(This may be different, depending on the Linux distribution used!)

```
# make lilo
# reboot
```

☑ Create device nodes on your target:

```
major=230
mknod "/dev/vme4l_a16d16"c $major 0
mknod "/dev/vme4l_a16d32"c $major 2
mknod "/dev/vme4l_a24d16"c $major 4
mknod "/dev/vme4l_a24d16_blt"c $major 5
mknod "/dev/vme4l_a24d32"c $major 6
mknod "/dev/vme4l_a24d32_blt"c $major 7
mknod "/dev/vme4l_a32d32"c $major 8
mknod "/dev/vme4l_a32d32_blt"c $major 9
mknod "/dev/vme4l_a32d64_blt"c $major 10
```

☑ If you have built VME4L as modules, load the modules on the target:

```
target:# modprobe vme4l-pldz002
```

This will load both modules. As in the previous build option, you should see an output on the console similar to this:

```
vme4l-core $Revision: 1.4 $,  (No bridge driver attached)
vme4l_z002_init_module
vme-menpldz002: found PLDZ002 bridge (rev x), irq y
VME4L bridge driver has registered: PLDZ002 VME bridge (rev 7), \
vme4l-pldz002 $Revision: 1.4 $
```

☑ Compile the *vme4l_api* library:

```
$ cd ~/vme4l/LIBSRC/VME4L_API
```

☑ Edit *Makefile* and change "CC=" to the compiler for your target. For ELinOS, this is done automatically when the project's *ELINOS.sh* file is sourced.

```
$ make
```

This will create *~/vme4l/LIBSRC/VME4L_API/libvme4l_api.a*, which can then be linked to your application program.

1.3.1 Compiling Test Programs

A *Makefile* is provided in *TOOLS/VME4L_API* to compile test programs without the MDIS build system.

☑ Change into this directory:

```
$ cd ~/vme4l/TOOLS/VME4L_API
$ make
```

This builds all test programs and places the binaries into *~/vme4l/TOOLS/VME4L_API.*

1.4 VME4L Major Number

By default, VME4L uses major number 230. If this number is occupied by another driver in your system, either change *VME4L_MAJOR* in *vme4l-core.c* or pass "major=xxx" when loading the *men_vme4l* module.

2 RTAI Support for VME Interrupts

MDIS supports several bus types, which are specified in *oss.h* (*OSS_BUSTYPE_xxx*). VME bus devices play a special role within the MDIS kernel. Their system interrupt is not installed from the MDIS kernel, but from the *vme4l-core* module. The central function there is *vme4l_irq( )*.

Three types of interrupt routines are handled:

- Kernel RTAI IRQ handlers: `#define VME4L_RTAI_IRQ`
- Kernel standard IRQ handlers: `#define VME4L_KERNEL_IRQ`
- User IRQ handlers (signals): `#define VME4L_USER_IRQ`

The following figure explains how each of the three IRQ types is dispatched:

Figure 2. *VME IRQ Handling for Different Handler Types*