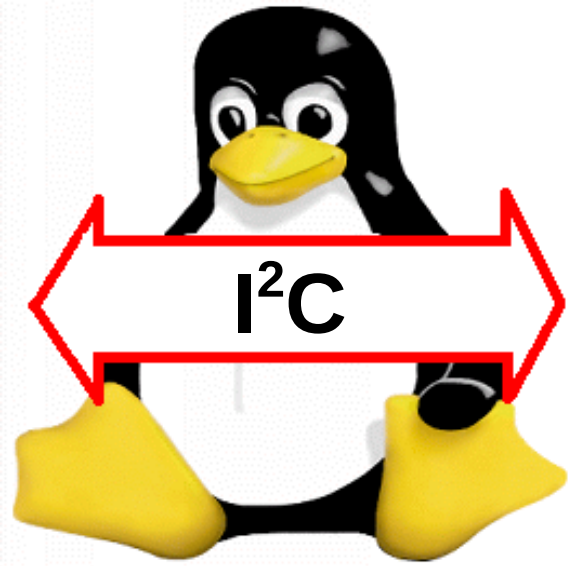
Installation Guide

iPort/USB

Virtual Communications Port Driver for Linux



MCC
Micro Computer Control
Small Area Network Specialists

www.mcc-us.com

Introduction

The MCC iPort/USB Virtual Communications Port Driver for Linux provides a quick and easy way to access a variety of I²C Bus devices from Linux (2.4). The driver is provided in several pre-compiled distribution/kernel versions for quick installation and use. Driver source is also provided. This driver is compatible with the MCC iPort/USB (#MIIC-204) I²C Bus host.

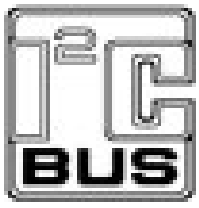
This user's guide describes downloading, installation, and loading of the iPortUSB driver module for Linux.

Are you new to I²C? Want to know more? We suggest you review "What is I²C?" at www.mcc-us.com/I2CBusTechnicalOverview.pdf.

MCC products use Philips components and are licensed to use the I²C Bus.

"Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components of the I²C system, provided the system conforms to the I²C specifications defined by Philips."

I²C is a trademark of Philips Corporation.



06-JUN-07

DISCLAIMER: Micro Computer Control Corporation makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Micro Computer Control Corporation reserves the right to revise the product described in this publication and to make changes from time to time in the content hereof without the obligation to notify any person of such revisions or changes.

WARNING - Life Support Applications: MCC products are not designed for use in life support appliances, devices, or systems where the malfunction of the product can reasonably be expected to result in a personal injury.

WARNING - Radio Frequency Emissions: This equipment can radiate levels of radio frequency energy that may cause interference to communications equipment. Operation of this equipment may cause interference with radio, television, or other communications equipment. The user is responsible for correcting such interference at the expense of the user.

WARNING - Electrostatic Discharge (ESD) Precautions: Any damage caused by Electrostatic Discharge (ESD) through inadequate earth grounding is NOT covered under the warranty of this product. See the “Electrostatic (ESD) Precautions” section of this guide for more information.

Printed in the United States of America

iPortUSB Linux Driver (V1.0.0)

www.mcc-us.com

WHAT THIS DOCUMENT CONTAINS

1. Provided Drivers

iPortUSB drivers are dependent on the Linux distribution you are running, and the kernel version installed on that distribution. MCC supports the following distributions and kernels by providing pre-compiled driver modules:

Debian GNU/Linux 3.1	kernel 2.4.27-2-686 and 2.4.27-3-686
Suse Linux 9.0	kernel 2.4.21-303
Redhat Linux 9	kernel 2.4.20-8
"Vanilla" kernels	kernel 2.4.32

"Vanilla" kernels are those that have been downloaded from www.kernel.org and compiled without any customized changes by a distribution (such as Redhat or Suse). This implies that you have compiled your own kernel.

NOTE: You must have "root" privileges to install driver modules. If you do not have “root” privileges, see your system administrator.

If your system is not supported by the above pre-compiled modules, see Section 6 below.

2. Downloading the Drivers

To download the drivers:

2.1. Visit MCC’s web site (www.mcc-us.com)
2.2. Click on the Downloads link.
2.3. Click on the iPortUSB Linux driver link.
2.4. Click to download the iPortUSB_driver100.tar.gz file.

3. Unpacking Drivers

To Unpack the downloaded driver file:

3.1. Open a command line terminal and navigate to the downloaded file folder.
3.2. Unpack the downloaded file using the following command:

```
tar -xvvzf iPortUSB_driver100.tar.gz
```

During unpacking, separate folders are created for each supported distribution and kernel version.

4. Installing Pre-Compiled Drivers

Once you determine which pre-compiled driver is best suited for your system, use a terminal window to navigate to its unpacked driver directory. Inside the driver directory will be the driver files, iPortUSB.o and usbserial.o. You must have "root" privileges to install driver modules. To install the driver, enter the follow command:

```
sh install
```

The driver can then be loaded manually by running the following commands:

```
modprobe usbserial
modprobe iPortUSB
```

If you attempt to load a module and you get an error informing you that the module is designed for a different kernel version than the one you are using, go back and select a different module.

It is also possible to configure your system such that the modules load automatically when the system starts up. To setup driver automatic loading at startup, navigate to the unpacked driver directory and running the following command:

```
sh make_perm
```

After running this command, you should reboot your computer to allow Linux to load the driver.

5. Uninstalling Drivers

Normally, the driver does not need to be uninstalled. If uninstalling is required, see you system administrator.

6. What to do if your system is not supported

If a pre-compiled driver module for your distribution and kernel is not provided, you have two options:

6.1. Install a supported kernel.

This is recommended as the modules are known to work on the systems specified. Installing a vanilla kernel is a good idea, as it can co-exist with any distribution. This option will require kernel compilation.

6.2. Compile the driver manually.

Note that the provided source code is only compliant with 2.4 series kernels. 2.6 series kernels are not supported at this time. If you wish to compile your own driver module, you must first understand that due to the myriad of possibilities of systems, COMPILING THE DRIVER MANUALLY IS NOT SUPPORTED BY MCC. If you wish to continue, navigate to the unpacked “source” directory and follow the instructions in Appendix A below.

7. Using the iPortUSB

After installing the driver (and optionally rebooting if you chose to have it load when the system boots up), you can verify that things are working by running the following command:

```
dmesg | grep "MCCI USB Adapter"
```

You should see output such as that shown below:

```
usbserial.c: USB Serial support registered for MCCI USB Adapter
```

Now connect your iPortUSB to a USB port on the computer. Verify that the device was detected by running the following command:

```
dmesg | grep "MCCI USB Adapter"
```

You should see output such as that shown below (along with the data from the previous command):

```
usbserial.c: MCCI USB Adapter converter now attached to ttyUSB0
```

This indicates that the iPortUSB was detected, and a virtual serial port named "/dev/ttyUSB0" has been configured to access it.

Appendix A. Compiling the iPortUSB Driver

NOTE 1: Compiling the iPourUSB driver module should only be considers IF none of the pre-compiled driver modules provided my MCC are suitable for your use.

NOTE 2: MCC cannot support the compiling of the iPortUSB module. This section is provided for reference only.

NOTE 3: The provided source code is only compliant with 2.4 series kernels. 2.6 series kernels are not supported at this time.

A.1. Requirements

You must satisfy thee following requirements before compiling the iPortUSB driver:

A.1.a. Linux Kernel 2.4

To determine what kernel you are running enter the command:

"uname -r" at a terminal.

The output should look something like "2.4.27-2-386" or similar. If you are running a 2.6 series kernel (i.e. "2.6.8-2-686" or similar), you cannot use this driver.

A.1.b. Install Kernel Header Files

Install the kernel header files (or the full kernel source, either one will do) from your Linux distributor. Make sure the version you are installing matches up with the kernel version from the previous step (it must match exactly!) You will not need to recompile the kernel, but you do need at least the headers. The files should unpack to a directory named something like "linux_2.4.26-8" or similar. This directory can be stored anywhere, but the usual choice is to put it in the "/usr/src" directory. Note that you will need root privileges to store files in "/usr/src".

Once the kernel header (or full source) is stored somewhere, write down the full pathname. For instance, if the kernel headers unpacked to a directory named "kernel_2.4.12-2", and you stored it in "/home/bob/randomfiles", then the full pathname would be "/home/bob/randomfiles/kernel_2.4.12-2". Write the pathname down! This is the "kernel include path".

A.1.c. GCC

This driver must be compiled with gcc-3.3. To determine whether you have this program, run the command "gcc-3.3" at the terminal. If the output looks like "gcc-3.3: no input files", you have the program. If not, you cannot compile this driver.

A.1.d. Read through this file completely BEFORE beginning!

A.2. Configuration

Enter the directory that contains the source code for the driver. Open up the file entitled "Makefile", and make the following changes:

A.2.a. MODULE_CP2101

Set MODULE_CP2101 equal to whatever you wish to call the completed module. There is no reason to change this, unless you wish to cross-compile several versions of the module.

A.2.b. MODULE_USBSERIAL

Same as MODULE_CP2101.

A.2.c. FULLKERNELPATH

Set this equal to the full pathname where you stored the kernel headers (See step 1.B above).

A.2.d. SYSTEMINCLUDEPATH

Don't change this unless you know that you need to.

A.2.e. COMPILERINCLUDEPATH

Using a terminal or file manager, enter the "/usr/lib/gcc-lib" directory. There should only be one directory inside of this one (if there's more than one, ask whoever installed your system). Inside that directory, there should be a directory named something like "3.3.6" or similar. It must be "3.3.x", where x is some other number. Inside that directory there should be a directory named "include". This is the "COMPILERINCLUDEPATH", and the full pathname should be entered into the makefile. For example, "/usr/lib/gcc-lib/i486-linux-gnu/3.3.6/include."

A.3. Compilation

A.3.a. In the directory containing the driver source code, run "make clean". This removes any past compiled files and any unneeded files.

A.3.b. Run "make". This will compile both the usb serial module and the iPort module.

A.4. Installation

A.4.a. In the "source" directory, run "sh install". This will copy the driver files to where they need to be to be found by Linux.

A.4.b. If you want the modules to load automatically when Linux boots, you need to talk to your system administrator, as this differs depending on what distribution you are using. However, if the file "/etc/modules" exists, it's a safe bet that adding "usbserial" followed by "iPortUSB" to the end of the list that it contains will do the trick.

A.4.c. Note that some distributions (big ones... SUSE, Redhat, etc.) supply usbserial themselves. Try doing a "modprobe usbserial" before compiling or installing this driver. If the operation completes successfully (i.e. "usbserial" can be found in the output of the "lsmod" command), you should remove all references to "usbserial.o" from the install script manually before continuing.