



Driver Manual
cifX Device Driver
Windows CE
V1.0.x.x

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC100703DRV01EN | Revision 1 | English | 2010-08 | Released | Public

Table of Contents

1	Introduction.....	3
1.1	About this Document.....	3
1.2	List of Revisions.....	3
1.3	Terms, Abbreviations and Definitions.....	4
1.4	References.....	4
1.5	Legal Notes.....	5
	1.5.1 Copyright.....	5
	1.5.2 Important Notes.....	5
	1.5.3 Exclusion of Liability.....	6
	1.5.4 Export.....	6
2	Windows CE Driver.....	7
2.1	Overview.....	7
2.2	Requirements.....	8
2.3	Features.....	8
2.4	Limitations.....	8
2.5	CD Contents.....	9
2.6	General Information about Windows CE Drivers.....	10
2.7	Compiling the Source Code.....	11
	2.7.1 Windows CE 5.0.....	12
	2.7.2 Windows CE 6.0.....	16
2.8	Driver Installation.....	21
	2.8.1 Using the Platform Builder.....	22
	2.8.2 On an existing Target System.....	24
	2.8.3 Driver Registry Settings.....	26
	2.8.4 User Specific Initialization.....	29
2.9	Driver Setup and Test Program.....	30
3	Error Numbers.....	31
4	Appendix.....	34
4.1	List of Tables.....	34
4.2	List of Figures.....	34
4.3	Contacts.....	35

1 Introduction

1.1 About this Document

This manual describes the cifX device driver for the Microsoft Windows CE operating systems.

Both versions of the cifX driver are offering the same functionality and also the same application programming interface (API) to access a netX based hardware (e.g. cifX, comX boards and the netX chip).

In general, the drivers are supporting various netX based hardware designs described under *Requirements* for each of the driver.

The API (CIFX API) is designed to give the user an easy access to all of the communication board functionalities. This manual also includes a detailed description of the CIFX API functions.

In addition, Hilscher also offers a free of charge *cifX Toolkit* (C-source code based) which allows to write own drivers based on the Hilscher netX DPM (dual-port memory) definitions including the CIFX API functions. The toolkit is described in a separate manual *cifX/netX Toolkit*.

1.2 List of Revisions

Rev	Date	Name	Chapter	Revision
1	2010-08-18	MT, SS	all	Initial version separated from cifX device driver manual for windows

Table 1: List of Revisions

1.3 Terms, Abbreviations and Definitions

Term	Description
cifX	Communication Interface based on netX
comX	C ommunication M odule based on netX
PCI	P eripheral C omponent I nterconnect
WDM	W indows D river M odel
DLL	D ynamic L ink L ibrary
API	A pplication P rogramming I nterface
SDO	S ervice D ata O bject
PDO	P rocess D ata O bject
DPM	D ual- P ort M emory Physical interface to all communication board (DPM is also used for PROFIBUS- DP Master).

Table 2: Terms, Abbreviations and Definitions

1.4 References

This document based on the following specification:

- [1] Real-time Communication-System for netX
- [2] netX Bootstrap Specification
- [3] netX Program Reference Guide (PCI)
- [4] netX DPM Interface Manual

Table 3: References

1.5 Legal Notes

1.5.1 Copyright

© 2006-2010 Hilscher Gesellschaft für Systemautomation mbH

All rights reserved.

The images, photographs and texts in the accompanying material (manual, accompanying texts, documentation, etc.) are protected by German and international copyright law as well as international trade and protection provisions. You are not authorized to duplicate these in whole or in part using technical or mechanical methods (printing, photocopying or other methods), to manipulate or transfer using electronic systems without prior written consent. You are not permitted to make changes to copyright notices, markings, trademarks or ownership declarations. The included diagrams do not take the patent situation into account. The company names and product descriptions included in this document may be trademarks or brands of the respective owners and may be trademarked or patented. Any form of further use requires the explicit consent of the respective rights owner.

1.5.2 Important Notes

The manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

1.5.3 Exclusion of Liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

1.5.4 Export

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different countries, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product.

2 Windows CE Driver

ATTENTION:

The Windows CE driver comes as source code with a demo project. Because of the hardware options of Windows CE, the driver will not be delivered as a pre-compiled runnable driver module. The driver source must be compiled for the specific platform it is used on.

Note: Sources and project files for Windows CE 5 and Windows CE 6. are located in separate directories on the CD.

2.1 Overview

- The *cifX Device Driver* for Windows CE is a stream driver, running in the kernel of the operating system.
- Access to the driver functions is offered by a driver API DLL (*cifXCEDLL*). The DLL covers the device IO control calls used to communicate with the driver and offers the same CIFX API like on the Windows desktop operating systems.

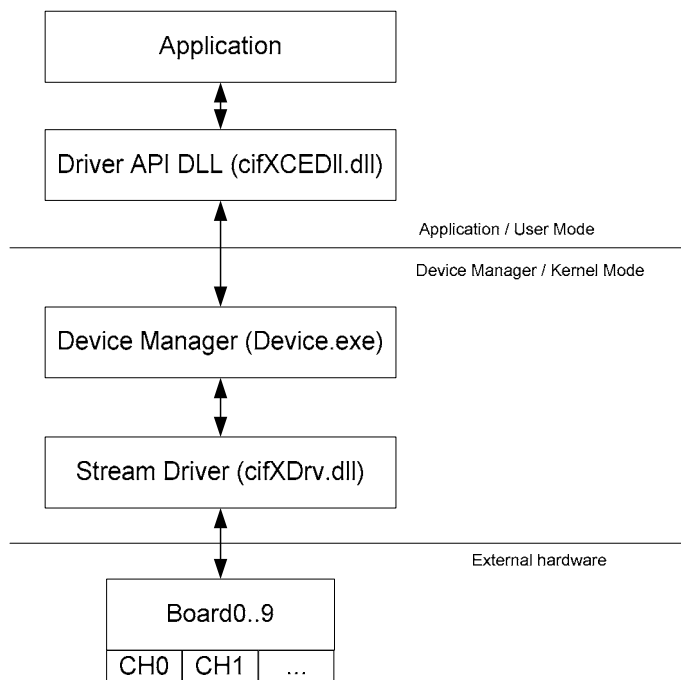


Figure 1 : CifX CE Driver - Architecture

2.2 Requirements

- Microsoft Windows CE 5.0 or CE 6.0
- Development environment:
Windows CE 5.0 - Microsoft Platform Builder / Microsoft Embedded Visual C++ 4.0
Windows CE 6.0 - Microsoft Visual Studio 2005 / Platformbuilder Addon
- Host CPU using the Little-Endian (Intel) data format

2.3 Features

- Supports up to 10 devices (limited by Windows CE)
- Full source code included
- Stream driver that can be included into a Windows CE image or loaded from an external disk
- ISA / DPM Card support
- Support for NXSB-PCA and NX-PCA-PCI Boards
- PCI support using PCI Enumerator templates
- Support for DMA data transfer for I/O data (cifX50/netPLC only)
- Support for Interrupt and interrupt notification

2.4 Limitations

- Tested and developed for Windows CE 5.0 / 6.0
- For RAM based cards like the cifX50, the driver needs access to firmware and configuration files during the start-up phase.

2.5 CD Contents

Folder	Content
Documentation	Documentation of the cifX Windows CE Driver
Driver	Application programming interface files
CE5	CE Driver for Windows CE 5.0
PlatformBuilder	Platform Builder Project
EVC4	Embedded Visual C++ 4.0 Project
Application	Example Applications
CE6	CE Driver for Windows CE 6.0
PlatformBuilder	Platform Builder Project
VS2005	Visual Studio (Smart Device) Project
Application	Example Applications
Sample Registry	Sample Registry files for driver installation

Table 4: CD Contents

2.6 General Information about Windows CE Drivers

Windows CE drivers are DLLs which must meet some special requirements. A driver must offer a specific interface to be accessible by the operating system. Windows CE uses a three letter prefix to distinguish between different drivers. The information about the driver prefix and the name of the driver file must be defined in the registry.

This information will be used by the operating system to call a driver.

Windows CE Driver Interface:

Following functions must be available in the driver interface and will be called by the operating system:

```
xxx_Init
xxx_Deinit
xxx_Open
xxx_Close
xxx_Read
xxx_Write
xxx_Seek
xxx_PowerDown
xxx_PowerUp
xxx_IOControl
```

Where xxx is the driver prefix. The prefix for Hilscher drivers is '**CFX**'.

Function Description:

- CFX_Init** This function is called by the *Device Manager* to initialize a driver.
- CFX_Deinit** When the user stops using a device, the *Device Manager* calls this function. The function frees all virtual memory that was allocated during the install step.
- CFX_Open** This function opens a device for reading and/or writing. An application indirectly invokes this function when it calls **CreateFile()** to open a cifX device.
The function creates a new device context information structure.
- CFX_Close** Is called when an application calls **CloseHandle()** to stop using a stream interface driver. It also removes an open context from the linked list and frees this context.
- CFX_Read, CIF_Write, CIF_Seek (not supported)** Standard interface for **ReadFile()** and **WriteFile()** functions. This is not supported by the cifX driver.
- CFX_PowerDown, CIF_PowerUp** Are used to manage power management events from the operating system. These functions have an empty body and are not used by the cifX driver.
- CFX_IOControl** This function sends a command to a device when an application uses the **DeviceIOControl()** function to specify an operation to be performed. All CIF functions are processed by **DeviceIOControl()** function.

2.7 Compiling the Source Code

Note: All drivers and applications must be compiled before they can be used.

Microsoft has divided the development process for Windows CE systems into two parts. One part is the generation of a running hardware platform using the Windows CE operating system while the other part focuses the application development for an existing Windows CE system.

Both parts are using different environments. Platforms are generated with the Microsoft Platform Builder and the application development is based on the Microsoft Embedded Visual tools (EVC) or Microsoft Visual Studio 2005 (depends on the Windows CE version).

There is one important dependency between the two parts. Application development requires information about the target system like the used CPU, available system drivers, available system services and platform depending libraries and definition files. These informations are only available in the platform generation process.

The only way to get the information into the application development process is a platform SDK (**S**oftware **D**evelopment **K**it) which must be generated by the Microsoft Platform Builder.

Because of the above described development process, it is not possible to offer pre-compiled drivers and application for a specific Windows CE target system.

2.7.1 Windows CE 5.0

2.7.1.1 Building CifXCEDriver, using the Platform Builder

The cifX CE driver includes a make file which can be used to compile the driver during the target build process. The integration of the driver into the build process is described under "*Driver Installation->Using the Platform Builder*".

Platform Builder Project - Directory Structure:

Directory	Description	
PlatformBuilder	Platform Builder project directory	
	Directory	Description
	.API	Source of the driver API DLL
	.Driver	Sources of the cifX driver
	.Toolkit	Additional driver sources used by the driver build process
	.BSL	Bootloader

Table 5: Platform Builder - Project Directory

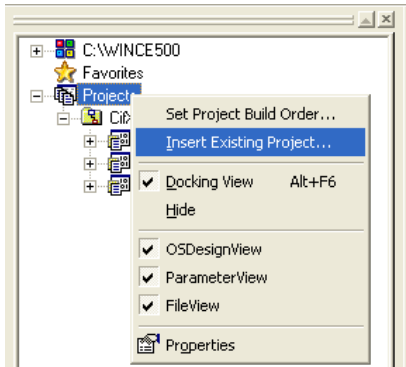
Build Procedure:

Note: Directories and files copied from the CD are Read-Only. Change the file attribute of all files to Read/Write before starting the build process!

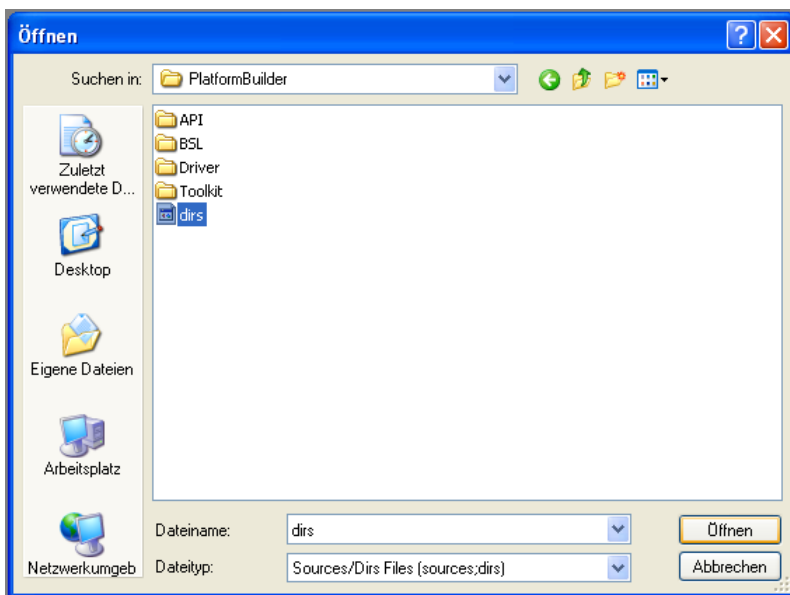
The following steps need to be done to build and integrate the driver into a runtime image:

- Copy the whole '*Platformbuilder*' directory to your development system. (usually `<Wince>\platform\<bsp>`).

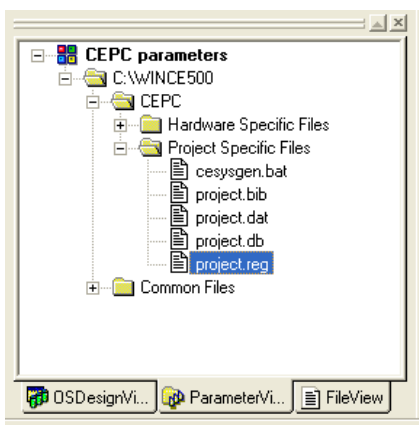
- Add the project to your Platform Builder solution



- Select file type 'Sources/Dir's'



- Add all necessary registry and .bib entries to your project/platform. An example registry file 'cifXCEDrv.reg' is placed under 'Platformbuilder\Driver\'. 'CifXDrv.bib' ('PlatformBuilder\CifXDrv.bib') contains information how to offer the required 'Driver-DLL' and the 'Bootloader' files.



- Build your platform

2.7.1.2 Building CifXCEDriver, using Embedded Visual C++ (EVC)

Compilation of the driver source, by using EVC, assumes a platform SDK for the target platform is available and installed. The SDK is necessary, because the driver project file references the '**ceddk.lib**' provided by the SDK.

Note: A platform SDK is generated by the Microsoft Platform Builder and installable like a normal program. It can only be offered by the company creating the target platform. Make sure the Microsoft Embedded Visual tools are already installed before installing a platform SDK.

EVC4 Project - Directory Structure:

Directory	Description	
EVC4	Embedded Visual C++ 4.0	
	Directory	Description
	.cifXCEDriver	cifX CE Driver, DLL and test programs
		Directory
		Description
	.BSL	Bootloader files
	.cifXCEDLL	Sources of the cifX driver API DLL, including the EVC project file
	.cifXCEDrv	Sources of the cifX driver, including the EVC project file
	.Toolkit	Additional driver sources used by the driver build process
	.ddk	Additional libraries and include files (Driver Development Kit)

Table 6: EVC4 - Project Directories

Build Procedure:

Note: Directories and files copied from the CD are Read-Only. Change the file attribute of all files to Read/Write before starting the build process!

- Copy the directory path, starting with '*EVC4*' and all sub-directories to your development system.
- Compile the different modules, driver and API-DLL (use the included project files (*.vcproj or cifXCE.vcproj for the whole workspace to open the projects with EVC). The API-DLL should be build after the build of the cifXCEDrv, because the build process depends on this project.

2.7.1.3 Building Example Application, using Embedded Visual C++ (EVC)

The application directory contains different example applications to test the driver functionality. Use the included workspace file (CifXApplication.vcw) to open all projects in one workspace.

Directory	Description										
Application	Example Applications for the cifX CE Driver (Embedded Visual C++ 4.0)										
	<table border="1"> <thead> <tr> <th>Directory</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.cifXTest</td> <td>Dialog based driver test program (320x240), including the EVC project file</td> </tr> <tr> <td>.cifXTest_Console</td> <td>Console based driver test program, including the EVC project file</td> </tr> <tr> <td>.cifXTCPServer</td> <td>TCP/IP Server for cifX devices, including the EVC project file</td> </tr> <tr> <td>.cifXSetup</td> <td>Dialog based tool for changing registry entries and settings</td> </tr> </tbody> </table>	Directory	Description	.cifXTest	Dialog based driver test program (320x240), including the EVC project file	.cifXTest_Console	Console based driver test program, including the EVC project file	.cifXTCPServer	TCP/IP Server for cifX devices, including the EVC project file	.cifXSetup	Dialog based tool for changing registry entries and settings
Directory	Description										
.cifXTest	Dialog based driver test program (320x240), including the EVC project file										
.cifXTest_Console	Console based driver test program, including the EVC project file										
.cifXTCPServer	TCP/IP Server for cifX devices, including the EVC project file										
.cifXSetup	Dialog based tool for changing registry entries and settings										

Table 7: Application - Project Directories

Build Procedure:

Note: Directories and files copied from the CD are Read-Only. Change the file attribute of all files to Read/Write before starting the build process!

- Copy the directory path, starting with '*Application*' and all sub-directories to your development system.
- Copy also the directory path '*EVC4*', because the applications link against the API-DLL. If the folder structure changes, renew the specified include paths.
- Compile the different applications (use the included project files (*.vcp) to open the projects with EVC). The test program should be build after the build of the API-DLL, because the build process depends on the API-DLL '*.lib*' file.
- **Note:** To run one of the application on the target, you must define (or integrate the corresponding modules in the OSDesign) at least the following Sysgen Variables:
 - **SYSGEN_CPP_EH_AND_RTTI** (C++ Runtime Support - Exception Handling)
 - **SYSGEN_CONSOLE** (Console Window)
 - **SYSGEN_PRINTING** (Printer Devices)
 - **SYSGEN_WININET** (Windows Internet Services)

2.7.2 Windows CE 6.0

2.7.2.1 Building CifXCEDriver, using the Platform Builder

The Platform Builder example will include all drivers into your Windows CE runtime image. To change the registry settings or the included files, you will need to edit "*cifXCEDrv.reg*" and "*cifXCEDrv.bib*" inside the project directory.

The Platform Builder project does not include the Setup/Testapplications. These must be compiled using the Visual Studio 2005 project.

Platform Builder Project - Directory Structure:

Directory	Description	
PlatformBuilder	Platform Builder Project	
	Directory	Description
	.\cifXCEDriver	cifX CE Driver, DLL and test programs
	Directory	Description
	.\API	Device driver interface DLL (cifXCEDll.dll)
	.\BSL	Bootloader files
	.\DRIVER	cifX Device driver (cifXCEDrv.dll)
	.\TOOLKIT	cifX Toolkit, additional driver sources used by the driver build process

Table 8: Platform Builder - Project Directories

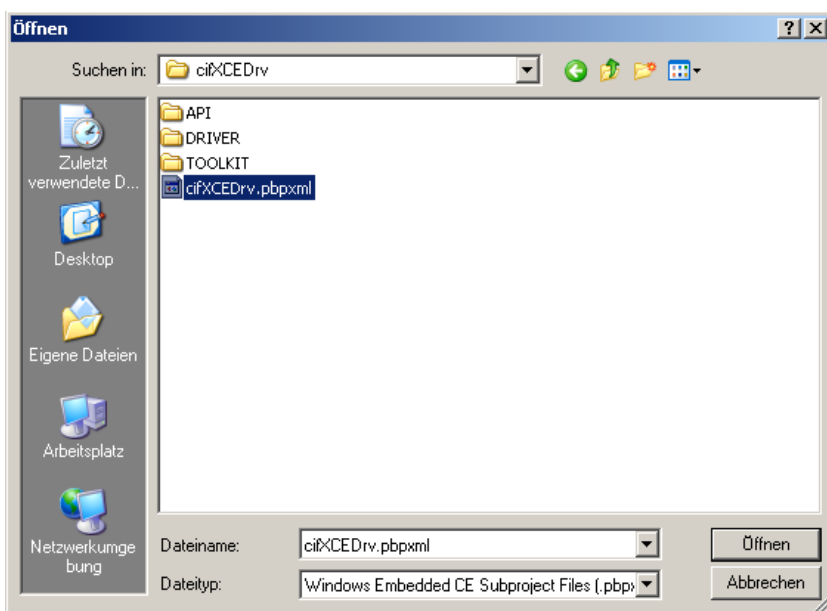
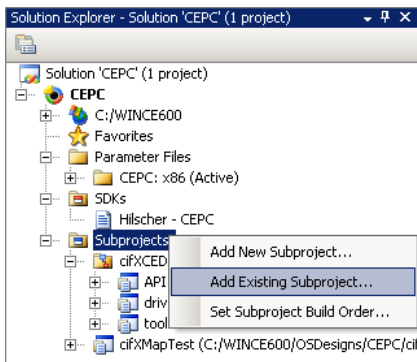
Build Procedure:

Note: Directories and files copied from the CD are Read-Only. Change the file attribute of all files to Read/Write before starting the build process!

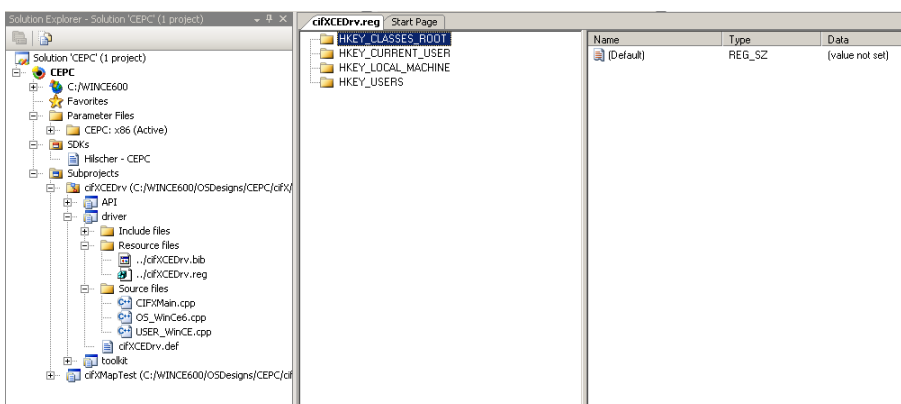
- Copy the directory path, starting with '*CifXCEDriver*' and all sub-directories to your development system.

The following steps need to be done to build and integrate the driver into a runtime image:

- Add the project to your Platform Builder solution



- Edit the registry settings either in "project.reg" or in "cifXCEDrv.reg"



- Build your platform

2.7.2.2 Building CifXCEDriver, using Microsoft Visual Studio 2005

Compilation of the driver source, by using Visual Studio, assumes a platform SDK for the target platform is available and installed. The SDK is necessary, because the driver project file references the '**ceddk.lib**' provided by the SDK.

Note: A platform SDK is generated by the Microsoft Platform Builder and installable like a normal program. It can only be offered by the company creating the target platform.

Visual Studio 2005 Project - Directory Structure:

Directory	Description	
VS2005	Visual Studio 2005 Solution	
	Directory	Description
	.cifXCEDriver	cifX CE Driver, DLL and test programs
	Directory	Description
	.BSL	Bootloader files
	.cifXCEDll	Device driver interface DLL (cifXCEDll.dll)
	.cifXCEDrv	cifX Device driver (cifXCEDrv.dll)
	.Toolkit	cifX Toolkit
	.DDK	required header files (Driver Development Kit)

Table 9: Visual Studio 2005 - Project Directories

Before a target platform SDK can be used, it must be installed and the platform configuration needs to be adjusted.

Build Procedure:

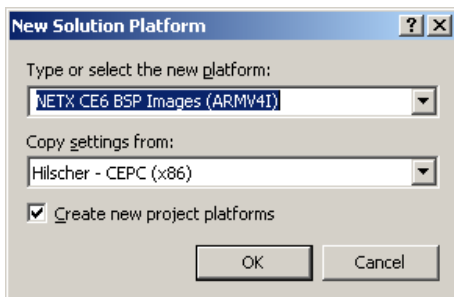
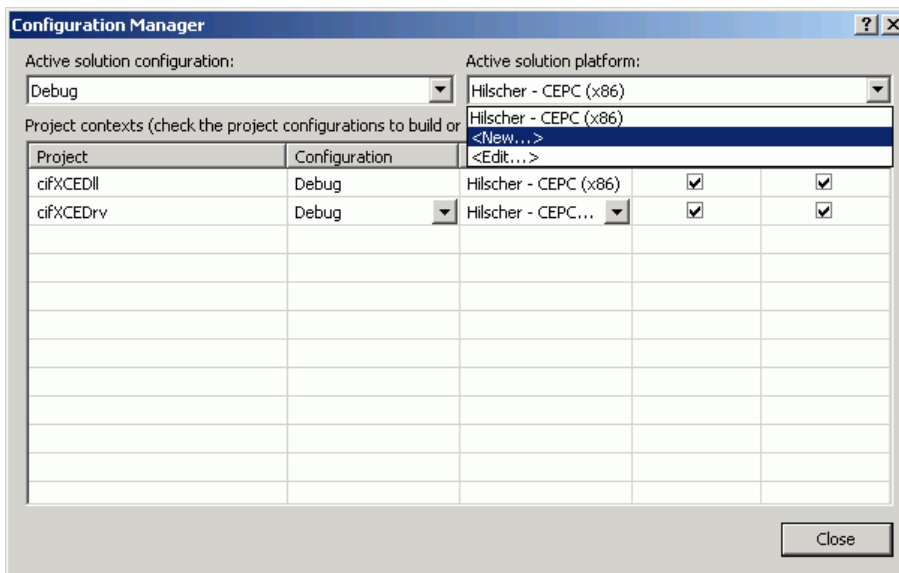
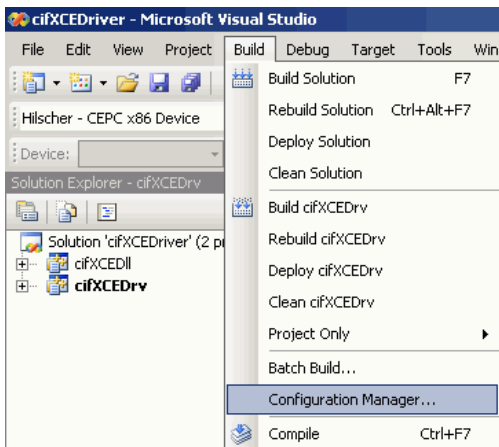
Note: Directories and files copied from the CD are Read-Only. Change the file attribute of all files to Read/Write before starting the build process!

- Copy the directory path, starting with '*CifXCEDriver*' and all sub-directories to your development system.
- First of all compile the cifX driver and the API-DLL (use the included solution file (*.sln) to open the projects with VS2005).

Note: If you load the included solution file the following dialog message may pop up: *"The project consists entirely of configurations that require support for platforms which are not installed on this machine. The project cannot be loaded."* In this case please install the *Standard CEPC Image for Testing* (Driver\CE6.0\VS2005\HIL_CEPC.msi), included on the driver CD.

The following steps need to be done to compile the solution:

- Add your target SDK to known configurations



- Rebuild the project

2.7.2.3 Building Example Application, using Microsoft Visual Studio 2005

The application directory contains different example applications to test the driver functionality. Use the included solution file (cifXApplications.sln) to open all projects in one workspace.

Directory	Description	
Application	Example Applications for the cifX CE Driver (Visual Studio 2005 Solution)	
	Directory	Description
	.\aygshell	Necessary for MFC applications if not included in SDK
	.\cifXCESetup	cifX Setup Utility
	.\cifXTCPServer	TCP/IP Server for cifX devices
	.\cifXTest	Dialog based driver test program (cifXTest.EXE) compiled for screen with at least 640x480
	.\cifXTest_console	Driver test application for the windows console (cifxtest_console.exe)

Table 10: Application - Project Directories

Build Procedure:

Note: Directories and files copied from the CD are Read-Only. Change the file attribute of all files to Read/Write before starting the build process!

- Copy the directory path, starting with 'Application' and all sub-directories to the same destination as the driver sources, because the applications link against the API-DLL.
- Compile the test applications (use the included solution file (*.sln) to open the projects with VS2005). The test program should be build after the build of the API-DLL, because the build process depends on the API-DLL '.lib' file.

2.8 Driver Installation

The cifX CE driver for Windows CE consist of two files, a driver file and an API DLL. The Installation of the driver can be done in two ways.

First one is the static integration of the driver in a Windows CE image. The second way is the dynamic loading of the driver, if an already existing Windows CE image is used.

In both cases, some driver specific registry settings are necessary to make the driver loadable by the system.

Installation can be done by using the following two ways:

- Static integrated into a Windows CE Image using the Platform Builder
- Dynamically loading the driver from an existing Windows CE

File	Description	Location
cifxcdrv.dll	Stream driver DLL	Default: '\Windows' is the standard Windows CE search path for DLLs and drivers. User Defined File Location: If the driver should be loaded from another path, the path must be included the Windows CE system search path. [HKEY_LOCAL_MACHINE\Loader] "SystemPath"=multi_sz:"\userpath\"
cifxcdll.dll	API DLL	

Table 11 : CE Driver - Files to Install

The subsequent chapters are describing the steps to integrate the cifX driver into Windows CE.

2.8.1 Using the Platform Builder

This chapter describes the integration of the cifX driver, the driver API DLL and the necessary registry settings into the build process of a Windows CE target image.

The delivered projects are compatible to the "sources" build sequence used by the Platform Builder and can directly be inserted into a Workspace (tested under Windows CE 5).

All necessary registry settings can either be done in the **project.reg** file of the workspace or by modifying any of the additionally included **.reg** files. The sample below uses the **project.reg** / **project.bib** files which are always available in a Platform Builder project and uses a PROFIBUS firmware/configuration located under "C:\FW".

Note: If the downloaded firmware resides on a removable/external disk, make sure to set the "Order" value to something higher, than the disk driver. Otherwise the disk is not available when the cifX driver starts, resulting in an error opening the firmware file

Sample ISA/DPM Configuration:

project.bib

```

MODULES
  cifXCEDll.dll  $_FLATRELEASEDIR)\cifXCEDll.dll  NK
  cifXCEDrv.dll  $_FLATRELEASEDIR)\cifXCEDrv.dll  NK SH
FILES
  config.nxd     C:\FW\config.nxd                    NK
  dpm.nxf        C:\FW\cifxdpm.nxf                    NK
Note: The firmware and configuration file name must be in 8.3 format

```

project.reg

```

[HKEY_LOCAL_MACHINE\Software\Hilscher GmbH\cifX Device Driver]
  "TraceLevel"=dword:0000000F

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\CIFXCEDRV0]
  "Prefix"="CFX"
  "Dll"="cifxcdrv.dll"
  "Order"=dword:0
  "Index"=dword:0
  "PhysicalAddress"=dword:26000000
  "DPMSize"=dword:00010000
  "Irq"=dword:7
  "SysIntr"=dword:0
  ; NOTE: If SysIntr is 0 the value will be automatically
  ;         requested by the driver (IOCTL_HAL_REQUEST_SYSINTR)
  "InterruptEnable"=dword:0
  "IsrDll"="giisr.dll"
  "IsrHandler"="ISRHandler"
  "Alias"="PROFIBUS"

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\CIFXCEDRV0\CHANNEL0]
  "ModuleCount"=dword:1
  "Module0"="\\Windows\\cifxdpm.nxf"
  "ConfigCount"=dword:1
  "Config0"="\\Windows\\config.nxd"

```

Sample PCI Configuration:**project.bib**

```

MODULES
  cifXCED11.dll  $(_FLATRELEASEDIR)\cifXCED11.dll  NK
  cifXCEDrv.dll  $(_FLATRELEASEDIR)\cifXCEDrv.dll  NK SH
FILES
  config.nxd      C:\FW\config.nxd          NK
  dpmcifx.nxf     C:\FW\dpmcifx.nxf        NK
  NETX50-BSL.bin  C:\FW\NETX50-BSL.bin     NK
  NETX100-BSL.bin C:\FW\NETX100-BSL.bin    NK
Note: The firmware and configuration file name must be in 8.3 format

```

project.reg

```

[HKEY_LOCAL_MACHINE\Software\Hilscher GmbH\cifX Device Driver]
  "TraceLevel"=dword:0000000F

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\PCI\Template\cifx]
  "Prefix"="CFX"
  "Dll"="cifxcdrv.dll"
  "Order"=dword:08
  "Class"=dword:FF
  "SubClass"=dword:00
  "ProgIF"=dword:00
  "VendorID"=multi_sz:"15CF", "15CF"
  "DeviceID"=multi_sz:"0000", "0010"
  ; NOTE: To start up NXSB-PCA/NX-PCA-PCI boards a
  ;       different configuration must be used!
  ;       "Class"=dword:00000006
  ;       "SubClass"=dword:00000080
  ;       "VendorID"=multi_sz:"10B5"
  ;       "DeviceID"=multi_sz:"9030"
  "DMAEnable"=dword:1
  "InterruptEnable"=dword:1
  "IsrDll"="giisr.dll"
  "IsrHandler"="ISRHandler"

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\PCI\Template\cifx\Channel0]
  "ModuleCount"=dword:1
  "Module0"="\\Windows\\cifxdpm.nxf"
  "ConfigCount"=dword:1
  "Config0"="\\Windows\\config.nxd"

```

2.8.2 On an existing Target System

This chapter describes the dynamic integration of the cifX CE driver on an existing Windows CE target.

First step is the compilation of the driver source code for the given target (described under 'Compiling the Source Code').

The compilation can be done by either by the Microsoft Platform Builder for Windows CE, the Microsoft Embedded Visual C++ 4.0 compiler (EVC++ 4.0) or Visual Studio 2005.

Both possibilities are supported by the necessary make and project files. After compiling the sources for the given target, the following steps must be processed to include the driver

- Copy the driver and API DLL files to a persistent disk drive
- Creating the drivers registry entries
- Setting up the registry entries, so the system is able to start the driver
- Store the registry persistent

Note: If the downloaded firmware resides on a removable/external disk, make sure to set the "Order" value to something higher, than the disk driver. Otherwise the disk is not available when the cifX driver starts, resulting in an error opening the firmware file

Sample ISA/DPM Registry:

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\CIFXCEDRV0]
"Prefix"="CFX"
"Dll"="cifxcedrv.dll"
"Order"=dword:0
"Index"=dword:0
"PhysicalAddress"=dword:26000000
"DPMSize"=dword:00010000
"Irq"=dword:7
"SysIntr"=dword:0
; NOTE: If SysIntr is 0 the value will be automatically
; requested by the driver (IOCTL_HAL_REQUEST_SYSINTR)
"InterruptEnable"=dword:0
"IsrDll"="giisr.dll"
"IsrHandler"="ISRHandler"
"Alias"="PROFIBUS"
"BootloaderFile"="\\FlashDisk\\NETX100-BSL.bin"

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\CIFXCEDRV0\CHANNEL0]
"ModuleCount"=dword:1
"Module0"="\\FlashDisk\\cifxdpm.nxf"
"ConfigCount"=dword:1
"Config0"="\\FlashDisk\\config.nxd"

[HKEY_LOCAL_MACHINE\Software\Hilscher GmbH\cifX Device Driver]
"TraceLevel"=dword:0000000F

[HKEY_LOCAL_MACHINE\Loader]
"SystemPath"=multi_sz:"\\FlashDisk\\"
```

Sample PCI Registry:

```
[HKEY_LOCAL_MACHINE\Software\Hilscher GmbH\cifX Device Driver]
  "TraceLevel"=dword:0000000F

[HKEY_LOCAL_MACHINE\Software\Hilscher GmbH\cifX Device Driver\NX-PCA-PCI]
  "DPM_8_Bit"=dword:5431F962
  "DPM_16_Bit"=dword:4073F8E2
  "DPM_32_Bit"=dword:40824122

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\PCI\Template\cifx]
  "Prefix"="CFX"
  "Dll"="cifxcedrv.dll"
  "Order"=dword:08
  "Class"=dword:FF
  "SubClass"=dword:00
  "ProgIF"=dword:00
  "VendorID"=multi_sz:"15CF", "15CF"
  "DeviceID"=multi_sz:"0000", "0010"
  ; NOTE: To start up NXSB-PCA/NX-PCA-PCI boards a
  ;       different configuration must be used!
  "Class"=dword:06
  "SubClass"=dword:80
  "VendorID"=multi_sz:"10B5"
  "DeviceID"=multi_sz:"9030"
  "DMAEnable"=dword:1
  "InterruptEnable"=dword:1
  "IsrDll"="giisr.dll"
  "IsrHandler"="ISRHandler"
  "BootloaderFile"="\\FlashDisk\NETX100-BSL.bin"

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\PCI\Template\cifx\Channel0]
  "ModuleCount"=dword:1
  "Module0"="\\FlashDisk\cifxdpm.nxf"
  "ConfigCount"=dword:1
  "Config0"="\\FlashDisk\config.nxd"

[HKEY_LOCAL_MACHINE\Loader]
  "SystemPath"=multi_sz:"\\FlashDisk\\"
```

Sample Registry (ISA/DPM being forced to PCI handling mode):

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\CIFXCEDRV0]
  "Prefix"="CFX"
  "Dll"="cifxcedrv.dll"
  "Order"=dword:0
  "Index"=dword:0
  "PhysicalAddress"=dword:26000000
  "DPMSize"=dword:00010000
  "InitDll"="SetupTimings.dll"
  "Alias"="PROFIBUS"
  "ForcePCIDPM"=dword:00000001

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\CIFXCEDRV0\CHANNEL0]
  "ModuleCount"=dword:1
  "Module0"="\\Windows\dpmcifx.nxf"
  "ConfigCount"=dword:1
  "Config0"="\\Windows\config.nxd"

[HKEY_LOCAL_MACHINE\Software\Hilscher GmbH\cifX Device Driver]
  "TraceLevel"=dword:0000000F
```

2.8.3 Driver Registry Settings

The registry is used to store the device specific settings. When Plug&Play cards are used, the driver offers the possibility to store multiple configurations and assigning them by evaluating the Device- and Serial number of the card.

On a custom hardware the user might want to statically assign a configuration to the device. This is also offered by the driver.

The following keys are available:

Basic Driver and Device Configuration		
Value	Type	Description
ISA Mode: [HKLM\Drivers\BuiltIn\ <devicename>]< td=""> </devicename>]<>		
PCI Mode: [HKLM\Drivers\BuiltIn\PCI\Template\ <templatename>]< td=""> </templatename>]<>		
Prefix	REG_SZ	Driver Prefix. Must be " CFX "
Dll	REG_SZ	Name of the driver dll. defaults to " cifxcdrv.dll "
Order	REG_DWORD	Load order (CE specific, see Platform Builder documentation)
Index	REG_DWORD	Optional: Device Index (CE specific, see Platform Builder documentation)
ISA Device Specific Setting		
PhysicalAddress	REG_DWORD	Physical DPM address
DPMSize	REG_DWORD	Size of the DPM in bytes
Irq	REG_DWORD	Irq number
SysIntr	REG_DWORD	Sysintr value mapped to the irq. Note: If SysIntr is 0 the value will be automatically requested by the driver (IOCTL_HAL_REQUEST_SYSINTR)
ForcePCIDPM	REG_DWORD	Force a DPM board to be handled like a PCI device (Device will be reset and only modules can be downloaded)
Device handling		
InitDll	REG_SZ	Initialization DLL called offering special initialization functions (can be used if the hardware needs specific settings before the device can be accessed)
BootloaderFile	REG_SZ	Optional: Definition of a 2 nd stage bootloader file. Default: "\\Windows\NETX100-BSL.BIN"
IsrDll	REG_SZ	Optional: Tell the OS which installable ISR DLL to use. Default: " giisr.dll " (generic ISR DLL)
IsrHandler	REG_SZ	Optional: Tell the OS the installable ISR DLL entry point. Default: " ISRHandler " (generic ISR handler function)

Table 12 : CE-Driver - Basic Driver and Device Configuration

Global Driver Settings		
Value	Type	Description
[HKLM\Software\Hilscher GmbH\cifX Device Driver]		
TraceLevel	REG_DWORD	Adjust the amount of debug output of the driver. Each bit represents a level Bit 0 : Enable debug Bit 1 : Enable informational data Bit 2 :Enable warnings Bit 3 :Enable errors
PollingInterval	REG_DWORD	Poll interval in [ms] for devices operating in polling mode (Used for COS flag handling)
[...\NX-PCA-PCI] Timing settings for PLX coupled devices (NXPCA-PCI)		
DPM_8_Bit	REG_DWORD	Timing for 8 bit mode. Default: 0x5431F962
DPM_16_Bit	REG_DWORD	Timing for 16 bit mode. Default: 0x4073F8E2
DPM_32_Bit	REG_DWORD	Timing for 32 bit mode. Default: 0x40824122

Table 13 : CE-Driver - Global Driver Settings

To allow device specific configuration, every file that needs to be downloaded must be declared in the registry of Windows CE. The driver provides three ways to assign the device specific configuration:

- Use a static assignment via the basic driver and device configuration

A fix definition of firmware and configuration can be declared in the basic driver and device configuration (see table 10). Please note that in case of PCI templates, each found card will get the same firmware/configuration definition.

- Use the *Slotnumber* (hardware rotary switch)

If the driver can't find a static assignment, the *Slotnumber* is used to find firmware/configuration definitions. The *Slotnumber* serves to distinguish cifX cards from each other clearly, especially if more cifX cards are installed in one PC. The *Slotnumber* must be set at the cifX card using the Rotary Switch *Slotnumber*. While *Slotnumber* 0 means, that the cifX card is identified via its device and serial number, values from 1 to 9 corresponds to the *Slotnumber* 1 to 9.

- Use the device and serial number

If the driver can't find a static definition and the cifX device is not equipped with a rotary switch or the *Slotnumber* should not be used, the device is identified by its device and serial number.

Device Specific Settings		
Value	Type	Description
[HKLM\Drivers\Builtin\<devicename>]		
[HKLM\Drivers\Builtin\PCI\Template\<templatename>]		
[HKLM\Software\Hilscher GmbH\cifX Device Driver\DeviceConfig\<DeviceNr>\<SerialNr>]		
[HKLM\Software\Hilscher GmbH\cifX Device Driver\DeviceConfig\Slot_<1..9>]		
Alias	REG_SZ	Assign a alias name for the device Note: This option should not be statically assigned, if more than one PCI card is used, as each card will get the same Alias. Use the dynamic assignment via device and serial number.
InterruptEnable	REG_DWORD	Set to 1 to enable interrupt support Default: 0 (Interrupt disabled)
DMAEnable	REG_DWORD	Set to 1 to enable DMA support Default: 0 (DMA disabled)
OSFile	REG_DWORD	Optional: Definition of a Base firmware file. To use loadable modules, a rcX base firmware is required. Default: no base firmware used
[...\Channel<0..5>]		
ModuleCount	REG_DWORD	Number of configured modules
Module0	REG_SZ	Each module gets an own entry with an index as suffix. This specifies the complete path to the file.
ConfigCount	REG_DWORD	Number of configured fieldbus databases
Config0	REG_SZ	Each database gets an own entry with an index as suffix. This specifies the complete path to the file.
WarmstartFile	REG_SZ	Full file name to warmstart parameter file

Table 14 : CE-Driver - Device Specific Settings

2.8.4 User Specific Initialization

On a custom hardware, it might be necessary to adjust the memory timing and setting parameters for the connected comX dual port memory, in order to enable the driver to correctly access it.

This user/hardware specific initialization process can be integrated into the driver startup process, by defining a special initialization DLL (definable under the '*InitDll*' registry value).

If such a DLL is defined, the driver will load it during the startup phase and tries to locate two functions within the DLL. One is an initialization function, which is called before the driver accesses the DPM the first time. The second function is called after the bootloader is downloaded to the hardware.

The following functions will be called from an initialization DLL:

Prototype/Name	Description
void CIFX_Init(uint32_t ulPhysAddr, uint32_t ulDPMSize)	Called before accessing the DPM the first time.
void CIFX_PostLoaderInit(uint32_t ulPhysAddr, uint32_t ulDPMSize, void* pvDPM)	Called after starting the bootloader (PCI cards or cards in DPM boot mode). This function can be used to adjust memory timings / bus width. If the bootloader doesn't use the netX ROM Loader defaults. (e.g. 16Bits indicated by WIF pin)

Table 15 : CE Driver - Initialization DLL Functions

2.9 Driver Setup and Test Program

Currently the cifX driver does not offer an own setup program. All driver settings are located in the registry and described before.

A dialog based test program, including the necessary function is available and can be used to test the driver functions or as a starting point for own application development.

The source of the test program is located on the cifX Windows CE driver CD.

Windows CE 5.0:

Directory	Description
EVC4	Project files for Embedded Visual C++ 4.0
.\cifXCEDriver	Source and projects files of the cifX driver and the API DLL
Application	Project files for Embedded Visual C++ 4.0
.\CifxTest	Driver test application GUI based
.\CifXTest_Console	Driver test application console based
.\CifXTCPServer	Remote driver test application
.\CifXSetup	Setup utility
PlatformBuilder	Project files for the Windows CE Platform Builder
.\ cifXCEDriver	Source and project files for the Windows CE Platform Builder

Windows CE 6.0:

Directory	Description
VS2005	Solution and project files for Visual Studio 2005
.\cifXCEDriver	Source and projects files of the cifX driver and the API DLL
Application	Test Applications for the cifX driver
.\CifxTest	Driver test application GUI based
.\CifxTest_console	Driver test application console based
.\CifXTCPServer	Remote driver test application
.\CifXSetup	Setup utility
PlatformBuilder	Project files for the Windows CE Platform Builder
.\cifXCEDriver	Source and project files for the Windows CE Platform Builder

Dependencies of the Test Program:

- cifXCEDLL.LIB cifX driver interface DLL
- cifXUser.h cifX driver interface definition file

3 Error Numbers

Value	Symbol	Description
0x00000000	CIFX_NO_ERROR	No error
0x800Axxxx		
0x800A0001	CIFX_INVALID_POINTER	An invalid pointer (NULL) was passed to the function
0x800A0002	CIFX_INVALID_BOARD	No board with the given name / index available
0x800A0003	CIFX_INVALID_CHANNEL	No channel with the given index is available
0x800A0004	CIFX_INVALID_HANDLE	An invalid handle was passed to the function
0x800A0005	CIFX_INVALID_PARAMETER	Invalid parameter passed to function
0x800A0006	CIFX_INVALID_COMMAND	Command parameter is invalid
0x800A0007	CIFX_INVALID_BUFFERSIZE	The supplied buffer does not match the expected size
0x800A0008	CIFX_INVALID_ACCESS_SIZE	Invalid Access Size (e.g. IO Area is exceeded by Offset and size)
0x800A0009	CIFX_FUNCTION_FAILED	Generic Function failure
0x800A000A	CIFX_FILE_OPEN_FAILED	A file could not be opened
0x800A000B	CIFX_FILE_SIZE_ZERO	File size is zero
0x800A000C	CIFX_FILE_LOAD_INSUFF_MEM	Insufficient memory to load file
0x800A000E	CIFX_FILE_READ_ERROR	Error reading file data
0x800A000F	CIFX_FILE_TYPE_INVALID	The given file is invalid for the operation
0x800A0010	CIFX_FILE_NAME_INVALID	Invalid filename given
0x800A0011	CIFX_FUNCTION_NOT_AVAILABLE	Function is not available on the driver
0x800A0012	CIFX_BUFFER_TOO_SHORT	The passed buffer is too short, to fit the device data
0x800A0013	CIFX_MEMORY_MAPPING_FAILED	Error mapping dual port memory
0x800A0014	CIFX_NO_MORE_ENTRIES	No more entries available (e.g. while enumerating directories)
0x800A0015	CIFX_CALLBACK_MODE_UNKNOWN	Unkown callback handling mode
0x800A0016	CIFX_CALLBACK_CREATE_EVENT_FAILED	Failed to create callback events
0x800A0017	CIFX_CALLBACK_CREATE_RECV_BUFFER	Failed to create callback receive buffer
0x800A0018	CIFX_CALLBACK_ALREADY_USED	Callback already used
0x800A0019	CIFX_CALLBACK_NOT_REGISTERED	Callback was not registerd before
0x800A001A	CIFX_INTERRUPT_DISABLED	Interrupt is disabled
0x800Bxxxx		
0x800B0001	CIFX_DRV_NOT_INITIALIZED	Driver not initialized
0x800B0002	CIFX_DRV_INIT_STATE_ERROR	Driver init state error
0x800B0003	CIFX_DRV_READ_STATE_ERROR	Driver read state error
0x800B0004	CIFX_DRV_CMD_ACTIVE	Command is active on device
0x800B0005	CIFX_DRV_DOWNLOAD_FAILED	General error during download
0x800B0006	CIFX_DRV_WRONG_DRIVER_VERSION	Wrong driver version

Table 16: Error Numbers (1)

Value	Symbol	Description
0x800B0030	CIFX_DRV_DRIVER_NOT_LOADED	CIFx driver is not running
0x800B0031	CIFX_DRV_INIT_ERROR	Failed to initialize the device
0x800B0032	CIFX_DRV_CHANNEL_NOT_INITIALIZED	Channel not initialized (xOpenChannel() not called)
0x800B0033	CIFX_DRV_IO_CONTROL_FAILED	IOControl call failed
0x800B0034	CIFX_DRV_NOT_OPENED	Driver was not opened
0x800B0040	CIFX_DRV_DOWNLOAD_STORAGE_UNKNOWN	Unknown download storage type (RAM/FLASH based) found
0x800B0041	CIFX_DRV_DOWNLOAD_FW_WRONG_CHANNEL	Channel number for a firmware download not supported
0x800B0042	CIFX_DRV_DOWNLOAD_MODULE_NO_BASEOS	Modules are not allowed without a Base OS firmware
0x800C0010	CIFX_DEV_DPM_ACCESS_ERROR	Dual port memory not accessible (board not found)
0x800C0011	CIFX_DEV_NOT_READY	Device not ready (ready flag failed)
0x800C0012	CIFX_DEV_NOT_RUNNING	Device not running (running flag failed)
0x800C0013	CIFX_DEV_WATCHDOG_FAILED	Watchdog test failed
0x800C0015	CIFX_DEV_SYSERR	Error in handshake flags
0x800C0016	CIFX_DEV_MAILBOX_FULL	Send mailbox is full
0x800C0017	CIFX_DEV_PUT_TIMEOUT	Send packet timeout
0x800C0018	CIFX_DEV_GET_TIMEOUT	Receive packet timeout
0x800C0019	CIFX_DEV_GET_NO_PACKET	No packet available
0x800C001A	CIFX_DEV_MAILBOX_TOO_SHORT	Mailbox is too short for a packet
0x800C0020	CIFX_DEV_RESET_TIMEOUT	Reset command timeout
0x800C0021	CIFX_DEV_NO_COM_FLAG	Communication flag not set
0x800C0022	CIFX_DEV_EXCHANGE_FAILED	I/O data exchange failed
0x800C0023	CIFX_DEV_EXCHANGE_TIMEOUT	I/O data exchange timeout
0x800C0024	CIFX_DEV_COM_MODE_UNKNOWN	Unknown I/O exchange mode
0x800C0025	CIFX_DEV_FUNCTION_FAILED	Device function failed
0x800C0026	CIFX_DEV_DPMSIZE_MISMATCH	DPM size differs from configuration
0x800C0027	CIFX_DEV_STATE_MODE_UNKNOWN	Unknown state mode
0x800C0028	CIFX_DEV_HW_PORT_IS_USED	Device is still accessed
0x800C0029	CIFX_DEV_CONFIG_LOCK_TIMEOUT	Configuration locking timeout
0x800C002A	CIFX_DEV_CONFIG_UNLOCK_TIMEOUT	Configuration unlocking timeout
0x800C002B	CIFX_DEV_HOST_STATE_SET_TIMEOUT	Set HOST state timeout
0x800C002C	CIFX_DEV_HOST_STATE_CLEAR_TIMEOUT	Clear HOST state timeout
0x800C002D	CIFX_DEV_INITIALIZATION_TIMEOUT	Timeout during channel initialization
0x800C002E	CIFX_DEV_BUS_STATE_ON_TIMEOUT	Timeout setting bus on flag
0x800C002F	CIFX_DEV_BUS_STATE_OFF_TIMEOUT	Timeout setting bus off flag
0x800C0040	CIFX_DEV_MODULE_ALREADY_RUNNING	Module already running
0x800C0041	CIFX_DEV_MODULE_ALREADY_EXISTS	Module already exists

Table 17: Error Numbers (2)

Value	Symbol	Description
0x800C0050	CIFX_DEV_DMA_INSUFF_BUFFER_COUNT	Number of configured DMA buffers insufficient
0x800C0051	CIFX_DEV_DMA_BUFFER_TOO_SMALL	DMA buffers size too small (min size 256Byte)
0x800C0052	CIFX_DEV_DMA_BUFFER_TOO_BIG	DMA buffers size too big (max size 63,75KByte)
0x800C0053	CIFX_DEV_DMA_BUFFER_NOT_ALIGNED	DMA buffer alignment failed (must be 256Byte)
0x800C0054	CIFX_DEV_DMA_HANSHAKEMODE_NOT_SUPPORTED	I/O data uncontrolled handshake mode not supported
0x800C0055	CIFX_DEV_DMA_IO_AREA_NOT_SUPPORTED	I/O area in DMA mode not supported (only area 0 possible)
0x800C0056	CIFX_DEV_DMA_STATE_ON_TIMEOUT	Set DMA ON Timeout
0x800C0057	CIFX_DEV_DMA_STATE_OFF_TIMEOUT	Set DMA OFF Timeout
0x800C0058	CIFX_DEV_SYNC_STATE_INVALID_MODE	Device is in invalid mode for this operation
0x800C0059	CIFX_DEV_SYNC_STATE_TIMEOUT	Waiting for synchronization event bits timed out

Table 18: Error Numbers (3)

4 Appendix

4.1 List of Tables

Table 1: List of Revisions	3
Table 2: Terms, Abbreviations and Definitions	4
Table 3: References	4
Table 4: CD Contents	9
Table 5: Platform Builder - Project Directory	12
Table 6: EVC4 - Project Directories	14
Table 7: Application - Project Directories	15
Table 8: Platform Builder - Project Directories	16
Table 9: Visual Studio 2005 - Project Directories	18
Table 10: Application - Project Directories	20
Table 11 : CE Driver - Files to Install.....	21
Table 12 : CE-Driver - Basic Driver and Device Configuration	26
Table 13 : CE-Driver - Global Driver Settings.....	27
Table 14 : CE-Driver - Device Specific Settings.....	28
Table 15 : CE Driver - Initialization DLL Functions.....	29
Table 16: Error Numbers (1)	31
Table 17: Error Numbers (2)	32
Table 18: Error Numbers (3)	33

4.2 List of Figures

Figure 1 : CifX CE Driver - Architecture.....	7
---	---

4.3 Contacts

Headquarters

Germany

Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support

Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China

Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support

Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France

Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support

Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India

Hilscher India Pvt. Ltd.
New Delhi - 110 025
Phone: +91 11 40515640
E-Mail: info@hilscher.in

Italy

Hilscher Italia srl
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support

Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan

Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support

Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea

Hilscher Korea Inc.
Suwon, 443-810
Phone: +82-31-204-6190
E-Mail: info@hilscher.kr

Switzerland

Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support

Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA

Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support

Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com