Thesycon® Systemsoftware & Consulting GmbH

USB CDC/NCM Class Driver

For Windows 2000, XP, Vista and 7

Reference Manual

Version 1.44 **March 22, 2013**

Thesycon® Systemsoftware & Consulting GmbH

Werner-von-Siemens-Str. 2 · D-98693 Ilmenau · GERMANY

Tel: +49 3677 / 8462-0

Fax: +49 3677 / 8462-18

e-mail: info @ thesycon.de

http://www.thesycon.de

Disclaimer

Trademarks

The following trade names are referenced throughout this manual:

Microsoft, Windows, Win32, Windows NT, Windows XP, Windows Vista, Windows 7 and Visual C++ are either trademarks or registered trademarks of Microsoft Corporation.

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

Contents

1 Introduction

The CDCNCM driver is a generic device driver for Windows. It creates a Network Interface. On the USB layer it expects a device with a Communication Device Class (CDC) Network Control Model (NCM) compliant interface. The device driver supports USB 2.0 with the operating speeds full and high speed.

This document describes the architecture and the features of the CDCNCM device driver. Furthermore, it includes instructions for installing and using the device driver.

The reader of this document is assumed to be familiar with the specification of the Universal Serial Bus Version 1.1 and 2.0 and the CDC/NCM specification.

2 Overview

2.1 Platforms

The CDCNCM driver package contains 32 bit and 64 bit versions. The driver supports the following operating system platforms:

- Windows 8
- Windows 7
- Windows Vista
- Windows XP
- Windows 2000
- Windows Embedded Standard 7 (WES7)
- Windows Embedded Enterprise
- Windows Embedded POSReady
- Windows Embedded Server
- Windows XP embedded
- Windows Server 2008 R2
- Windows Server 2008
- Windows Server 2003
- Windows Home Server

2.2 Features

The CDCNCM driver provides the following features:

- USB Support. The CDCNCM driver supports USB 3.0, USB 2.0 and USB 1.1. It supports low, full, high and super speed mode. For USB 3.0 controllers Microsoft has actually no bus driver released. In market are different USB 3.0 host controllers with own bus drivers. Thesycon does not guarantee that the CDCNCM runs with all solutions.
- NDIS Support. Exposes a network adapter interface according to the NDIS 5.1 model (NDIS 5.0 on Windows 2000).
- Plug&Play. The CDCNCM driver fully supports hot plug and play. It supports Plug&Play notifications for applications.
- Power Management. The driver supports the Windows power management model. The network adapter is not stopped when the PC enters standby, sleep or hibernate. A device with Auto-IP address negotiation is not required to restart the negotiation after resume from a power down state.
- Remote Wakeup Support. When the device indicates the remote wake feature in the USB configuration descriptor and when the USB port where the device is connected supports remote wakeup the device can wake up the PC. The user can modify the behaviour of the network adapter with the Power Management tab in the properties dialog of the driver.
- VLAN Support. The driver supports Virtual LAN's. In the advanced property dialog page the user can select a VLAN number. The VLAN number 0 means the feature is turned off. The driver sends untagged packets and accepts all packets. If a VLAN number grater than 0 is selected the driver accepts packets without VLAN tag and packets with the correct VLAN tag. It sends packets with the correct VLAN tag.
- MAC Address. By default, the driver uses the MAC address reported by the device. If the device cannot provide a MAC address, the driver can use a MAC address defined in the INF file. The user can assign the current MAC address in the advanced property page dialog.
- IP Address. The IP address of the adapter can be assigned via DHCP, Auto-IP, or manually.
- Multiple USB Interfaces. The NCM class driver can be used with devices that implement multiple USB interfaces. A separate network adapter instance will be created for each NCM interface. Thesycon offers a multi-interface driver that is required to build an individual device node for each interface. For more information, check out http://www.thesycon.de USB Multi Interface Driver.
- Multiple USB Devices. Multiple USB devices can be controlled by the driver at the same time.
- Customizing. The CDCNCM allows vendor- and product-specific adaptations.
- Installation/Uninstallation. For customized driver installations Thesycon offers the PnP Driver Installer. Additional information are available at **http://www.thesycon.de/pnpinstaller**.
- WHQL Certification. The driver conforms to Microsoft's Windows Driver Model (WDM) and NDIS, and it can be certified by Windows Hardware Quality Labs (WHQL) for all current 32-bit and 64-bit operating systems.

2.3 USB 2.0 support

The CDCNCM device driver supports USB 2.0 and the Enhanced Host Controller on Windows 2000, Windows XP, Windows Vista, Windows 7 and Windows 8. However, CDCNCM must be used on top of the driver stack that is provided by Microsoft. Thesycon does not guarantee that the CDCNCM driver works in conjunction with USB driver stacks provided by third parties.

Third-party drivers are available for USB 2.0 host controllers from NEC, INTEL or VIA. Because the Enhanced Host Controller hardware interface is standardized to the EHCI specification, the USB 2.0 drivers provided by Microsoft can be used with host controllers from any vendor. However, the user must ensure that these drivers are installed.

2.4 USB 3.0 Support

Currently more and more computers with Extended (USB 3.0) Host Controllers (XHC) are coming to the market. The XHC handles full and high speed data traffic. If a customer connects your CDCNCM device to a USB 3.0 host controller connector it will run with an XHC and the installed bus driver stack.

Microsoft has released the USB 3.0 bus driver for XHCs with Windows 8. Other vendors of USB 3.0 controllers provide their own driver stacks for older operating systems. It is nearly impossible to test with all of these controllers and versions of USB driver stacks.

Thesycon tests using the Renesas and Intel controllers with the vendor-provided driver stacks on Windows 7 and the Microsoft-provided driver stack on Windows 8. The CDCNCM driver can also work with USB 3.0 controllers from other vendors, but Thesycon cannot give a warranty that this works without problems.

Today, most devices with a USB 3.0 super speed interface are mass storage devices. They use system-provided drivers. Other devices with a USB 3.0 interface are rare on the market. For that reason it is difficult to make comprehensive tests with USB 3.0 devices at this moment in time. The CDCNCM driver is designed to support USB 3.0 devices with super speed, however because of the test situation, Thesycon cannot give warranty that it works in every case. If you intend to plan a project using a USB 3.0 device please contact Thesycon (www.thesycon.de).

2.5 Known Issues

2.5.1 Driver is not loaded after disconnect

The problem occurs on Windows 7, Windows Vista and Windows XP 32/64 bit. When the device is disconnected from the USB bus and reconnected in a short time period the driver is not loaded by the system. In the device manager the device node is marked with error code 31. The network interface is not available. Debug output from the driver shows clearly that the driver is stopped and unload when the device is removed. But it is not loaded when the device is re-connected in the problem situation.

It seems to be a problem in the Microsoft NDIS driver framework that it is not able to handle a device add event correctly when the device remove operation is not yet completed.

A workaround for this problem is to delay the soft connect in the device for some seconds. Normally the device connects to the USB bus by turning on the connect resistor as soon as it is ready

to handle USB requests. The workaround means that the device delays this connect operation. It is difficult to determine exactly how long the device must be disconnected to avoid the problem. The time period depends on the host hardware, on the load situation when the device is removed as well as on the software that is installed on the PC. The error situation is also resolved when the device is removed and re-connected to the PC a second time.

3 Architecture

Figure 1 shows the USB driver stack that is part of the Windows operating system. All drivers are embedded within the WDM layered architecture.

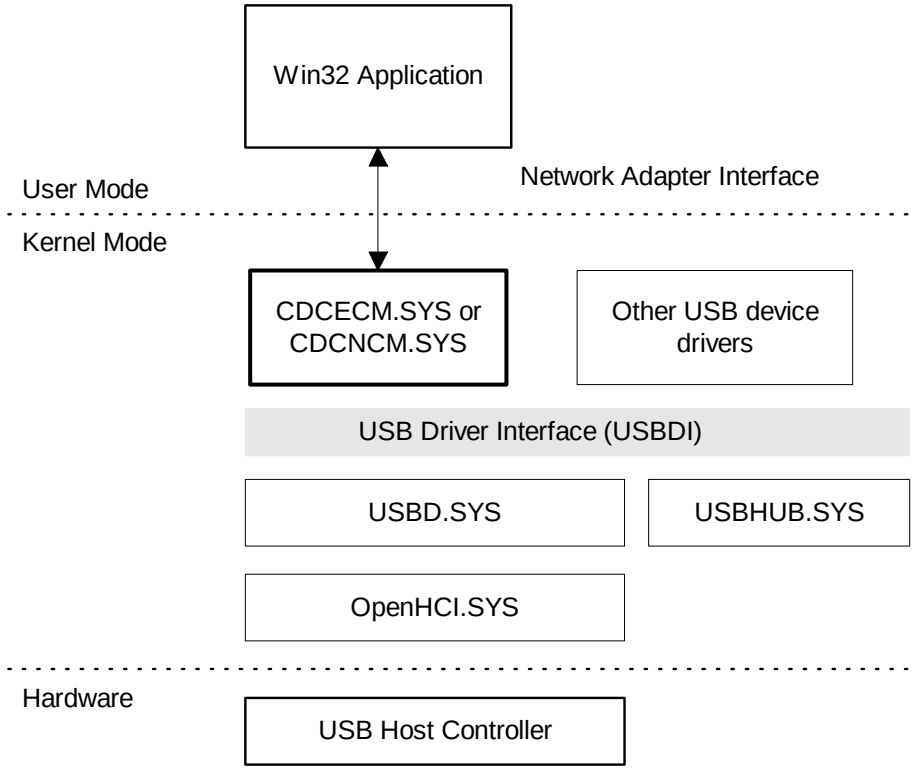

Figure 1: USB Driver Stack

The following modules are shown in Figure 1:

- USB Host Controller is the hardware component that controls the Universal Serial Bus. It also contains the USB Root Hub. There are two implementations of the host controller that support USB full speed: Open Host Controller (OHC) and Universal Host Controller (UHC). There is one implementation of the host controller that supports USB high speed: Enhanced Host Controller (EHC).
- OpenHCI.SYS is the host controller driver for controllers that conform with the Open Host Controller Interface specification. Optionally, it can be replaced by a driver for a controller that is compliant with UHCI (Universal Host Controller Interface) or EHCI (Enhanced Host Controller Interface). Which driver is used depends on the main board chip set of the computer. For instance, Intel chip sets contain Enhanced Host Controllers and Universal Host Controllers.
- USBD.SYS is the USB Bus Driver that controls and manages all devices connected to the USB. It is provided by Microsoft as part of the operating system.
- USBHUB.SYS is the USB Hub Driver. It is responsible for managing and controlling USB Hubs.

- The CDCNCM.SYS is a kernel mode driver that supports the CDC/NCM protocol and provides a Network Adapter Interface.

The software interface that is provided by the operating system for use by USB device drivers is called USB Driver Interface (USBDI). It is exported by the USBD at the top of the driver stack. USBDI is an IRP-based interface. This means that each individual request is packaged into an I/O request packet (IRP), a data structure that is defined by WDM. The I/O request packets are passed to the next driver in the stack for processing and will return to the caller after completion.

3.1 Special Properties of the Driver

3.1.1 NDIS Model

Microsoft supports different versions of the NDIS model. On Windows 2000 the NDIS version 5.0 is supported. Drivers that are designed for NDIS version 5.1 cannot run on Windows 2000. On the other hand NDIS version 5.1 provides some new features on Windows XP and later. For that reason we provides two build versions of the driver. The driver for Windows 2000 has the extension 'w2k' and is compiled with the NDIS 5.0 model. The driver for Windows XP and better is compiled with the NDIS 5.1 model.

3.1.2 Surprise Removal Dialog

Windows can show plug and play devices on the PnP card in the systray and can complain if such devices are removed from the system without stopping it. The driver suppresses the entry in the systray. The device can be removed from the PC without any preparation.

3.1.3 Power Management

The driver supports the requests for power management. For that reason the adapter is not stopped if the PC enters standby, sleep or hibernate. This has the advantage that a device with AutoIP address negotiation must not restart the negotiation after the power down state of the PC. It is not recommended to use AutoIP because it takes a long time before Windows assigns a valid IP address.

3.1.4 MAC Address

The driver determines the valid network address (MAC Address) in the following way:

1. The driver reads the network address from the device. If this fails
2. The driver reads the network address from the registry. If this fails
3. The driver has a build in MAC address with the value AE:DE:48:02:01:00. This address has set the local bit and is not neccessary unique.

3.1.5 Statistic Counters

The driver does not request the statistic counters from the device. It counts relevant events by it's self.

3.1.6 Multiple interfaces on one device

The CDCNCM driver can be used with devices that implement multiple USB interfaces. In this case a multi-interface driver is required. This driver splits the interfaces to separate device nodes. On each device node a new driver stack can be installed.

The system provided multi-interface driver before Windows XP SP3 cannot be used together with the CDCNCM driver. The reason is that this CDCNCM interface consists of two USB interfaces. This two interfaces must be mapped to one device node. The system supplied multi-interface driver is not able to detect USB interfaces that belong together. Later versions of the system provided MI driver can handle the IAD. You should add an IAD to your configuration descriptor.

Thesycon provides a special multi-interface driver that can handle this problem in a correct way. This driver is required if you have to support systems before Windows XP SP3. This driver can be licensed separately.

4 Driver Customization

4.1 Customization Overview

The CDCNCM device driver supports various features that enable a licensee to create a customized device driver package to be shipped with an end product. Customization includes:

- Modification of the file name of the driver executable,
- Modification of text strings shown at the Windows user interface,
- Definition of a unique software interface identifier,
- Adaptation of driver behavior for a specific device.

Note that the driver package which is shipped to end users has to be customized. This is required in order to avoid potential conflicts with other products of other vendors that are also using the CDCNCM device driver. Please consider the following example scenario: An end user buys product A which includes the CDCNCM device driver version 1.15. The user installs this driver on its machine. At a later point in time the user buys another product of another vendor which is called B. Product B includes the CDCNCM device driver version 1.20. If the user installs the device driver for Product B on the same machine then a conflict will occur because another version of the driver is already installed on that machine.

There are several problems that may result from this conflict situation:

- **Driver version conflict**
  We assume that during driver installation any existing device driver will be removed if the existing driver has an older version than the driver to be installed. If the existing driver is newer then no driver will be installed. In the example scenario described above this means: When product B is installed the existing driver (V1.15) will be replaced by a newer one (V1.20). The result is that both product A and product B now use the new driver. This should be fine for product B. However, product A will now run with driver version 1.20 which is critical because probably the product was never tested with that version. Thus, installation of a new product can break an existing installation of another product.
- **Driver software interface ambiguity**
  If two or more different products (of different vendors) use the same driver then a conflict can arise when Windows applications open the device driver to communicate with the device. In the example scenario described above an application designed for product A could inadvertently open device B and try to configure the hardware which will probably not work.
- **Device naming ambiguity**
  If two or more different products (of different vendors) use the same driver then a device name conflict could occur. Particularly, this applies to device names displayed in Device Manager. In the example scenario described above an end user could get confused if the item shown in Device Manager for product A is named identically to the item shown for product B.
- **Uninstallation**
  If one of the driver would be uninstalled the driver file may be removed. This causes the other product to become non-functional.

The customization features of the CDCNCM driver enable you to avoid all of these conflict situations. A customized driver can be considered as a specific driver for a specific product. There will be no overlaps with (customized) CDCNCM drivers shipped with other products of other vendors. In the example scenario described above, if both product A and product B are shipped with a customized driver then there will be two separate driver installations on the end user's machine. There will be two sets of driver files on hard disk and two separate drivers loaded into memory.

Note that it is possible to create a customized driver package which supports several products with similar properties, e.g. a product family of a vendor. In this case, if several products of the family are used on one machine, there will be only one set of driver files on hard disk and only one driver executable loaded into memory. A vendor can decide which of its products will be supported by a particular driver package and how many different driver packages need to be created.

To summarize the customization strategy the following rules are given:

1. A driver package provided to end users should always contain a customized driver. Do not ship the original driver provided by Thesycon together with your products.
2. If you offer a family of products, you may create a customized driver package that supports all products of this family. Windows applications shipped with the driver should be designed in such a way that all products of the family are supported. If an updated driver is delivered to end users, either as a software-only package or as part of a new product of the family, then you have to ensure that the new driver version works with all released products of the family.

In the following sections, the customization procedure is described in detail.

4.2 Preparing Customization

Thesycon provides a set of batch scripts and tools these support generation of customized driver packages.

Before the scripts can be used, the following steps are to be executed on the build machine:

1. Install SignTools
   The following Microsoft tools are required:

   - signtools.exe (part of WDK)
   - inf2cat.exe (part of WDK and "Winqual submission tools")
   - cross certificate files (available on the Microsoft Web Site)

   Create a directory called `SignTools` on your system. This must be located at a local drive. Some of the executables do not run correctly on a shared network folder.

   Locate all EXE files at a subdirectory called `SignTools\bin`. Locate the cross certificate in subdirectory called `SignTools\crosscert`. For more information about cross certificates have a look at section 4.6 on page 26. Create an environment variable called `SIGNTOOLS` and set this to the directory `SignTools`.

   For your convenience a SignTools installer can be requested on Thesycon. This will install all required tools into a directory of your choice and set the SIGNTOOLS environment variable to point to this directory.

2. Validate your Vendor Certificate
   Check if your Vendor Certificate is present under the Personal folder by launching the Certificates Microsoft Management Console (MMC). To do this, click Start - Run and enter `certmgr.msc`. The Personal folder should look as shown in figure 2.
   Please contact the publisher of your Vendor Certificate for information how to install the certificate.

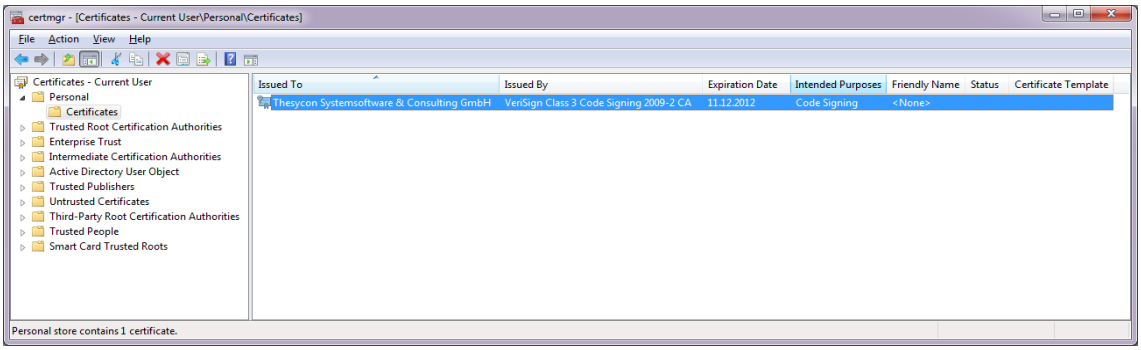

Figure 2: Certificates Microsoft Management Console

To verify whether the certificate is valid for Code Signing double-click on the certificate to show Certificate Information. The Certificate Information should look as shown in figure 3. For code signing it is required that the private key corresponds to the available certificate. This is indicated by the key sign at the bottom of the Certificate Information General page.

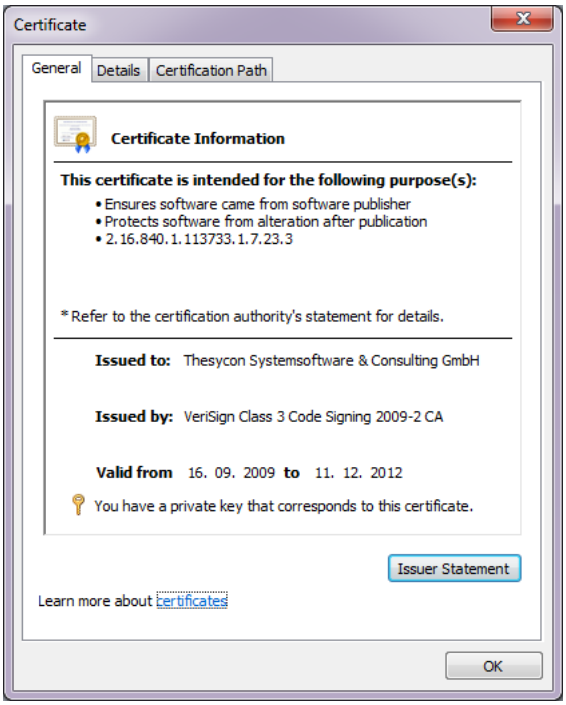

Figure 3: Certificate Information

3. Set Certificate Variables
   Edit the `set_vendor_certificate.cmd` script (it's located in the subdirectory `CustomPackageBuilder` of the driver kit) to specify the vendor certificate imported in the previous step. Set the following variables according to your certificate:

   - VENDOR_CERTIFICATE
     the "issued to" name of the certificate as shown in certmgr.msc
   - CROSS_CERTIFICATE
     file name of the cross certificate for your certificate provider
     (in case of a VeriSign certificate, keep the default)
     more information about cross certificates see at section 4.6 on page 26
   - SIGNTOOL_TIMESTAMP_URL
     URL of a trusted timestamp server
     (in case of a VeriSign certificate, keep the default)

   For details, please refer to the comments in `set_vendor_certificate.cmd`.
   As an alternative you can add these variables to your machine's environment.

4. Prepare for GUID Generation
   GUIDs are generated using the `guidgen.exe` tool provided by Microsoft. The tool `guidgen.exe` is part of Microsoft Visual Studio 2005 and Microsoft Visual Studio 2008. Alternatively, the tool can be downloaded at:
   http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=17252

4.3 Customization Steps

After preparing the customization as described in section 4.2 on page 19 the customization can be started. Below, the steps needed to create a customized driver package are summarized. Some of these steps are required and some are optional.

1. **Required:** Copy driver
   Define a driver *Package Identifier*, e.g. Productname. This should be used as argument for all command files delivered by Thesycon. A subdirectory with this name will be created by the scripts as working directory for customization.

   Make a private copy of the SYS and INF files in a subdirectory of `CustomPackageBuilder` directory. Use *Package Identifier* as name of this subdirectory. Adapt the `set_w2k_support.cmd` for your requirements. If you need Windows 2000 drivers you has to set **W2K_SUPPORT** to 1 else you can set it to 0. Use the `create_new_package.cmd` with *Package Identifier* as parameter to copy correct files. This command file creates all necessary subdirectories. The unsigned driver files (SYS) from `Drv_unsigned` and the template setup information files (INF) from directory `idisk` are copied.

   To verify check that the 64 bit files (extension `_x64`) are located in subdirectory `x64`. The 32 bit files (no extension) are located in subdirectory `x86`. The files for Windows 2000 (extension `_w2k`) are located in subdirectory `w2k`.

2. **Required:** Rename driver.
   Choose a new name for the driver. The complete driver package (Windows 2000 drivers included) consists of three parts:

   - supported Winodws 32 bit Editions: cdcncm.sys and cdcncm.inf
   - supported Windows 64 bit Editions: cdcncm_x64.sys and cdcncm_x64.inf
   - supported Windows 2000 Edition: cdcncm_w2k.sys and cdcncm_w2k.inf

   The new names must not contain spaces and must not cause conflicts with drivers included with Windows.

   Rename all files to your new names. Edit the renamed INF files to contain the new driver name. See section 4.4.1 on page 22 for detailed information.

3. Edit your INF file.

   - **Required:** Edit hardware IDs
     Insert the correct hardware ID for your device. Refer to section 4.4.2 on page 23 for more information.
   - **Recommended:** Update driver version.
     Adapt driver version and release date to your needs. See section 4.4.3 on page 24 for more information.
   - **Optional:** Adapt default driver settings.
     Refer to section 4.4.4 on page 24 for more information.

4. **Optional:** Edit version resources.
   These are contained in cdcncm.sys/cdcncm_x64.sys/cdcncm_w2k.sys (further CDCNCM SYS). See section 4.5 on page 26 for more information.

5. **Optional:** Sign the new driver package.
   Sign package with your digital signature. See section 4.6 on page 26 for more information.

4.4 Customizing INF files

The cdcncm.inf/cdcncm_x64.inf/cdcncm_w2k.inf (further CDCNCM INF) file is used to install the kernel-mode device driver CDCNCM SYS file. It has to conform to INF file conventions defined for WDM drivers. Refer to the Windows DDK documentation [5] for more information about INF files.

The INF files itself can be renamed to any name of your choice. However, the file name extension has to be .inf.

4.4.1 Configuration of Names

In CDCNCM INF there is a Strings section that permits to define the driver name and some text strings that will be shown at the Windows user interface.

```
[Strings]
S_Provider="Thesycon"
S_Mfg="Thesycon"
S_DiskName="CDCNCM driver disk"
S_DeviceDesc="Thesycon CDC/NCM network adapter"
S_DriverName="cdcncm"
S_ServiceName="cdcncm"
```

`S_Provider`
This variable defines the provider of the driver. You should use your company's name here.

`S_Mfg`
This variable defines the manufacturer of the driver. You should use your company's name here.

`S_DeviceDesc`
This variable defines the device description which is shown during installation. When driver installation is finished the device description is shown in Device Manager next to the item representing your device. You should use your product's name here. The name can contain spaces but the size is limited to 32 characters.

`S_DiskName`
This variable defines the name of your installation disk. It should correspond to the label that is printed on the disk. The disk name is displayed by the operating system if it requests the user to insert the installation media.

`S_DriverName`
This variable defines the name of the CDCNCM driver executable which is CDCNCM SYS by default. Note that the name is given without the .sys extension. You have to set this value to the file name you want to use for CDCNCM SYS file. It is strongly recommended to use a vendor-specific driver name in order to avoid file naming conflicts with other drivers.

Important: The driver name must not contain spaces. If you modify `S_DriverName` then you

have to modify the following sections as well:
`[_CopyFiles_sys]`, `[SourceDisksFiles]`.
Due to technical limitations, these sections cannot use the `S_DriverName` variable to specify the driver executable. Therefore, you have to edit them as well. Do a search for cdcncm to locate the lines to be modified.

`S_ServiceName`
The service name must be unique on one PC as well as the driver file name. It should contain a name equal or similar to the driver file name without any extension. Use a name without spaces and special characters.

4.4.2 Configuration of Hardware ID

Your USB device is initially enumerated by the system-provided USB bus driver. The bus driver uses vendor and product ID's from the device descriptor to create a unique hardware ID string according to the following scheme:

```
USB\VID_VVVV&PID_PPPP
```

The fields `VVVV` and `PPPP` will be replaced by the hexadecimal form of the vendor ID and the product ID. You can check the resulting hardware IDs by looking at the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB
```

Additionally the hardware ID is displayed in the device manager under Windows XP SP2 and higher under Properties -> Details -> Hardware ID.

The cdcncm(_x64).inf file needs to specify the resulting hardware ID for the device within the following section:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_VVVV&PID_PPPP
```

Replace `VVVV` and `PPPP` by your specific values, for example:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001
```

Windows can distinguish between different device release numbers. The device release number is part of the device descriptor in the field `bcdDevice`. If the INF file should work with one special device release number of a device the hardware ID can be specified in the following way:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&REV_ZZZZ
```

The parameter `ZZZZ` is the hexadecimal value of the `bcdDevice` value.

If your device contains more than one USB interface and the CDCNCM driver should be loaded on an interface of your device the following hardware ID must be used:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&MI_00
```

The two digits after MI are the USB interface number. If you want to load the driver for more than one interface you have to add a hardware ID for each interface, e.g.:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&MI_00
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&MI_02
```

4.4.3 Update of the Driver Version

The key word `DriverVer` is used in the Version section or in the DD Install section of the INF file. It contains the version of the driver and the release date. The release date should be modified to the release date of the customized version. Please take care of the right date format (mm/dd/yyyy).

4.4.4 Customizing Default Driver Settings

The INF file specifies some settings that define the default behavior of the driver. These settings are defined in the following section.

```
[_AddReg_SW]

HKR,,PnPCapabilities,%REG_DWORD%, 0
HKR,,RxBuffers,%REG_DWORD%, 10
HKR,,TxBuffers,%REG_DWORD%, 10
HKR,,NetworkAddress,%REG_SZ%, "AEDE48020100"
HKR,,DefaultMediaState,%REG_DWORD%, 1
HKR,,DefaultLinkSpeed,%REG_DWORD%,1000000
HKR,,OneByteTermination,%REG_DWORD%, 0
HKR,,ConfigurationIndex,%REG_DWORD%, 0
HKR,,SendClearEpHaltOnStart,%REG_DWORD%, 0
HKR,,StatisticsUpdateInterval,%REG_DWORD%, 5000
HKR,,VendorDescription,%REG_SZ%, %S_VendorDescription%
HKR,,TxTimeout,%REG_DWORD%, 5000
HKR,,VlanEnabled,%REG_DWORD%, 0
HKR,,NcmTxPeriod,%REG_DWORD%, 1
```

`PnPCapabilities`
This parameter defines the default settings of the power management tab. A value of 0 means the adapter can wake the PC. If the bit 0x10 is set the device is not allowed to wake up the PC. If the flag 0x100 is set the check box for "Allow magic pattern only" is checked. When the device does not support remote wakeup the check box for wakeup is disabled and grey.

`RxBuffers`
This parameter defines number of read buffers. The default value is 40.

`TxBuffers`
This parameter defines number of write buffers. The default value is 40.

`NetworkAddress`
With this parameter a Network address can be assigned when the device is not able to report one.

When the device reports a Network address the address of the device is used. **Note:** You have to use a valid MAC address. Do not use the example above.

`DefaultMediaState`
If this parameter is 1 the driver reports the media state as connected. The PC can sends immediately requests to the device. If this parameter is set to 0 the driver waits on a notification from the device that the media state is changed to connected. Default is 1.

`DefaultLinkSpeed`
This parameter can be used to provide a default link speed. It is overwritten as soon as the device reports the link speed the first time. It seems that some protocol driver like PPPoE request the link speed even if the adapter is not connected. The value is in units of 100 bps.

`OneByteTermination`
If this parameter is 1 the PC driver terminates a packet that can be divided by the FIFO size without rest with an additional byte. If this parameter is zero the driver terminates such a packet with a zero length packet. Set this parameter to 1 if your hardware cannot handle zero length packets. The default value is 0. Note: Linux use the same algorithm.

`ConfigurationIndex`
Is the zero based USB configuration index that is activated by the driver. Some Linux gadget implementations exposes the CDC/NCM interface on index 1.

`SendClearEpHaltOnStart`
If this value is different from 0 the driver sends the standard command clear feature endpoint stall on each data pipe before the data transfer is started.

`StatisticsUpdateInterval`
This value is used to update the Ethernet statistics when the device supports it. To save bandwidth the driver requests only the parameters that are request by the PC.

`VendorDescription`
This is the vendor description reported to the NDIS layer.

`TxTimeout`
TxTimeout is the period in milliseconds the driver expect that at least one TX packet has been transferred to the adapter. When the timeout expires the driver assumes a hardware failure in the device and indicates the problem to the system. A value of 0 turns the feature off.

`VlanEnabled`
This parameter is 0 or 1, default is 0. When this parameter is set to 1 the driver reserves 4 bytes space for the VLAN tag. It reports a 4 bytes smaller payload size to the NDIS as reported in the ECM descriptor with wMaxSegmentSize. The parameter wMaxSegmentSize is always the largest USB transfer accepted by the driver. ă

`NcmTxPeriod`
NcmTxPeriod is the time period the driver is waiting in NCM mode to receive further TX packets. The packets are submitted when the NCM buffer is full or when the timeout expires. Note that

the timer granularity in windows is about 16 ms. When no Tx data transfer is in progress the first Ethernet packet is sent in each case immediately. This improves responds times. Default value is 1 millisecond.

4.5 Customizing Version Resources

The CDCNCM SYS executable includes version resources. These information will be shown in Device Manager on the Driver Details dialog page or on the file's property page. You may want to modify the version resources to include your product's name or to modify copyright information. This can be done in a two-step procedure as described below.

1. Make a private copy of CDCNCM SYS. Use the unsigned file from the directory `Drv_unsigned`. Rename the private copy of the CDCNCM SYS file and rename the INF file in the same way. Modifications on a signed file lead to a not usable driver file. Use Visual Studio to open the private copy of CDCNCM SYS in resource mode. You have to select Open as Resources in the File Open dialog. Edit the version resources according to your preferences and save the modified file.
2. Open a Command Prompt window and run the UpdateChecksum.exe tool on the modified file. You have to enter the following command line (example uses the original file name):
   ```
   UpdateChecksum cdcncm.sys
   ```
   The program UpdateChecksum.exe is part of the CDCNCM development kit. You can find it on the `Tools` directory.

4.6 Digital Signature since release of Windows Vista

Since release of Windows Vista a new feature to verify a vendor of a software component is fully integrated in Windows operating system. The vendor can add a signature to a software component to identify itself. This signature grants that the software was signed by the vendor and that the software was not modified after it was signed. If a signed Plug&Play driver is installed the first time Windows shows the vendor name and the user can choose between

- Abort the installation,
- Allow the installation this time or
- Always trust the vendor and allow installations from this vendor.

If the user selects "always trust this vendor" the certificate of the vendor is stored in the certification manager under Trusted Vendors. If later a Plug&Play driver of this vendor is installed again, the installation can be performed silently without user interactions if the driver was pre-installed before the device is connected.

On x64 Editions it is required that the driver has a digital signature. Otherwise, the driver is not loaded on a normal system. To test a driver without signature on a x64 system the driver signing enforcement can be turned off in the boot options menu. But the setting is valid for one boot process, only. An other way to load driver without signature the system can run with a kernel debugger. On x86 Editions a driver without signature can be installed.

The signature of a driver is not the same as a WHQL certification. A driver can have a digital vendor signature without passing any WHQL tests.

A signature for a Plug&Play driver is part of a CAT file. It is also possible to add a signature to the SYS file. If the system runs in normal mode the signature is expected on the CAT file referenced by the INF file. Please note! Attaching a signature to a SYS file will modify this. The CRC of such a file is modified too. So a previously generated CAT file contains an invalid CRC.

On Windows 8 x64 the installation of the driver requires a valid signed CAT file.

Why can Thesycon not deliver a signed driver package? The signature becomes invalid if the driver or the INF file are modified. To install the driver on any device a customized INF file must be generated. At least the USB vendor and product ID's must match the ID's of the device. The modification of these ID's would make a signature from Thesycon invalid.

Why Thesycon cannot deliver all the tools required to create the signature? Microsoft does not allow re-distribution of the signing tools.

Why the CDCNCM driver should be signed? A signed driver contains information about the vendor. It grants that the driver is not modified after product release.

The following sections will guide you through the process of obtaining an "Authenticode Digital ID", to maintain it, to get the tools required for signing and to create a signature for a Plug&Play driver. It is strongly recommended to follow these steps. It is not possible to perform the code signing on a Windows 2000 system. Windows XP or better is required.

To get more information about code signing, read the document "Kernel-Mode Code Signing Walkthrough" available on the Microsoft web site.

4.6.1 Get an Authenticode Digital ID

A digital signature key contains some information about the owner. The entity and correctness of the information guarantees a so called Certification Authority (CA). Microsoft accepts a number of CA's for the Authenticode technology. The company VerySign, now Symantec, is one of them.

A signature consists of a private and a public key. This keys can be delivered in separate files or as a package. How they are delivered depends of the CA.

At first you have to buy a signature key pair from one of the CA's. The key is valid for a given time interval. You can select the time interval during the order of the key pair. The time interval means that the key can be used in this interval to create new signatures. A signature that was created with a timestamp is still valid after the key has been expired.

You may read the Microsoft document "Code-Signing Best Practices" to get more information how to maintain the key.

The key must be stored in the Certificate Manager under Personal\Certificates to use it later. The way to get this certificate installed depends on your certificate provider. Please follow the instructions that are given by the certificate provider to install your certificate properly. To validate the correct location of your certificate refer to section 4.2 point 2 on page 19.

If certificate is not correct installed on the computer contact the certificate provider for help.

4.6.2 Get the Tools for Code Signing

To create a CAT file the program inf2cat.exe is required. This file is part of the WDK and the "Winqual submission tools". Please download one of these packages from the Microsoft web site and extract it. The inf2cat.exe program requires the DLL's that starts with "Microsoft.Whos...". These are part of the package.

The signtool.exe is required to attach the signature to the CAT file. This program exists in different versions with the same file name. Only the version that is part of the WDK is sufficient to create a signature for a kernel mode driver. So you have to install the WDK. You will find the signtool program under `bin\SelfSign`. Please note! Versions that are delivered with the .NET framework and the SDK cannot be used to sign kernel mode drivers.

Kernel mode driver signing requires a cross certificate. This cross certificate must be downloaded at the Microsoft Web site "Cross-Certificates for Kernel Mode Code Signing". Please select the correct cross certificate for the CA.

For your convenience a SignTools installer can be requested on Thesycon. This will install all required tools into a directory of your choice and set the SIGNTOOLS environment variable to point to this directory.

4.6.3 Create a Signature with Provided Scripts

The required scripts are located in the folder `CustomPackageBuilder`. The scripts are using relative pathes. For that reason the script folder must not be moved autside the driver package. The customized driver package is created in a subfolder of `CustomPackageBuilder`. This folder name is the argument to all scripts. The scripts are designed to use the SignTools installer from Thesycon. Make sure the SignTools are installed correctly. The evironment variables `SIGNTOOLS` must be set to a local folder on your hard disk. This environment variable is set by the SignTolls installer. Open a command window when running the scripts to see the output of the operations and to pass the driver package folder to the scripts.

With the script `create_new_package.cmd` a new subfolder is created and the SYS and INF files are copied into this folder. After the copy operation the files must be customized. Please refer to section 4.3 for details.

Adapt the file `set_vendor_certificate.cmd` to the used certificate. Set the correct time stamp server and the cross certificate. To do this refer to section 4.2 point 3 on page 20.

Create the CAT files with the script `generate_CAT.cmd`. This command file creates CAT files for the 32bit and the 64bit driver version. Each time the INF or SYS file is modified the CAT files must be created and signed again. To limit the supported operating systems the environment variables `OSLIST_x86` and/or `OSLIST_x64` in the command file can be adapted.

The signature for the CAT file is created with the script `sign_CAT.cmd`. This command file signs both CAT files (32- and 64-bit version) with the certificate defined in `set_vendor_certificate.cmd`. Processing of this file requires access to the Internet to get the timestamp from timestamp server. Make sure that the program writes the line "Successfully signed and timestamped" on the console.

Finally the correct certification can be checked with command file `verify_signature.cmd`. Before the verification can be started the command file has to be adapted to the driver package. For adaptation the variables `CAT_FILE_NAME_x86` and `CAT_FILE_NAME_x64` has to be fit

to the customized driver package. Use the CAT file names without extension to change the entries in the file. The package identifier has to be used as argument to this script. This file verifies the signature of CAT files first and list

- Signing Certificate Chain
  Shows verification chain of your certificate
- Timestamp Verification
  Shows time of file signing and verification chain for timestamp
- Cross Certificate Chain
  Shows verification chain of used cross certificate, this chain must start at a `Microsoft Root Cert` like `Microsoft Code Verification Root`.
- In the last step the script file verifies SYS and INF files referenced in CAT.

If no problem is shown the driver should be able to install. The most common error occurs if the cross certificate is not added correctly.

4.6.4 Modified System Behavior

If the driver package has a valid signature from a vendor the system behavior is modified in the following way:

On Windows 2000 and Windows XP the property page of the device manager shows that the driver does not has a valid signature. Both operating systems cannot validate the signature because the trusted chain of the certificates is not completely stored in the operating system. Both systems accept only a signature created by Microsoft during a WHQL certification process.

Since release of Windows Vista during the driver installation a message shows the name of vendor that has signed the driver package. The user can select if the driver package should be installed or if the installation should be aborted. If the certificate of the vendor is stored in the Trusted Vendors section of the certificate manager and if the driver is pre-installed the driver is installed silent since Windows Vista is released.

On the x64 version of the operating systems Windows Vista, Windows 7 and Windows 8 a digitally signed driver package is required. Otherwise the driver cannot be installed.

5 Driver Installation and Uninstallation

This section discusses topics relating to the installation and un-installation of the CDCNCM device driver.

5.1 Driver Installation for Developers

This section describes some methods how developers can install and un-install the CDCNCM driver on a PC.

5.1.1 Installing CDCNCM Driver Manually

The manually installation of driver should be used by developers, only.

In order to install the CDCNCM driver manually you have to prepare an INF file that matches your device.

The steps required to install the driver are described below.

- Copy the .inf, .sys and .cat files that are required for the used operating system to the hard disk. Use a folder like
  `c:\program files\<Vendor>\<Product>\<Version>\drivers`.
  Do not copy the files in system folders.

  If you install the driver files direct from a removable medium the system asks you on the next installation process to insert the medium again.

- Connect your USB device to the system. After the device has been plugged in, Windows launches the New Hardware Wizard and prompts you for a device driver. On Windows Vista and later the system shows you a short notification that the driver was not installed correctly. You have to open the device manager manually and install the NCM driver.

  Provide the New Hardware Wizard with the path of your installation files (e.g. cdcncm.inf and cdcncm.sys for Windows 2000/XP/Vista/7). Complete the wizard by following the instructions shown on screen. If the INF file matches your device, the driver should be installed successfully.

  Note that the New Hardware Wizard shows a warning message that complains about the fact that the driver is not certified and digitally signed. You may ignore this warning and continue with driver installation. The CDCNCM driver is not signed because it is not an end-user product. When the driver is integrated into such a product, it is possible to get a certification and a digital signature from the Windows Hardware Quality Labs (WHQL). On Windows Vista and later a digital signature of the vendor improves this behavior.

- If the operating system contains a driver that is suitable for your device, the system does not launch the New Hardware Wizard after the device is plugged in. Instead of that a system-provided device driver will be installed silently. The operating system does not ask for a driver because it finds a matching entry for the device in its internal INF file data base.

  You have to use the Device Manager to install the driver for a device for which a driver is already running. To start the Device Manager, right-click on the "My Computer" icon and

choose Properties. In the Device Manager, right-click on your device and choose Properties. On the property page that pops up choose Driver and click the button labeled "Update Driver". The Upgrade Device Driver Wizard is started which is similar to the New Hardware Wizard described above. Provide the wizard with the location of your installation files and complete driver installation by following the instructions shown on screen. To make sure the system installs the correct driver don't search for the best driver. Select it from a list and use the button "Have Disk" to browse to the file location.

- After the driver installation has been successfully completed your device should be shown in the Device Manager in the section Network Adapters. You may use the Properties dialog box of that entry to verify that the driver is installed and running.
- To verify that the driver is working properly with your device, you can check the adapter with the command line tool `ipconfig`. Depending on the device capabilities the adapter gets an IP address from DHCP or Auto IP. You can assign a static IP address in the network dialog. You can try to use the command line tool `ping` to get contact to your device. Consider fire wall settings on the PC.

5.1.2 Uninstalling CDCNCM manually

The manually un-installation of driver should be used by developers, only. If an important system driver is removed the system can become unusable.

To uninstall the device driver for a given device, use the Device Manager. The Device Manager can be accessed by right-clicking the "My Computer" icon on the desktop, choosing "Properties" from the context menu and then opening the Device Manager window. Within the Device Manager window, double-click on the entry for the device and choose the property page labeled "Driver". There are two options to uninstall the device driver:

- Remove the selected device from the system by clicking the button "Uninstall". The operating system will re-install a driver the next time the device is connected or the system is rebooted.
- Install a new driver for the selected device by clicking the button "Update Driver". The operating system launches the Upgrade Device Driver Wizard which searches for driver files or lets you select a driver.

In order to avoid automatic and silent re-installation of the driver by the operating system, it is necessary to remove the driver installation from the system.

On Windows Vista and higher the complete driver package with .sys, .inf and .cat files is stored in the driver store. To remove it from the driver store the dialog that is shown during the uninstallation allows to delete the drivers in the store. On Windows 2000 and Windows XP the .inf file must be removed manually.

During driver installation, Windows 2000 and Windows XP store a copy of the INF file in its internal INF file data base located in `%WINDIR%\INF\`. The name of the INF file is changed before it is stored in the database. On Windows 2000 and Windows XP the INF file is stored as `oemX.inf`, where X is a decimal number.

The best way to find the correct INF file is to do a search for some significant string in all the INF files in the directory `%WINDIR%\INF\` and its subdirectories. Note that on Windows 2000/XP, by default the `%WINDIR%\INF\` directory has the attribute Hidden. Therefore, by default the directory is not shown in Windows Explorer.

On Windows Vista and higher the .cat file is stored on an additional place. If you install the same driver a second time, the system behavior may be different, because the system knows the driver and does not warn the user for a not certified driver. Furthermore the digital signature can be stored in the certificate manager under Trusted Publishers. This happens if the user selects "always trust this vendor" during the installation process. If the certificate is stored there all digitally signed drivers can be installed in the same way as with a WHQL signature.

Once you have located the INF file, delete it. This will prevent Windows from reinstalling the driver. Instead, the New Hardware Wizard will be launched and you will be asked for a driver.

5.2 Installing CDCNCM driver for End Users

This section describes ways how the driver should be installed on a PC by the end user of the product.

5.2.1 Installing CDCNCM driver with the PnP Driver Installer

Thesycon provides a PnP Driver Installer Package that can be used to install kernel mode drivers in a convenient and reliable way. This installer is not part of this package. It can be downloaded separately under the following link: **http://www.thesycon.de/pnpinstaller**.

The installation program can be run in an interactive mode with graphical user interface or it can be run in a command line mode. The command line mode is designed to integrate the driver installer into other installation programs. The GUI mode guides the user through the installation. It supports different languages.

The driver installer package can be customized. A detailed description of the customization options is part of the reference documentation.

The PnP Driver Installer Package can handle the driver installation and un-installation in different situations:

- During the first time installation the driver is pre-installed in the system. In this step the user needs administrator privileges. When the driver is certified the pre-installation of the driver is performed silent. This means the hardware wizard is not launched and the system does not show a warning box for not certified software. When the device is connected to the PC during the installation the correct driver software is installed immediately. When a device is connected later to the PC the certified driver is installed silent. At the point of time where the device is connected to the PC no administrator privileges are required.
- The installer can perform a driver update. Old drivers and driver instances are removed from the system regardless whether the devices are connected or not. Exactly the driver version from the installer package is installed. This way of driver updates enables the upgrade to higher driver versions as well as the installation of older versions. The driver installation behaves in the same way as the first time installation.

- The installer can remove the driver software for a PnP device. This step can be performed by calling the installation program with a command line option or by using the appropriate option in the Windows control panel that is created during the driver installation. When the driver is removed all device nodes are uninstalled and the pre-installed drivers are removed. The installer makes sure that all drivers that are matching the USB VID and PID are removed from the system. The result is a system that behaves in the same way as when the driver software was never installed. The device nodes are left uninstalled. When a device is connected to the PC in this state the Found New Hardware wizard is launched.

The PnP Driver Installer Package supports all current Windows systems including 32 and 64 bit versions. The Thesycon team provides support and warranty for the product. A comprehensive documentation is part of the demo package available at **http://www.thesycon.de/pnpinstaller**.

5.2.2 Installing the Driver with DIFx

The DIFx software has some limitations that are explained in this section. For that reason the installation with this tools is only the second choice and not recommended by Thesycon.

DIFx is the Driver Install Frameworks provided by Microsoft. This software component can be redistributed. DIFx consists of the driver framework for applications, the Driver Install Frameworks API and the Driver Package Installer. This section introduces the Driver Package Installer, only. For details to the DIFx framework please refer to the Microsoft documentation.

The Driver Package Installer is an executable program. It is available for x86 and x64 in two different executables with the same name 'DPInst.exe'.

The Driver Package Installer can be run in silent mode or with user interface. It can be used to uninstall a driver package and it can be configured with command line parameters.

The installer has some drawbacks:

- Depending on the device connection state it updates the driver for connected devices and it pre-installs the driver for not connected devices. If the device is connected later the system performs the normal driver rating and may select a different driver. If the driver is updated with the /f (force) flag other drivers are overwritten for connected devices. The result of the installation depends from the fact whether the device was connected or not during the installation process. It is not sure that the driver from the installation package is installed for the device.

- After uninstalling a driver the installer activates an other driver that is pre-installed for the device or it leaves the node uninstalled. This can be an older version of the driver or a demo package. The state after uninstalling depends from other drivers that may be installed on the system.

6 Debug Support

6.1 Event Log Entries

If the driver detects a major problem during the startup process it creates an entry in the event log of the system. The event log can be opened with the context menu on 'My Computer' -> Manage -> Event Viewer -> System. The entries are created in the category error.

Common problems are:

6.1.1 Clear Feature Endpoint Halt

The device cannot handle this request. Solution: Set the flag `SendClearEpHaltOnStart` to 0.

6.1.2 Demo is Expired

The demo version has an internal clock. If the demo period has been expired the driver fails all operations. To re-enable the driver reboot the PC. This does not occur with the release version.

6.2 Enable Debug Traces

The licensed version of the driver package contains a folder drv_chk with the debug version of the driver. This folder is not part of the demo version. The debug version of the driver can generate text messages on a kernel debugger or a similar application to view the kernel output. These messages can help to analyze problems.

To enable the debug traces follow these steps.

- Install the driver with the normal setup.
- Copy the debug version of the driver to the folder `%SystemRoot%\system32\drivers`. If the driver has been renamed the debug version must be renamed in the same way.
- Reboot the PC to make sure the new driver is loaded. Connect your device.
- Open the registry editor. On Windows 2000/XP/Vista/W7 the following path must be opened:

  `\HKEY_LOCAL_MACHINE\System\CurrentControlSet\services\YOUR_SERVICE_NAME\`.

  YOUR_SERVICE_NAME is the name of the service name defined in the .INF file. It is typically the name of your driver.

  Edit the DWORD value key TraceMask. The messages are grouped in special topics. Each topic can be enabled with a bit in the debug mask. To enable the messages on bit 5 the TraceMask must be set to 0x00000020. The TraceMask contains the or'ed value of all active message bits.

- On Windows Vista and later an additional registry key must be created: `\HKLM\system\CurrentControlSet\Control\Session Manager\Debug Print Filter` In this key a DWORD value with name `DEFAULT` and the value `0xf` must be created. This enables the system debug print filter.
- Disconnect all devices or reboot the PC to make sure the driver is loaded again. The driver reads the registry key TraceMask if it is started.
- Start a kernel debugger like WinDbg or an application like DebugView to receive the kernel traces. DebugView is a free software that can be downloaded on the Microsoft web page. Connect your device.
- If your problem ends up in a blue screen of death or you see an unexpected re-boot of the PC please change the following settings: System Properties -> Advanced -> Startup and Recovery -> Settings -> Write Debugging Information to "Kernel Memory Dump". Set the registry key `TraceLogSizeKB` in the same location as the `TraceMask` to 128. This includes the last 128kb traces to the memory dump. Reproduce the BSOD again. Transfer the memory dump (typically `%SystemRoot%\MEMORY.DMP`) to Thesycon for analysis.

The following table summarizes the meaning of the debug bits.

Table 1: Trace Mask Bit Content

Bit	**Content**
0	Fatal errors
1	Warnings
2	Information
3	Extended information
4	OID based requests
8	Tx data path handling
12	Data dump
16	Rx data path handling

If you submit a problem description to Thesycon please include the following:

- operating system and Service Pack
- USB host controller (EHC, UHC, OHC) and usage of external USB hubs
- Driver version and driver build (release/debug/demo) that has been used
- used TraceMask if kernel Traces have been recorded

- The actions you have performed to cause the problem

Do not send memory dumps with e-mail. We can provide a FTP account.

7 Related Documents

References

[1] Universal Serial Bus Specification 1.1,
http://www.usb.org

[2] Universal Serial Bus Specification 2.0,
http://www.usb.org

[3] USB device class specifications (Audio, CDC, NCM etc.),
http://www.usb.org

[4] Microsoft Developer Network (MSDN) Library,
http://msdn.microsoft.com/library/

[5] Windows Driver Development Kit,
http://msdn.microsoft.com/library/

[6] Windows Platform SDK,
http://msdn.microsoft.com/library/