

---

# **POSITIONING AND CONTOURING CONTROL SYSTEM APCI-8001 AND APCI-8008**

## **PROGRAMMING AND REFERENCE MANUAL FOR LINUX PM Linux**

Rev. 5/112018

[www.addi-data.com](http://www.addi-data.com)



<b>1</b>	<b>Introduction .....</b>	<b>5</b>
<b>2</b>	<b>Documentation .....</b>	<b>5</b>
<b>3</b>	<b>Scope of delivery of the xPCI-800x TOOLSET software for Linux.....</b>	<b>6</b>
<b>4</b>	<b>Kernel module mcug3.[o,ko].....</b>	<b>7</b>
4.1	Creating the kernel module mcug3.[o, ko] .....	7
4.2	Installing the kernel module mcug3.[o, ko].....	7
4.3	Loading and unloading the kernel driver mcug3.[o,ko] .....	8
4.3.1	Linux Kernel 2.4 characteristics .....	8
4.3.2	Linux Kernel 2.6 und 3.x characteristics .....	8
4.4	Creating the device nodes for mcug3.[o, ko] .....	9
4.4.1	Linux Kernel 2.4 characteristics .....	9
4.4.2	Linux Kernel 2.6 und 3.x characteristics .....	9
<b>5</b>	<b>SOAP – Simple Object Access Protocol and more .....</b>	<b>10</b>
5.1	Web service mcug3Xserver – based on gsoap .....	10
5.2	mcug3Xserver – installation and configuration .....	11
<b>6</b>	<b>mcfg.exe – configuration and commissioning .....</b>	<b>12</b>
6.1	WINE – emulator for the Windows application mcfg.exe .....	12
6.2	mcug3X.dll – interface to the mcug3Xserver .....	13
6.3	Project parameter – activating the web service interface .....	13
<b>7</b>	<b>Programming xPCI-800x devices under Linux .....</b>	<b>15</b>
7.1	Driver library libmcug3.a, libmcug3.so, header file mcug3.h .....	15
7.2	Opening Linux device drivers.....	15
7.3	Examples.....	16
<b>8</b>	<b>Final comments.....</b>	<b>17</b>



# 1 Introduction

Linux is now seen as a free operating system for computers. It is released to the public under the terms and conditions of the GNU General Public License. It can be used, copied, forwarded and changed by anyone. The source texts are freely available (see also Open Source). Among other things, Linux represents an alternative to the proprietary Microsoft Windows and the commercial UNIX systems.

Deployment of the Linux operating system in the industrial environment has been increasingly sought after and requested for some time. This trend is now also being taken into account by supporting Linux software for the APCI-8001 and APCI-8008 (= APCI-800x). Although there are numerous reports and success stories on the many benefits of this highly stable, freely available and mature operating system, they are not covered in more detail below.

This PM Linux document (Programming Manual for Linux) was written for program developers. It contains the necessary steps for commissioning and configuring the TOOLSET software APCI-8008 for Linux and completes the programming with relevant examples.

Interested users or programmers should have basic knowledge of the Linux operating system and be familiar with the C programming language. Administration rights to configure the Linux kernel and to install the necessary kernel sources and kernel module drivers are also required for commissioning.

# 2 Documentation

The OM (Operating Manual), CM (Commissioning Manual) and PM (Programming Manual) documents belonging to the xPCI-800x control unit family will remain valid and should be used for commissioning and program development.

With the exception of an additional handle parameter preceding all functions [Section 7.1], the Linux API (Application Programming Interface) command set is identical to the xPCI-800x Windows API command set. Thus, the Programming Manual (PM) is also valid for the Linux programming environment. This fact is particularly interesting for users who already have experience with the xPCI-800x controller under the Windows operating system.

### 3 Scope of delivery of the xPCI-800x TOOLSET software for Linux

The scope of delivery of the xPCI-800x TOOLSET software includes several archives which are, in turn, subdivided into different parts. This concerns files with the following names:

- mcug3-linux-i686-2012-08-08.tar.gz (description in Chapters 4 and 5)
- mcug3-linux-mipsel-2012-08-08.tar.gz (description in Chapters 4 and 5)
- mcug3-windows-2012-08-08.tar.gz (description in Chapter 6)

The date in the file name represents the corresponding stage of development. In principle, the mcug3-linux-{i686, mipsel} archive comprises driver sources, libraries, gsoap-based web servers and sample programs. The mcug3-windows archive contains the commissioning and configuration program mcfg.exe, including driver DLL for web services. First, the archives must be copied to the target system and unpacked in any working directory:

- *tar -xzf mcug3-linux-i686-2012-08-08.tar.gz (target system PC)*
- *tar -xzf mcug3-linux-mipsel-2012-08-08.tar.gz (target system CANTastic, MSX-Box, APCI-8008)*
- *tar -xzf mcug3-windows-2012-08-08.tar.gz*

The various components of these unpacked archives are explained later in this document.

## 4 Kernel module mcug3.[o,ko]

A kernel module (also LKM for Loadable Kernel Module) is a special program code which is loaded during normal operation in the kernel of an operating system – in this case under Linux – or which can be removed from it again. Kernel modules are often used for device drivers, since a large selection of kernel modules for the various hardware components can be supplied with the operating system, but only the drivers which are actually needed have to be loaded in the memory.

The procedure of dynamically adding kernel modules is used for the Linux kernel in order to dynamically adapt a standard kernel to the hardware on which it is run. For example, the driver of a motion control board which has been found can be loaded while the drivers present for hardware that does not exist can be ignored and thus also do not use any main memory. Another benefit is that kernel extensions can be integrated without having to restart the operating system.

The LKM driver *mcug3.[o, ko]* under GPL (General Public License) works with all APCI-800x devices (APCI-8001 and APCI-8008) and covers the functions for initialisation, parameterisation, data exchange, command defaults and transfer of status information. The driver sources can be found in the *mcug3lkm* subdirectory of the archive mentioned above.

### 4.1 Creating the kernel module mcug3.[o, ko]

The available kernel module sources were created for the Linux operating system using kernel versions 2.4, 2.6 and 3.x. Since there is no general version of the Linux operating system available, i.e. each user can use a different version of the kernel in his Linux environment, the kernel module driver *mcug3.[o, ko]* must be created for the relevant target system. In turn, this requires the relevant kernel sources for the compilation process to be present as well. How and from where the kernel sources can be installed should be read in the documentation of the relevant Linux distribution.

Before the make process for creating the *mcug3.[o, ko]* driver can be started, few adaptations may still have to be made in the make file of the *mcug3lkm* directory. This could include the *KERNELDIR* and *INSTALLDIR* macros with kernel version 2.4. With kernel versions 2.6 however, the *KDIR* macro can be adapted where required. When all the adaptations have been made, the make process can now be started by calling the make command in the subdirectory. It is preferable for this to be carried out in a console on the development machine. If the make process runs without errors, the *mcug3.o* (Linux-2.4) or *mcug3.ko* (Linux-2.6, Linux-3.x) driver should be present in the *mcug3lkm* subdirectory at the end.

### 4.2 Installing the kernel module mcug3.[o, ko]

If the kernel module driver was successfully created as described above, this can now be incorporated in the archive of the module driver. This is done by running the *make install* command. However, you require administrator rights for this process. This driver can now be loaded in order to check the module driver. At least one xPCI-800x controller must be installed in the target system for the function test. The module load process is triggered using the *modprobe mcug3* command. The *cat /proc/mcug3* command can be run to initially check the loading process. This command should output a screen message similar to that shown below:

```

ADDI-DATA GmbH mcug3 LKM driver, version: 1.0.1.
ADDI-DATA APCI-8008 High Performance Motion Controller at
0xf75a6880 (dev 16)
PCI Registers:
0x00: 0x0180102f 0x02b00017 0x05800020 0x00004008
0x10: 0x00000000 0x00000000 0xfc000008 0x00000000
0x20: 0xfea00000 0x0000d801 0x00000000 0x04015257
0x30: 0x00000000 0x000000dc 0x00000000 0x0a020103

```

## 4.3 Loading and unloading the kernel driver mcug3.[o,ko]

The kernel module created above can be manually loaded anytime with either the command `insmod ./mcug3.[o,ko]` or `modprobe mcug3`. It is unloaded with the command `rmmod mcug3`.

### 4.3.1 Linux Kernel 2.4 characteristics

To load the kernel module, a second program called *mcug3.sh* is contained in the subdirectory “scripts” of the Linux TOOLSET software. Probably, this shell script needs to be adapted to your system environment. The script is used to load or unload the kernel module *mcug3.o*. The *mcug3.sh* shell script also supports the start of the web service with the aid of the Init V-process. According to the distribution, there are various directories, i.e. one for each run level. These directories are either directly in `/etc/` (Debian) or under `/etc/rc.d/` (Mandrake, RedHat) or under `/sbin/init.d` (SuSE), and according to the run level, they are called `rc0.d`, `rc1.d`, `rc2.d`... The script can be copied into the script directory of the corresponding distribution (e.g. `/etc/init.d`), and corresponding combinations can be generated in the desired run levels. The advantage is that the kernel driver *mcug3.o* is automatically loaded at the system start of the Linux operating system.

### 4.3.2 Linux Kernel 2.6 und 3.x characteristics

Normally, the module is automatically loaded with kernel versions 2.6 and 3.x. In the event that the automatic loading fails, the *mcug3.sh* script described in Chapter 4.3.1 can be used where required.

Here is a short description for those who are technically interested in the automatic loading of the kernel driver *mcug3.ko*:

Device drivers compiled as a module may have integrated aliases, which is the case with the *mcug3* driver. These aliases can be viewed with the command `modinfo`. Usually, they are connected with the bus-specific identification marks of a device that is supported by the driver. For example, the *mcug3* driver supports PCI devices with the manufacturer ID `0x102F` and device ID `0x0180`.

The corresponding alias is `pci:v0000102Fd00000180sv00005257sd00000401bc*sc*i*`. For most of the devices, the bus driver exports the alias of the required driver to `sysfs`. Thus, the file `/sys/bus/pci/devices/0000:02:02.0/modalias`, for example, would contain the value `pci:v0000102Fd00000180sv00005257sd00000401bc05sc80i00`. The Udev rules installed in the current Linux distributions ensure that `udev` `/sbin/modprobe` calls up `MODALIAS` with the content of the `uevent` environment variable (it should contain the same as the file `modalias` in `sysfs`). In this way, all the modules whose aliases correspond to the value are called up.



## 4.4 Creating the device nodes for mcug3.[o, ko]

In the device concept under Linux, the devices or device groups are handled in exactly the same way as files or file folders. A device (e.g. a disk drive) or a device group (e.g. xPCI-800x controller) appears, for example, as a file folder in which various devices can be found as files. A printer, a monitor, a mouse, etc. also only appear as a file or file folder. Unlike other ("normal") files, device files cannot simply be deleted or new ones created. Devices are treated as normal files and can be found after the root directory in the directory /dev (i.e. device). The name is shown as an abbreviation of the English name of the device. Under Linux, device files are referred to as "special files".

### 4.4.1 Linux Kernel 2.4 characteristics

Corresponding device files must be created for the xPCI-800x controller already installed. This can be carried out very easily by running the *mcug3.sh* script. The script can be found in the "scripts" subdirectory. The following command must be pasted at the beginning of the script: *mcug3\_cdn=y*.

After running the script with the command *./mcug3.sh start*, files similar to those shown below should have been created in the device directory:

```
tux@knoppix:~$ ls /dev/mcu* -l
crw-rw---- 1 root staff 253, 0 2005-06-10 13:27 /dev/mcug3c0
crw-rw---- 1 root staff 253, 1 2005-06-10 13:27 /dev/mcug3c1
crw-rw---- 1 root staff 253, 2 2005-06-10 13:27 /dev/mcug3c2
crw-rw---- 1 root staff 253, 3 2005-06-10 13:27 /dev/mcug3c3
crw-rw---- 1 root staff 253, 16 2005-06-10 13:27 /dev/mcug3i0
crw-rw---- 1 root staff 253, 17 2005-06-10 13:27 /dev/mcug3i1
crw-rw---- 1 root staff 253, 18 2005-06-10 13:27 /dev/mcug3i2
crw-rw---- 1 root staff 253, 19 2005-06-10 13:27 /dev/mcug3i3
```

Two different device files are created for each xPCI-800x controller (here up to 4 cards), where the type *mcug3cx* [*x* = 0 ..3] is needed for normal operation and the device files of the type *mcug3ix* are needed for interrupt operation.

### 4.4.2 Linux Kernel 2.6 und 3.x characteristics

Under Linux 2.6 and 3.0, the device nodes are created automatically to a large extent when the kernel driver is loaded. In addition, the udev rules file *92-mcug3.rules* should be copied into the directory */etc/udev/rules.d/*. This file creates symbolic links in addition to the already existing device nodes.

The following entries should then be viewable in the device directory:

```
ls /dev/mcug3* -l
lrwxrwxrwx 1 root root 9 Aug 8 15:07 /dev/mcug3c0 -> mcug3/c/0
lrwxrwxrwx 1 root root 9 Aug 8 15:07 /dev/mcug3i0 -> mcug3/i/0
ls /dev/mcug3/c/0 -l
crw-rw---T 1 root plugdev 252, 0 Aug 8 15:07 /dev/mcug3/c/0
ls /dev/mcug3/i/0 -l
crw-rw---T 1 root plugdev 252, 16 Aug 8 15:07 /dev/mcug3/i/0
```

## 5 SOAP – Simple Object Access Protocol and more

SOAP is a protocol used to exchange data between systems and to carry out remote procedure calls. SOAP is based on the services of other standards: XML to represent the data and Internet protocols in the Transport and Application layer (see TCP/IP reference model) to transfer the messages. The most widely used combination is SOAP over HTTP and TCP. Originally, SOAP was the abbreviation for Simple Object Access Protocol, but, as of version 1.2, SOAP is officially no longer an abbreviation since it is not (only) used to access objects.

### 5.1 Web service mcug3Xserver – based on gsoap

For the xPCI-800x family, a *mcug3Xserver* web service based on gsoap (<http://gsoap2.sourceforge.net>) is also included in the scope of delivery of the TOOLSET software. This web service was primarily developed for planning and commissioning the control unit and includes practically all function interfaces to the xPCI-800x API. It is used as an interface for client applications such as the *mcfg.exe* program [Section 6]. This application was created for the Windows operating system and now has such a wide range of functions that a port for the Linux operating system is currently ruled out. However, with the *mcug3Xserver* described above, the current *mcfg.exe* application can also be used for the Linux operating system. There are currently two variants for this.

Variant 1: Where a PC with an i386-compatible system architecture and graphical user interface (GUI) is used, an emulator such as *WINE* [Section 6.1] can be used to load the 32-bit *mcfg.exe* application created for Windows directly onto the Linux target system and execute it there. In this method, the *mcfg.exe* application communicates with the web service *mcug3Xserver* [Section 5.2] with the aid of a special DLL library *mcug3X.dll* [Section 6.2] and the web service, in turn, takes over the communication with the xPCI-800x control unit. The web service runs as a daemon in the background in the Linux operating system and waits in an idle state for the actions of client applications to be executed.

Variant 2 is identical to variant 1, apart from the fact that the *mcfg.exe* application is not executed on the target system, but on a second PC with a Windows or Linux operating system. This would, for example, be the case if the target system did not have a GUI or an i386-CPU based architecture.

## 5.2 mcug3Xserver – installation and configuration

The *mcug3Xserver* web service is located in the *gsoap* subdirectory of the Linux TOOLSET software. This program can be copied to any directory on the target system. To start the web service, a second program with the name *mcug3Xserver.sh*, can be found in the “scripts” subdirectory of the Linux TOOLSET software. It is assumed that this shell script still needs to be adapted to your system environment. The `DAEMON_PATH` and `LOGFILE` variables in particular should be checked! The `PORT` variable included in the script with the default value 10001 will also be needed later for the *mcfg.exe* project planning [Section 6.3].

The script is used to start or end the web service. The *mcug3Xserver* shell script also supports the start of the web service with the aid of the Init-V process. Depending on the distribution, there are different directories, one for each run level. These directories are located either in `/etc/` (Debian) or under `/etc/rc.d/` (Mandrake, RedHat) or `/sbin/init.d` (SuSE) and are called `rc0.d`, `rc1.d`, `rc2.d` ... according to the run level. The script can be copied to the script directory of the relevant distribution (e.g. `/etc/init.d`) and relevant links to the desired run levels can be created. This has the advantage that the *mcug3Xserver* web service is started automatically when the Linux OS system is started.

## 6 mcfg.exe – configuration and commissioning

The *mcfg.exe* program is described in detail in the xPCI-800x Operating Manual (OM). The *mcfg.exe* is located in the *windows* subdirectory of the Linux TOOLSET software and can be copied to any working directory on a Windows or Linux operating system. Alternatively, an *mcfg* MSI installation package as described in Chapter 6.2 can be used.

### 6.1 WINE – emulator for the Windows application mcfg.exe

*WINE*, a recursive acronym for “WINE Is Not an Emulator”, is a computer program for Linux and Unix machines. With *WINE*, it is possible to run programs which have been written for the Windows operating system on the X Windows system. *WINE* includes the source code and the documentation with examples and may be freely deployed. Portings exist for Linux, Solaris and for various BSD variants, for example. *WINE* can be used without an existing Windows installation. However, since some libraries are still not fully implemented, *WINE* provides the option of using DLLs and the registry of an existing Windows version to improve compatibility with Windows applications.

Where the *mcfg.exe* program is to be used under Linux, the *WINE* package must be installed. Current *WINE* emulators should be able to directly execute the *mcfg.exe* application without adaptations.

For earlier *WINE* versions, two additional comments can be made regarding the configuration of the *WINE* package where the paths specified here must be adapted as necessary:

- The *WINE* package is only supplied with a few fonts as these must not be forwarded in any way. One possible solution for this would be to copy the TTF font files from a Windows computer (in the Windows “Fonts” subdirectory) to the *WINE* target directory, e.g. `~/.wine/fake_windows/Windows/Fonts/`. Without these additional fonts, working with *mcfg.exe* is extremely laborious.
- If the integrated text editor of the *mcfg.exe* application is to be used, the native DLLs of the RichEdit *riched20.dll* and *riched32.dll* components should be copied from a Windows computer, from the Windows system directory to the `.wine/fake_windows/Windows/System` directory, since the RichEdit components embedded in *WINE* do not work correctly together with the *mcfg.exe* application. The `wine/config` file also needs to be adapted so that the native RichEdit components can also be used by the *WINE* package. For this, both entries below are needed in *DllOverrides*:

```
[DllOverrides]
...
"riched32" = "native"
"riched20" = "native"
```

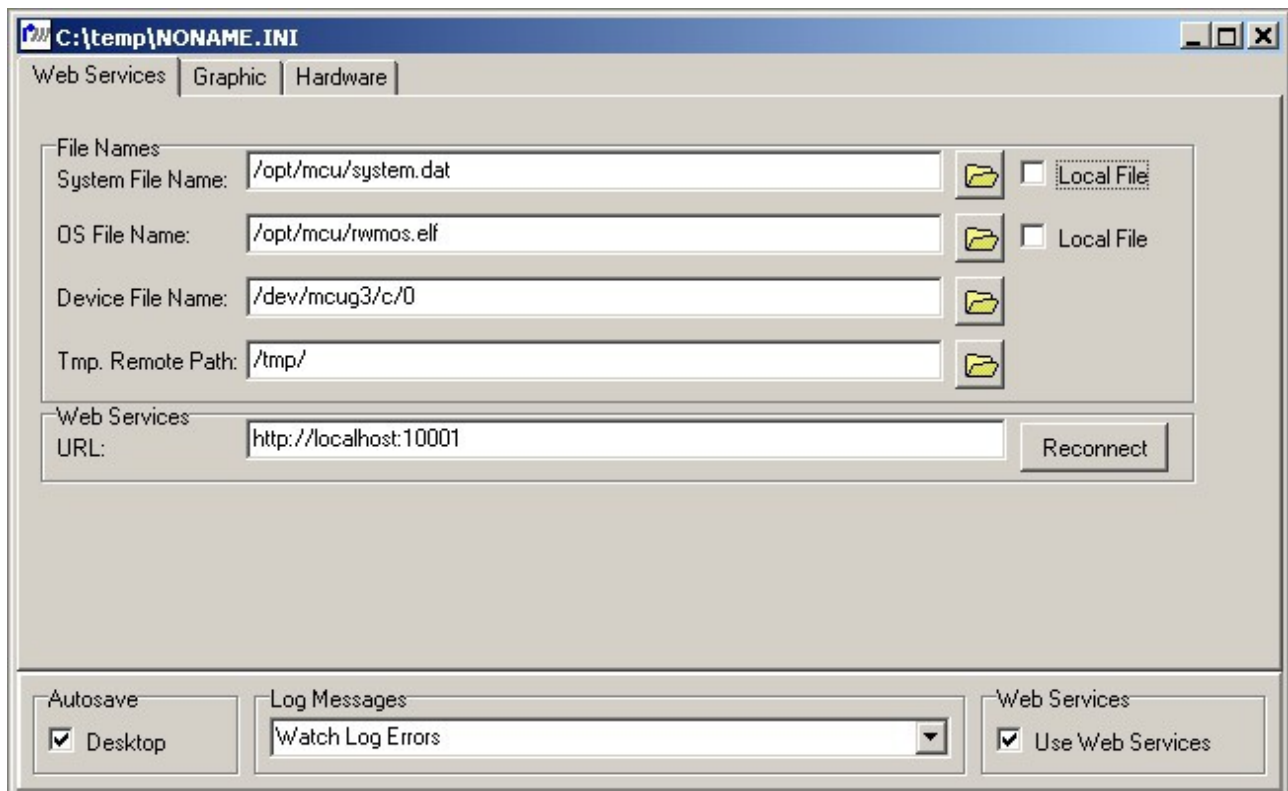
## 6.2 mcug3X.dll – interface to the mcug3Xserver

The DLL library mcug3X.dll was created to support the web service function for the xPCI-800x controller and also works with the *mcfg.exe* application, for example. If access to xPCI-800x devices is to be provided via web service methods, depending on where the *mcfg.exe* application is executed, this DLL should be copied to the system directory of the Windows machine or to the system directory *.wine/drive\_c/windows/system32* of the Linux system. The mcug3X.dll driver DLL is located in the *windows* subdirectory of the Linux TOOLSET software.

The complete package mcfg.exe + mcug3X.dll can also be installed under wine by means of an MSI setup package. In this case, the programs and files are automatically copied into the right places in the WINE environment. The current mcfg package (setup.exe) can be downloaded any time under [www.addi-data.com/downloads](http://www.addi-data.com/downloads) and can be executed with the WINE emulator. The installation is then carried out in the same way as with the normal Windows program installations.

## 6.3 Project parameter – activating the web service interface

Setting up the mcfg.exe application for remote access via the web service function takes just a few configuration steps. The configuration is explained using the graphic below:



The web service interface is configured in the [File][Project Parameter] dialog. Here, the “Use Web Services” box must first be ticked. Both fixed IP addresses and symbolic network names can be used when entering the URL. In plain language, the URL is the name of a web server on the Internet which can be used to address it. When entering the URL, the port selected (here 10001) is important as it must match that of the *mcug3Xserver* web service.

Examples:

- <http://localhost:10001>  
The web server (mcug3Xserver) runs on the local device in which the control unit is integrated.
- <http://192.168.178.98:10001>  
The web server runs on the computer with the IP address 192.168.178.98.
- <http://mcu:10001>  
The web server runs on a device in the local network.
- <http://company.org:10001>  
The web server runs on a device in the external network. There, an IP forward to a corresponding device within the network router must be set up.

The system.dat and rwmos.elf files and the device file should now be easy to select using the integrated file dialog. It should now also be possible to boot the control unit [TOOLS][SYSTEM BOOT]. A temporary path to the remote system in which temporary files can be stored must be entered in the “Tmp.Remote Path”. If local files for RWMOS.ELF or System.DAT are to be used, the relevant flag can be set. This can then be copied to the remote system in the Tmp.Remote Path as required.

## 7 Programming xPCI-800x devices under Linux

As already explained in the introduction, the programming of the devices under Linux is identical to the programming under Windows, with the exception that a handle also needs to be transferred as the first parameter for each function.

### 7.1 Driver library libmcug3.a, libmcug3.so, header file mcug3.h

An additional driver library *libmcug3.a* or *libmcug3.so* is needed to create user programs for Linux. This can be found in the *mcug3lib* subdirectory of the TOOLSET software for Linux. The *mcug3.h* header file, which lists the functions included in *libmcug3.a* [*a*, *so*], can also be found in this directory. The driver library is needed to create user mode programs and communicates with the xPCI-800x controller via the memory-mapped I/O access procedure with the help of the LKM driver *mcug3.[o, ko]*. The libraries include various functions and interfaces which cannot be forwarded under the GPL license. For this reason, the source text for *mcug3lib.a* [*a*, *so*] cannot be passed on to third parties.

The library *libmcug3.a* (archive) is required for static linking with the user application. The library *libmcug3.so* (shared object) however is required for dynamic linking and offers the advantage that various applications share the memory and therefore use the resources (Flash / Ram) as efficient as possible. Please make sure that the library *mcug3lib.so* and its symbolic combinations (*mcug3lib.so\**) are installed in the target system in the directory */usr/lib*. From revision V2.53S, the sample programs included in delivery (see Chapter 7.3) are configured for dynamic linking.

### 7.2 Opening Linux device drivers

The example below shows how the kernel module driver *mcug3.[o, ko]* is opened in order to obtain the relevant device handle for further access to the device driver.

```
/* Open and initialise MCUG3 device */
int hndl;

if (McuG3Open("dev/mcug3c0", &hndl) == 0)
{
    /* driver successfully opened. */
} else {
    /* driver open error /
    exit 1;
}
...
/* Insert your application code */
...
McuG3Close(hndl); /* close the driver */
```

## 7.3 Examples

The scope of delivery of the Linux TOOLSET software includes two sample programs for the xPCI-800x devices which can be found in the *sample01* and *sample02* sub-directories. These examples are partly based on Windows sample programs and show the simple porting and usage options for the Linux operating system.

Note: The following files and programs are required to execute the sample programs:

- *rwmos.elf* (operating system file for the xPCI-800x devices, possibly customer-specific!)
- *system.dat* (system file with configuration data for drives and machines).

These files are not part of the Linux TOOLSET software. However, they are included on the standard CD of the TOOLSET software or can be downloaded from the Internet.



## 8 Final comments

The Linux operating system opens up new and diverse options for using hardware and software. However, the configuration and administration sometimes require in-depth knowledge of the relevant system. This document was intended to remain general wherever possible and not to take the properties of the individual distributions into consideration.

If you, as the user, discover a problem or information which is incomplete, or even incorrect, we would be glad to receive your feedback so we can adapt the documents and software to the corresponding requirements as necessary. We would like to take this opportunity to thank you for your support in this matter.

Please send your support requests to: [info@addi-data.com](mailto:info@addi-data.com)