



Driver Manual
cifX Device Driver
QNX 6.4
V1.0.x.x

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC100702DRV01EN | Revision 1 | English | 2010-07 | 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	Overview.....	5
1.6	Requirement.....	6
1.7	Features.....	6
1.8	Limitations.....	6
1.9	CD Contents.....	7
1.10	Legal Notes.....	8
1.10.1	Copyright.....	8
1.10.2	Important Notes.....	8
1.10.3	Exclusion of Liability.....	9
1.10.4	Export.....	9
2	Licensing Terms.....	10
3	Building the cifX Driver.....	11
3.1	Firmware and Configuration File Storage.....	14
4	QNX Driver Specific Information.....	16
4.1	Additional Structures.....	16
4.1.1	Structure QNX_CIFXDRV_PARAMETERS_T.....	16
4.1.2	Structure QNX_CIFXDRV_DEVICEENTRY_T.....	17
4.1.3	Structure QNX_CIFXDRV_DEVICEBUSINFO_T.....	17
4.2	Additional Functions.....	18
4.2.1	cifXInitDriver ().....	18
4.2.2	cifXDeinitDriver().....	19
4.2.3	cifXFindDevice().....	20
4.2.4	cifXGetDriverVersion().....	21
4.3	Driver Startup Procedure.....	22
4.4	Device Configuration (device.conf).....	23
5	Using SYCON.net to Configure the Fieldbus System.....	24
6	Programming with the cifX QNX Driver.....	25
7	Appendix.....	26
7.1	List of Tables.....	26
7.2	List of Figures.....	26
7.3	Contacts.....	27

1 Introduction

1.1 About this Document

The QNX Neutrino RTOS is a robust real-time operating system, which meets the constrained resource requirements of realtime embedded systems.

This manual describes the Hilscher cifX driver for QNX and its architecture. The driver offers access to the Hilscher netX based hardware (e.g. CIFX 50, COMX) with the same functional API as the cifX device driver for Windows and offers transparent access to the different devices.

1.2 List of Revisions

Rev	Date	Name	Chapter	Revision
1	2010-07-01	SS	All	Created

Table 1: List of Revisions

1.3 Terms, Abbreviations and Definitions

Term	Description
cifX	Communication Interface based on netX
comX	Communication Module based on netX
PCI	Peripheral Component Interconnect
API	Application Programming Interface
DPM	Dual-Port Memory Physical interface to all communication board (DPM is also used for PROFIBUS-DP Master).
DMA	Direct Memory Access

Table 2 : Terms and Abbreviations

1.4 References

This document based on the following documents:

- [1] Hilscher Gesellschaft für Systemautomation mbH: Driver Manual cifX Device Driver - Windows 2000/XP/Vista/7/CE V1.0.x.x. Revision 15, english, 2010
- [2] Hilscher Gesellschaft für Systemautomation mbH: Program Reference Guide - netX Diagnostic and Remote Access - Fundamentals V1.0.x.x. Revision 1, english, 2010
- [3] Hilscher Gesellschaft für Systemautomation mbH: Program Reference Guide - netX Diagnostic and Remote Access - Target Device V2.0.x.x. Revision 3, english, 2010

Table 3: References

1.5 Overview

The cifX QNX driver is available as a static library or a shared object, which is built around the cifX Toolkit. Any application which needs to access a cifX device can use the device specific functions provided by this driver library. The concept of the cifX device driver is illustrated in the subsequent figure.

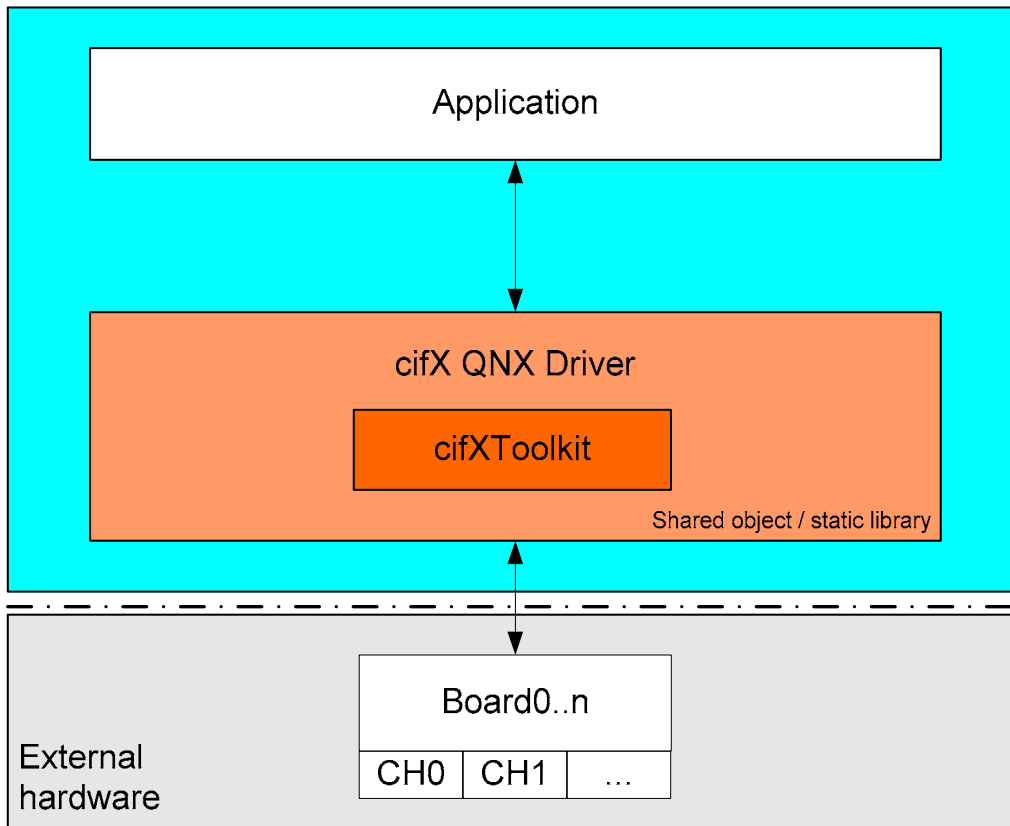


Figure 1 : cifX Driver Architecture for QNX

1.6 Requirement

- QNX Neutrino RTOS V6.4.0 or higher
- QNX Momentics IDE V4.5.0 or higher

1.7 Features

- Based on the cifX Toolkit source (V1.0.1.0)
- Unlimited number of cifX boards supported
- Support for NXSB-PCA or NX-PCA-PCI boards included
- Interrupt support for PCI based devices
- DMA data transfer for I/O data
- Support for loadable modules
- Interrupt notification for applications

1.8 Limitations

- No DMA support for NXSB-PCA and NX-PCA-PCI boards
- Only one application can access a card simultaneously. For multi-application access to a single card, a special application needs to be implemented by user
- Driver does compile for platforms ARM, PowerPC and SH-4, but functionality is not tested on target system

1.9 CD Contents

Content		
Documentation		Driver Documentation
Driver		cifX driver for QNX
	x86_debug	Driver library for x86 Platforms (Includes Debug Information)
	x86_release	Driver library for x86 Platforms
Sources		QNX Momentics IDE projects
	cifXDrv	cifX Driver Sources
	cifXToolkit	cifX Toolkit Sources
	cifXDrvTest	cifX Driver Test Application Sources
	cifXTCPServer	Source of TCP/IP Server for cifX Devices
Examples BaseDir		Example Card Configuration Directory (copy to your own base dir.)
Diagnostic and Remote Access		netX Diagnostic and Remote Access Components
	HilTransport Dissector Wireshark	Wireshark Dissector for the Hilscher Transport Protocol
	Host Device Components	Win32 Test Application for Remote Access
	Target Device Components	TCP/IP Server for cifX Devices (QNX, x86 Platforms)

Table 4 : CD Contents

1.10 Legal Notes

1.10.1 Copyright

© 2010 Hilscher Gesellschaft für Systemautomation mbH

All rights reserved.

The images, photographs and texts in the accompanying material (user 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.

1.10.2 Important Notes

The user 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 user 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 user 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 user 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.

1.10.3 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.

1.10.4 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 Licensing Terms

The cifX QNX driver offers full source code for the netX chip DPM adaptation to QNX.

The source code can be used for internal development, modification and debugging purpose.

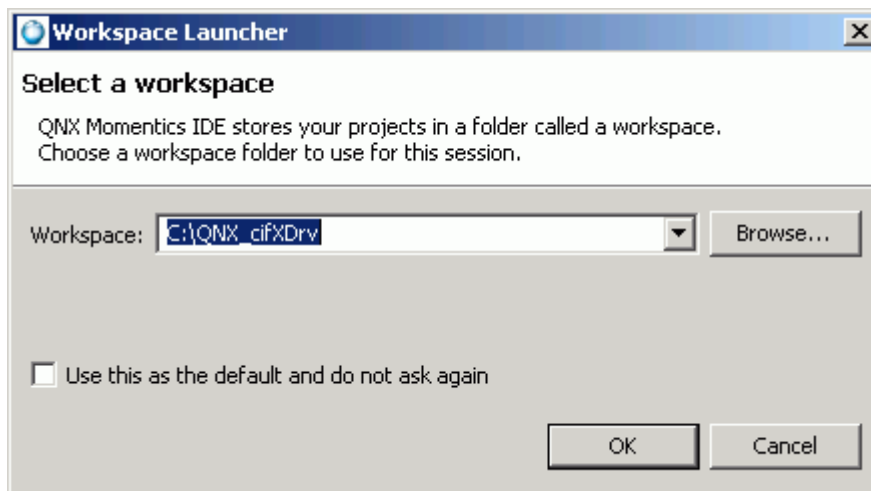
Distribution of the original source code, parts of the source code or modifications based on it is prohibited.

Binary distribution for use in products is allowed.

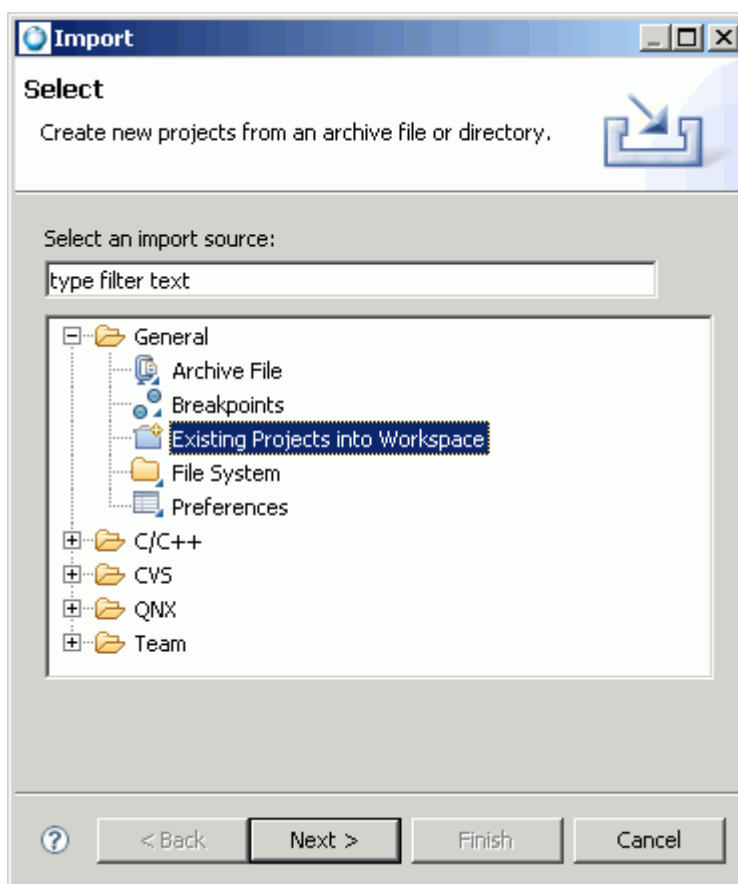
3 Building the cifX Driver

The installation CD includes project files for the development environment QNX Momentics. In addition to the driver source, the project contain an example application. To create a new workspace for your cifx application development follow this steps:

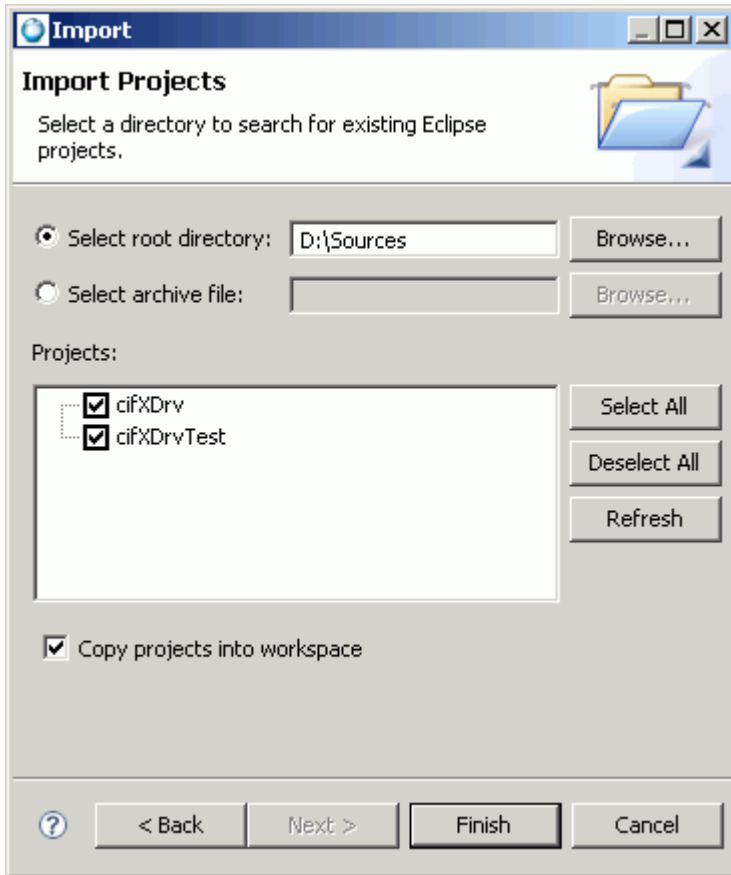
- Start the QNX Momentics IDE and select a clean workspace directory (e.g. C:\QNX_cifXDrv)



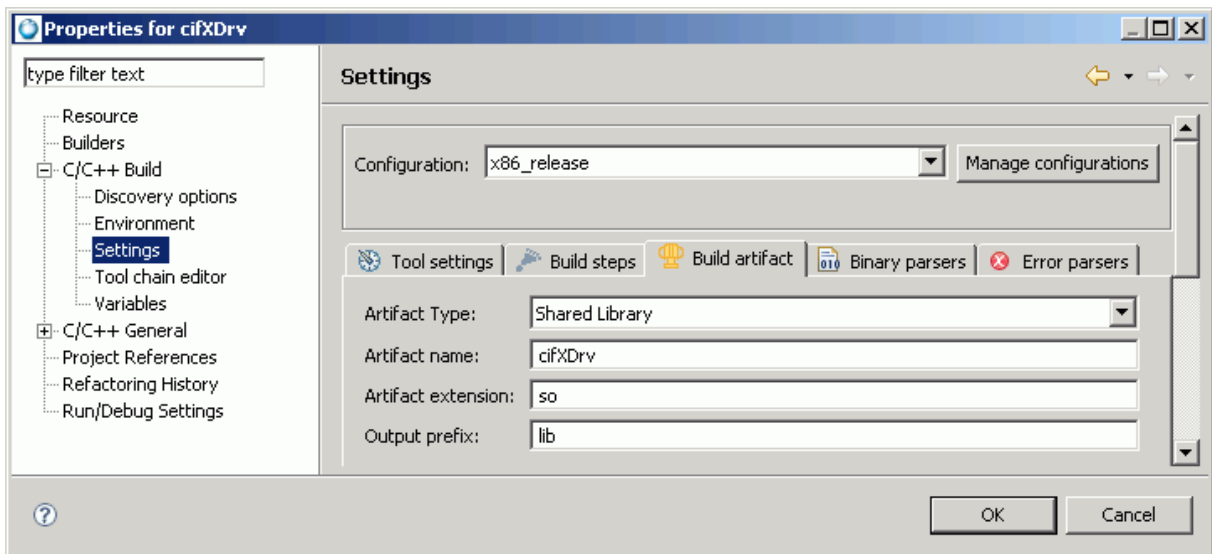
- Open the <File> entry from the menu bar and select <Import...>
- Extend the <General> folder, select <Existing Projects into Workspace> and press <Next>



- Browse to the directory *Sources* on the installation CD and select the project *<cifXDrv>*. If you need the example application, please select the *<cifXDrvTest>* project, too. Make sure that the option *<Copy projects into workspace>* is selected.



- Press *<Finish>* to import the selected projects into your workspace.
- In default state the cifX driver will be build as a shared object. If you want to build the cifX driver as a static library, you need to change the artifact type in the *<build artifact>* tab of the project properties.



To use a cifX device on your QNX environment the device driver must be initialized with the QNX specific driver routines, described in section 4. The cifX device driver initialization can be processed in an application by calling the driver initialization routines inside the application. The following C application demonstrates the minimum functions which must be called to enable an application to work with a cifX device.

```
#include <stdio.h>
#include <cifXQNX.h>
#include <cifXUser.h>
#include <cifXErrors.h>

/*****
/*! The main function
*   \return 0 on success
*****/
int main(int argc, char* argv[])
{
    QNX_CIFXDRV_PARAMETERS_T tDriverParams = {0};
    CIFXHANDLE                hDriver      = NULL;
    long                      lRet         = CIFX_NO_ERROR;

    /* Driver scans for all available cards */
    tDriverParams.fScanPCI      = TRUE;

    /* Set the trace level of the driver */
    tDriverParams.ulTraceLevel = TRACE_LEVEL_ERROR;

    /* Set the base directory of the driver */
    tDriverParams.szDriverBaseDir = "/opt/cifX";

    /* Do not use single firmware directory */
    tDriverParams.fSingleDir = FALSE;

    /* Polling intervall in milliseconds for non-interrupt cards */
    tDriverParams.ulPollInterval = 500;

    /* No DPM cards will be added to the driver */
    tDriverParams.ulUserDevCount = 0;
    tDriverParams.ptUserDevList  = NULL;

    /* Init the driver */
    cifXInitDriver(&tDriverParams);

    /* Open the cifX driver */
    lRet = xDriverOpen(&hDriver);
    if(CIFX_NO_ERROR != lRet)
    {
        printf("Error opening driver. lRet=0x%08lX\r\n", lRet);
    } else
    {
        /* Work with the cifX device */
        /* ... */
        /* Close the cifX driver */
        xDriverClose(hDriver);
    }
    return 0;
}
```

3.1 Firmware and Configuration File Storage

cifX PCI cards are not using any flash memory to store a firmware or configuration on the card. Every time the card is powered-up the firmware and configuration must be downloaded to the hardware.

Note: Firmware and configurations are not stored on the hardware and must be downloaded each time the card is powered-up.

It is the task of the driver to initialize the card and therefore the driver has to know which files must be loaded to the hardware. To allow device specific configuration, every file that needs to be downloaded must be stored in an own folder. These folder reside under a global base folder and must be passed during driver initialization (Parameter *szDriverBaseDir*, see section 4.1.1).

To assign the firmware and configuration files to a cifX device clearly, the driver provides the options below:

- If only one cifX device needs to be supported, a predefined directory can be used by setting the driver parameter *fSingleDir* (see section 4.1.1) accordingly. The firmware and configuration file must reside in the subdirectory *FW*.
- The Slot Number serves to distinguish cifX cards from each other clearly, especially if more cifX cards are installed into the very same PC. The Slot Number must be set at the cifX card using the Rotary Switch Slot Number. While Slot Number 0 means, that the cifX card is identified via its device and serial number, values from 1 to 9 corresponds to the Slot Number 1 to 9. The firmware and configuration file must reside in the subdirectory *Slot_<1..9>*.
- If the cifX device is not equipped with a Rotary Switch or the Slot Number should not be used, the device is identified by its device and serial number. The firmware and configuration file must reside in the subdirectory *<Device Number>_<Serial Number>*.

The following table describes the different subdirectory levels:

Subdirectory	Description
<BASEDIR>	Base directory (<i>szDriverBaseDir</i> , see section 4.1.1) Must be set during driver initialization. This directory must contain the second stage PCI bootloader (e.g. NETX100-BSL.bin)
FW Slot_<1..9> <Device Nr>_<Serial Nr>	Device and serial number of the device or slot number if the device provides a rotary switch. If the slot number is 0 the device and serial number is used to identify the device. The files always resides in the single directory if the corresponding option is used. Note: The configuration file (<i>device.conf</i>) must reside here! Note: This directory must contain the rcX base firmware if loadable modules are used.
Channel<#>	Channel specific files (loadable modules, monolithic firmware files, fieldbus database files) Note: Currently only channel 0 is supported

Table 5 : Firmware and Configuration File Storage

Sample file structure for a cifX device with device number 1250100 and serial number 20217:

```
+ <BASEDIR>
|
|-- NETX100-BSL.BIN (second stage PCI bootloader)
|
|--+ 1250100_20217
|   |--+ Channel0
|       |--cifXdps.nxf (monolithic firmware)
|       |--config.nxd (fieldbus database)
|   |--+ Channel1
|   |--+ Channel2
|   |--+ Channel3
|   |--+ Channel4
|   |--+ Channel5
|   |-- device.conf (configuration file)
```

Sample file structure for a cifX device identified by Slot number 2 and loadable module support:

```
+ <BASEDIR>
|
|-- NETX100-BSL.BIN (second stage PCI bootloader)
|
|--+ Slot_2
|   |--+ Channel0
|       |--nx100dpm.nxo (loadable module)
|       |--config.nxd (fieldbus database)
|   |--+ Channel1
|   |--+ Channel2
|   |--+ Channel3
|   |--+ Channel4
|   |--+ Channel5
|   |-- device.conf (configuration file)
|   |-- cifXrcX.nxf (rcX base firmware)
```

4 QNX Driver Specific Information

The QNX driver needs some special initialization functions and structures as described in section 1.

4.1 Additional Structures

Some of the QNX specific functions need parameters provided through structures. The structures and the meaning of the internal data are described in the following chapter.

4.1.1 Structure QNX_CIFXDRV_PARAMETERS_T

This structure is used to initialize the cifX driver.

Element	Data Type	Description
fScanPCI	int	<p>0 = CIFX_DRIVER_INIT_NOSCAN / FALSE Driver does not scan for PCI cards. Cards can be added manually by <i>ulUserDevCount</i> and <i>ptUserDevList</i> parameters.</p> <p>1 = CIFX_DRIVER_INIT_AUTOSCAN / TRUE Driver scans for all available PCI cards and adds them to the application.</p>
ulTraceLevel	uint32_t	<p>Set the trace level of the driver:</p> <p>1 = TRACE_LEVEL_DEBUG 2 = TRACE_LEVEL_INFO 4 = TRACE_LEVEL_WARNING 8 = TRACE_LEVEL_ERROR</p>
ulPollInterval	uint32_t	<p>Polling interval in milliseconds [ms] for non-interrupt cards (used for Change of State (COS) signaling) Use <i>CIFX_POLLINTERVAL_DISABLETHREAD</i> to disable polling completely</p> <p>0 = 500ms default</p>
iPollPriority	int	Priority of the polling thread
szDriverBaseDir	const char*	<p>Set the base directory of the driver, can be NULL to use the default of <i>'/opt/cifX'</i></p> <p>NULL = '/opt/cifX' default</p>
fSingleDir	int	<p>The driver will only look into <i>'/szDriverBaseDir/FW'</i> for the firmware. This can be used on single cifX OEM devices to prevent the need for a unique cifX device directory (NOTE: This only support one cifX device)</p> <p>0 = FALSE</p>
ulUserDevCount	uint32_t	<p>Number of user cards entries in the <i>ptUserDevList</i> to add to the driver (e.g. if a card is connected via DPM).</p> <p>0 = none</p>
ptUserDevList	QNX_CIFXDRV_DEVICEENTRY_T *	Array of user added cards. Number of entries are defined by <i>ulUserDevCount</i> . See section 4.1.2.

Table 6 : Structure Definition of QNX_CIFXDRV_PARAMETERS_T

4.1.2 Structure QNX_CIFXDRV_DEVICEENTRY_T

This structure describes a cifX device which should be added to the driver. This structure must be filled by the user if a custom DPM based card should be added.

Element	Data Type	Description
ulPhysicalAddress	uint32_t	Physical address of the DPM (this value is used to detect the PCI card linked to the DPM)
pvDPMAddress	void*	Virtual Pointer to card DPM
ulDPMSize	uint32_t	Size of the DPM in bytes
blrqNumber	uint8_t	Interrupt number
fPCICard	int	0 = FALSE Device is connected via DPM. 1 = TRUE Device is connected to PCI bus
pfnNotify	PFN_CIFX_NOTIFY_EVENT	Callback that is made at several stages when initializing a device. This allows the user to setup DPM and timings (if they are different from the netX ROM Loader settings) Pass NULL to suppress callback
tBusInfo	QNX_CIFXDRV_DEVICEBUSINFO_T	Bus information, see section 4.1.3

Table 7 : Structure Definition of QNX_CIFXDRV_DEVICEENTRY_T

4.1.3 Structure QNX_CIFXDRV_DEVICEBUSINFO_T

Bus information structure used to store bus specific information.

Element	Data Type	Description
uiBusNumber	uint16_t	Bus number
uiDevFuncNumber	uint16_t	Device/Function number. The device number is in bits 7 through 3, and the function number in bits 2 through 0.

Table 8 : Structure Definition of QNX_CIFXDRV_DEVICEBUSINFO_T

4.2 Additional Functions

This chapter describes functions which are only available for the QNX version of the driver. These functions are used to initialize the cifX device driver. The driver initialization can be performed inside an application or at system startup.

4.2.1 cifXInitDriver ()

This function must be called before accessing any driver function. The cifX driver initialization includes discovering all available cifX PCI devices and downloading the firmware and configuration files.

Note: If PCI scan is enabled via the driver parameter *fScanPCI* (see section 4.1.1) the *cifXInitDriver()* search and initialize every available cifX PCI card.

Note: For none PCI cards, the application has to create a `QNX_CIFXDRV_DEVICEENTRY_T` with the corresponding device information. Afterwards, the structure must be handed to the *cifXInitDriver()* routine via the driver parameters *ulUserDevCount* and *ptUserDevList* (see section 4.1.1).

Function call:

```
int32_t cifXInitDriver (QNX_CIFXDRV_PARAMETERS_T* ptDriverParams);
```

Arguments:

Argument	Data Type	Description
ptDevEntry	QNX_CIFXDRV_PARAMETERS_T*	Driver parameters, see section 4.1.1

Return Values:

Return Values	
CIFX_NO_ERROR	Driver initialization successful
CIFX_DRV_INIT_ERROR	Driver initialization failed (no cifX device available)

Example:

```
QNX_CIFXDRV_PARAMETERS_T tDriverParams = {0};

/* Set driver parameters */
tDriverParams.fScanPCI = TRUE;
tDriverParams.ulUserDevCount = 0;

/* Scan for all available cifX PCI devices and initialize the cifX device driver */
cifXInitDriver (&tDriverParams);
```

4.2.2 cifXDeinitDriver()

Deinitialize the driver and remove all devices from the control of the cifX driver library. After calling this function the application must not access any cifX driver API function any more.

Function call:

```
int32_t cifXDeinitDriver ( void );
```

Arguments:

Argument	Data Type	Description
none		

Return Values:

Return Values	
CIFX_NO_ERROR	Driver deinitialization successful
CIFX_DEV_HW_PORT_IS_USED	At least one channel has an open reference so deny deinitialization

4.2.3 cifXFindDevice()

This function scans for a cifX PCI device in the system and builds a QNX_CIFXDRV_DEVICEENTRY_T structure for each discovered device. The discovered device information can be handed to the driver initialization routine via the driver parameters *ulUserDevCount* and *ptUserDevList* (see 4.1.1). The PCI scan of the driver initialization routine should be disabled via the driver parameter *fScanPCI*, to prevent the discovery of every available PCI device in the system.

Function call:

```
BOOL cifXFindDevice (QNX_CIFXDRV_DEVICEENTRY_T* ptDevEntry,
                   int iNum);
```

Arguments:

Argument	Data Type	Description
ptDevEntry	QNX_CIFXDRV_DEVICEENTRY_T*	Pointer to a QNX_CIFXDRV_DEVICEENTRY_T structure, to place returned values in
iNum	int	Number of the device in the system 0 = first device

Return Values:

Return Values	
TRUE	A device with number iNum was found
FALSE	A device with number iNum could not be found

Example:

```
QNX_CIFXDRV_PARAMETERS_T tDriverParams = {0};
QNX_CIFXDRV_DEVICEENTRY_T tDevEntry = {0};

/* Find the first cifX PCI device */
if (cifXFindDevice (&tDevEntry, 0))
{
    tDriverParams.fScanPCI = FALSE;
    tDriverParams.ulUserDevCount = 1;
    tDriverParams.ptUserDevList = &tDevEntry;

    /* initialize the cifX device driver */
    cifXInitDriver (&tDriverParams);
}
```

4.2.4 cifXGetDriverVersion()

This function returns the version of the cifX driver for QNX.

Function call:

```
int32_t cifXGetDriverVersion ( uint32_t ulSize, char* szVersion);
```

Arguments:

Argument	Data Type	Description
ulSize	unsigned long	Size of buffer referenced by parameter <i>szVersion</i>
szVersion	char*	Buffer to return driver version string

Return Values:

Return Values	
CIFX_NO_ERROR	Memory mapping successful
CIFX_INVALID_BUFFERSIZE	Size of supplied buffer is too small

Example:

```
char szDrvVersion[20] = "";  
  
/* Get driver version */  
cifXGetDriverVersion( sizeof(szDrvVersion)/sizeof(*szDrvVersion), szDrvVersion);  
  
/* Print driver version to screen */  
printf("%s", szDrvVersion);
```

4.3 Driver Startup Procedure

The driver startup procedure can be controlled by the user.

The following two use cases are available:

- Automatically add all found cifX PCI devices and optionally add user specific devices
- Skip cifX PCI device scan and only add user specific device

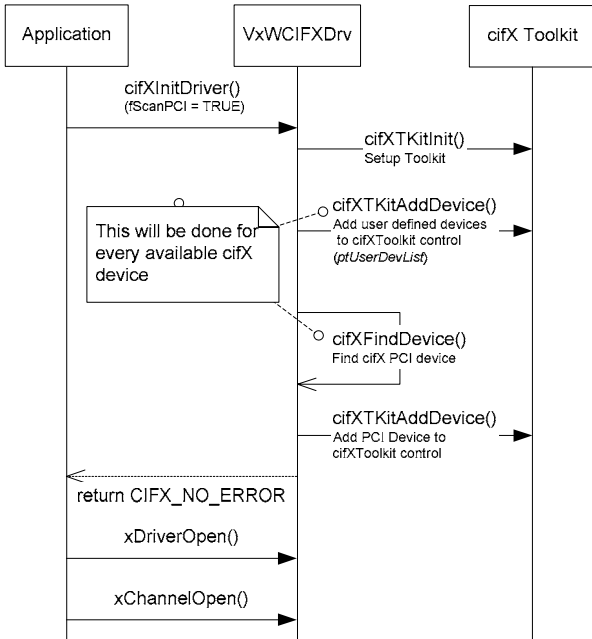


Figure 2 : Initialization of the cifX Driver with fScanPCI = CIFX_DRIVER_INIT_AUTOSCAN / TRUE

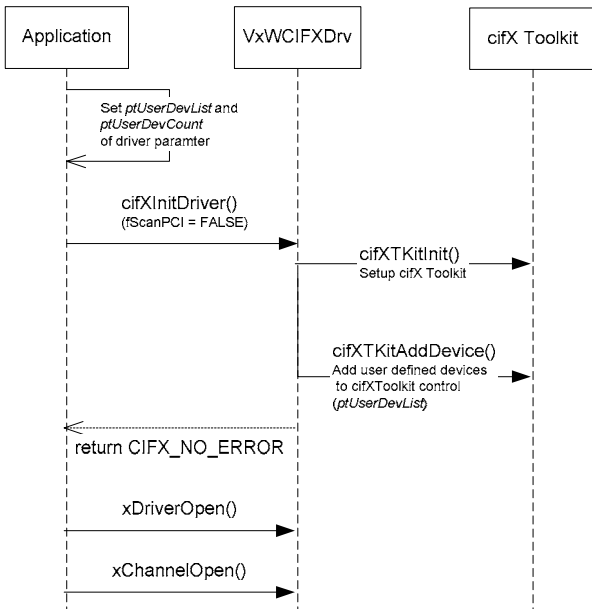


Figure 3 : Initialization of the cifX Driver with fScanPCI = CIFX_DRIVER_INIT_NOSCAN / FALSE

4.4 Device Configuration (device.conf)

Each device has several parameters which can be configured via a per device configuration file. Available parameters are shown below.

The configuration file must be called *'device.conf'* and must reside in the top level directory of a device, e.g. (base directory *'/hd0/cifx'*):

- *'/hd0/cifx/1250100_20217/device.conf'* if a device is identified by its device/serial number
- *'/hd0/cifx/Slot_1/device.conf'* if a device is identified by slot number 1
- *'/hd0/cifx/FW/device.conf'* if a single directory is used

Note: Currently DMA support is only provided by cifX50 and netPLC PCI cards.

The file may contain the following keys:

Key	Data Type	Description
alias	char[16]	Alias name for the device. Must be less than 16 characters.
irq	string	'yes' = enable IRQ on the device 'no' = disable IRQ on the device
dma	string	'on' = Switch channels into DMA mode 'off' = Switch off DMA mode for all channels

Table 9 : Device Configuration Parameters - device.conf

Sample device.conf:

```
#Sample device configuration file
alias=PROFIBUS
irq=yes
dma=on
```

5 Using SYCON.net to Configure the Fieldbus System

The Hilscher fieldbus hardware will be configured by a Windows application called SYCON.net. SYCON.net is based on the FDT/DTM concept and generates the configuration files for the hardware. It is also able to update the firmware for a specific card.

Please use the following steps to create a configuration:

- Install SYCON.net
- Open SYCON.net and create a configuration
- Store the SYCON.net configuration project and export the configuration from SYCON.net into a database file.
- Copy the database and the firmware files to the device configuration directory.
- Now start/restart the cifX Linux driver. This will load the firmware and configuration into the cifX card.

SYCON.net is also able to connect to a remote device supporting the Hilscher "cifX Diagnostics and Remote Access" functions. Please consult [2] and [3] for further information about the remote access interface.

An example standalone TCP/IP server, offering these functions, can be found in the sources directory of the QNX driver CD.

Note: Currently, the cifXTCPServer does not support firmware and configuration download.

6 Programming with the cifX QNX Driver

The Application Programming Interface (API) of the Hilscher QNX driver is based on the already known cifX Device Driver Interface [1]. Therefore the cifX Device Driver manual can be used. This manual describes the driver functions (API), error codes and shows some program examples. The installation CD also includes a '*cifXDrvTest*' directory with a QNX specific example.

7 Appendix

7.1 List of Tables

Table 1: List of Revisions	3
Table 2 : Terms and Abbreviations	4
Table 3: References	4
Table 4 : CD Contents.....	7
Table 5 : Firmware and Configuration File Storage.....	14
Table 6 : Structure Definition of QNX_CIFXDRV_PARAMETERS_T.....	16
Table 7 : Structure Definition of QNX_CIFXDRV_DEVICEENTRY_T.....	17
Table 8 : Structure Definition of QNX_CIFXDRV_DEVICEBUSINFO_T	17
Table 9 : Device Configuration Parameters - device.conf	23

7.2 List of Figures

Figure 1 : cifX Driver Architecture for QNX.....	5
Figure 2 : Initialization of the cifX Driver with fScanPCI = CIFX_DRIVER_INIT_AUTOSCAN / TRUE.....	22
Figure 3 : Initialization of the cifX Driver with fScanPCI = CIFX_DRIVER_INIT_NOSCAN / FALSE	22

7.3 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