

**Driver Manual**

**INtime® Real-Time Extension for Windows® Device  
Driver for the Hilscher GmbH CIF/COM line of Fieldbus  
Devices**

---

Index	Date	Version	Chapter	Revision
1	27.07.04	1	All	created
2	28.07.04	1	All	1.01
3	09.09.04	1	All	1.02
4	10.15.10	1.1	All	1.10

**Please notice that software and hardware names, used in this manual, are normally protected by trademarks or patents of corresponding companies.**

---

<b>1</b>	<b>INTRODUCTION</b>	<b>5</b>
1.1	Implementation	5
1.1.1	INtime CIF Driver Windows Service	5
1.1.2	Simultaneous INtime and Windows application access support	6
1.1.3	INtime CIF Driver Topology	7
1.2	Versions, Features and Limitations	8
1.3	Supported CIF Cards, Fieldbus System and Protocols	9
<b>2</b>	<b>INSTALLATION</b>	<b>10</b>
<b>3</b>	<b>FIELDBUS CONFIGURATION DETAILS</b>	<b>11</b>
3.1	Fieldbus Startup Behaviour	11
3.2	Supported Handshake Modes	12
<b>4</b>	<b>DRIVER FUNCTIONS</b>	<b>13</b>
<b>5</b>	<b>PROGRAMMING</b>	<b>13</b>
5.1	API Files	13
5.2	Calling Sequence	14
5.3	Example Program INtimeCIFTTest	15
5.4	Example Program WinCIFTTest	15
5.5	Programming hints	15
<b>6</b>	<b>ADDITIONAL INFORMATION</b>	<b>16</b>
6.1	Hilscher GmbH Reference Manuals	16
6.2	Multiple Hilscher CIF Card Support	16
6.3	INtime Node Browser	17
6.4	Limitations to CIF Driver Access from a Windows Application	17
6.5	Windows Handling of Error -25 DRV_DEV_FUNCTION_FAILED	20
6.6	Use of Windows-based Driver Test Program to Monitor Fieldbus systems driven by INtime Applications	20
6.7	Use of Hilscher GmbH SyCon Program with the INtime CIF Driver	21
<b>7</b>	<b>ERROR NUMBERS</b>	<b>22</b>
7.1	List of Error Numbers (Courtesy of Hilscher GmbH)	22

---

---

**7.2 Additional Error Information..... 25**

# 1 Introduction

TenAsys Corporation's INtime product is a real time extension for Microsoft Windows.

Please refer to section '**1.2 Versions, Features, and Limitations**' which lists supported Windows, INtime, and INtime CIF Driver version combinations.

The INtime CIF Driver is a real time (RT) Driver (CIFINtimeDrv.rta) that supports CIF30 (ISA), CIF50 (PCI) and CIF80 (CompactPCI) cards.

Please refer to section '**1.3 Supported CIF Cards and Fieldbus Systems**' which lists the fieldbus systems and devices that are currently supported by the driver.

## 1.1 Implementation

The INtime CIF driver API is based on, and is identical to, the existing CIF device driver API for Windows defined by Hilscher GmbH.

The driver offers transparent access to the different devices. Therefore it hides the functional differences for ISA (including PC104), PCI and CompactPCI cards. The cards can be selected by board number between 0 and 3.

### 1.1.1 INtime CIF Driver Windows Service

The INtime CIF Driver Windows Service is provided as part of the INtime CIF Driver package. Once installed, it allows the INtime CIF driver to be started from Windows either manually using the Services Applet in the Windows Control Panel (Start=>Control Panel=>Administrative Tools=>Services) or automatically when Windows is booted. This Automatic Start feature is selected in the Services Applet.

By default, the INtime CIF Driver Windows Service will load the INtime CIF Driver for the first Hilscher CIF card found in the system.

See Section 6.2 for information on how to support additional instances of the Hilscher cards in a system (up to 4).

### 1.1.2 Simultaneous INtime and Windows application access support

The INtime CIF driver uses the INtime Service/Message Ports model to allow multiple users to access the device managed by the driver in a safe manner. The driver is packaged as an INtime Service with a Name corresponding to the Board number assigned to the hardware.

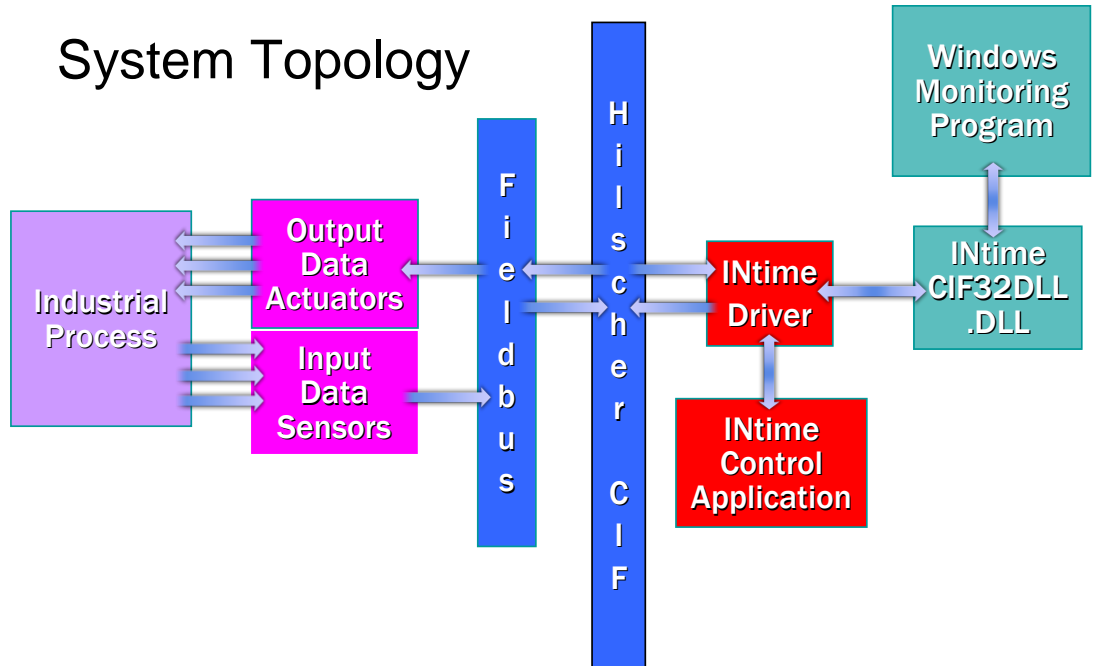
Each CIF API call is translated into Messages sent via Message Ports to the INtime CIF Driver Service. The translation is handled by the interface library <INtime Install Path>\rt\lib\CIFrtlib.lib. The standard Hilscher header file (cifuser.h) provides the function prototypes.

INtime services are visible to, and accessible from a Windows application via a Windows Extension (ntx) API. Part of the INtime CIF Driver package is a replacement DLL named cif32dll.dll. This DLL replaces the one provided by Hilscher GmbH with their Windows driver. Once installed, the TenAsys cif32dll.dll DLL routes calls from Windows applications such as SyCon and Msg\_Dbg (both from Hilscher GmbH) to the INtime CIF driver. Thus, with the INtime CIF driver installed, both INtime and Windows applications can use the driver to manage devices on the fieldbus controlled by the supported Hilscher card.

See Section 6.3 for limitations imposed by the driver on simultaneous INtime and Windows application access of the driver. This “Dual Headed Driver” feature allows both fieldbus configuration and operation to occur with only the INtime CIF driver loaded.

### 1.1.3 INtime CIF Driver Topology

# INtime CIF Driver



## 1.2 Versions, Features and Limitations

### Versions:

- TenAsys INtime product version 2.20 or later

INtime Version	Supported Windows System	INtime CIF Driver Version
V2.x	Windows XP Windows XP Embedded Windows 2000 Windows NT 4.0 SP6	V1.0
V3.x	Windows Vista Windows XP Windows XP Embedded Windows 2000 SP1	V1.0 / V1.1
4.x	Windows 7 Windows Vista Windows XP Windows XP Embedded	V1.1

- Microsoft Visual Studio V6.0 SP 3 or later

### Features:

- Automatic hardware detection for ISA, PCI and CompactPCI hardware
- Device selection by board number
- All devices can be used simultaneously (ISA, PCI, CompactPCI)
- Up to four devices (ISA, PCI or CompactPCI) are possible within a system

### Limitations:

- Up to 32 INtime applications can access the driver at the same time
- Only one Windows application can have an active connection to the driver at any given time.
- When one or more INtime applications have an active connection to the driver, then the Windows application connected to the driver is restricted to "READ ONLY" calls (See Section 6.3 for a list of calls and their support from a Windows application when any INtime application is connected to the INtime CIF driver.
- Only device polling is supported

### 1.3 Supported CIF Cards, Fieldbus System and Protocols

The following tables (courtesy of Hilscher GmbH) show the currently supported CIF cards, fieldbus system and protocols.

CIF Card	Fieldbus System	Protocols
CIF30	Master / Slave	Profibus DeviceNet CANOpen Interbus ASI ControlNet ASCII Modnet 1N Modbus RTU 3964R RK512
CIF50	Master / Slave	
CIF80	Master / Slave	
CIF104	Master / Slave	

## 2 Installation

To install the INtime CIF Driver package, simply invoke the INtimeCIFInstall.msi program using the Add/Remove Applet in the Windows Control Panel (Start=>Control Panel=>Add/Remove Programs). After invoking the Applet, select "Add New Programs" and then "CD or Floppy".

Browse to the INtimeCIFInstall.exe program (either in the directory to which you downloaded the file from the TenAsys Corporation website ([www.tenasys.com](http://www.tenasys.com)), or on the INtime CIF driver installation CD), and then Select Finish to start the installation process.

The installation program insures INtime is installed on the target system and then copies the following files into directories that are part of the INtime directory structure:

File Name	Description	Directory
INtimeCIFSrv.exe	INtime CIF Driver Windows Service	<INtime Install Path>\CIF
CIFIntimeDrv.rta	INtime CIF Driver	<INtime Install Path>\CIF
INtimeCIFDriver.pdf	INtime CIF Driver Manual (this document)	<INtime Install Path>\CIF
Windows-based Driver Test Program (Msg_Dbg.exe or new Hilscher DrvTest.exe)	Windows Test Program developed by Hilscher GmbH that is used to exercise/monitor the devices on the fieldbus	<INtime Install Path>\CIF
Cif32dll.dll	NTX-based Windows DLL	<INtime Install Path>\bin Windows\system32 (if Hilscher Windows CIF Driver was found to be installed)
CIFrtlib.lib	INtime CIF Driver Interface Library	<INtime Install Path>\rt\lib
INtimeCIFTTest.rta and its source/generation MSVC files	Source and executable for an INtime CIF driver test program.	<INtime Install Path>\Projects\CIFTTests\INtimeCIFTTest. Executable ends up in the RTDebug subdirectory
WinCIFTTest.exe and its source/generation MSVC files	Source and executable for a Windows-based INtime CIF driver test program	<INtime Install Path>\Projects\CIFTTests\INtimeCIFTTest. Executable ends up in the Debug subdirectory
INtimeCIF.h Hilscher header files	Header files used to generate the INtimeCIFTTest.rta and WinCIFTTest.exe test programs	<INtime Install Path>\Projects\CIFTTests\INtimeCIFTTest\CifAPI

### 3 Fieldbus Configuration Details

A fieldbus configuration is created by the Hilscher GmbH SyCon (**S**ystem **C**onfigurator) program.

Please refer to the SyCon Manual for information on how to use this powerful tool.

NOTE: Because of the “Dual-headed” capability of the INtime CIF driver, the SyCon program can be run with the INtime driver installed. When you run it to configure your hardware and fieldbus, make sure there are no INtime applications running which have a connection to the driver. Otherwise, the write/modify portions of SyCon will fail.

#### 3.1 Fieldbus Startup Behaviour

SyCon can be used to configure the start up behaviour of a fieldbus system. The following information is provided courtesy of Hilscher GmbH.

It can be ‘*Automatic release of the communication by the device*’, which means the master starts the fieldbus system as soon as the card has finished it’s power on sequence. The fieldbus becomes active and slaves are able to driver their outputs.

The option ‘*Controlled release of the communication by the application program*’ can be used to prevent an automatic startup of the field bus system. This enables the application to be fully initialized before starting the fieldbus. In this case the application has to use the function *DevSetHostState(.., HOST\_READY,..)* to signal the master to start the fieldbus system.

SyCon settings	
‘ <i>Automatic release of the communication by the device</i> ’	The fieldbus system will start up as soon as the card has finished it’s power on sequence
‘ <i>Controlled release of the communication by the application program</i> ’ (recommended)	The start of the fieldbus system can be controlled by an application

### 3.2 Supported Handshake Modes

Handshake modes are used to control access to the I/O process data image between the PC (Host) and the CIF card (Device). The setting of the transfer mode is very important because it will influence the consistency of the process data and the fieldbus behaviour. Transfer modes are only available on master cards.

The modes are configured via SyCon which offers up to six modes, depending on the used CIF hardware and fieldbus system (Table courtesy of Hilscher GmbH).

Mode	Supported
Bus synchronous, device controlled	<b>NO</b> (Only possible on dedicated systems, because the system must respond during a bus data cycle which can be less than 350 micro seconds)
Buffered device controlled	<b>YES</b>
No consistence, uncontrolled	<b>YES</b> (Not recommended, process data which are not of the type byte can be transferred inconsistent)
Buffered host controlled	<b>YES</b> (recommended)
Bus synchronous, host controlled	<b>YES</b> (Not on all fieldbus systems and CIF cards available. System is responsible to drive the bus)
Buffered, extended host controlled	<b>NO</b>

**Note:** For more information about transfer modes, consult the 'TOOLKIT' manual section 'IO Communication with a Process Image'

## 4 Driver Functions

Refer to the Hilscher GmbH Device Driver Manual for a list of Driver Functions and their use.

## 5 Programming

The standard Hilscher programming API is used.

### 5.1 API Files

#### **CIFINtimeDrv.rta**

This is the INtime CIF driver.

#### **CIFrtlib.lib**

This is the interface library used by an INtime application to access the INtime CIF driver. It requires the standard CIF API definition file CIFUSER.H and the hardware specific definitions from RCS\_USER.H.

The following fieldbus specific header files are also required in order to specify bus-specific features:

COM_user.h	CANOpen Master
COS_user.h	CANOpen Slave
DNM_user.h	DeviceNet Master
DNS_user.h	DeviceNet Slave
DPM_user.h	PROFIBUS-DP Master
DPS_user.h	PROFIBUS-DP Slave
IBM_user.h	InterBus Master

#### **CIF32dll.lib**

This is the interface library (developed by Hilscher GmbH) used by a Windows application to access either the Windows or INtime CIF driver. The same header files apply here as listed above with the CIFrtlib.lib file.

#### **CIF32dll.dll**

This is the INtime version of the Hilscher GmbH DLL used to talk to the CIF driver. With this DLL installed, a Windows application linked to CIF32dll.lib can access the INtime CIF driver.

See Section 6.3 for the limitations for this access when an INtime application has an active connection to the CIF driver.

## 5.2 Calling Sequence

In general, the following call sequence should be followed when writing an application to communicate with the INtime CIF driver. Refer to the Hilscher GmbH Device Driver Manual for additional information.

An application needs to make the following calls, in the order specified, to obtain an active connection to the driver.

DevOpenDriver

DevInitBoard

Board and Driver Information can be obtained using the following calls:

DevGetBoardInfo

DevGetBoardInfoEx

DevGetInfo

Depending on the Startup Behavior setup via the SyCon program, the application should make the following call to initiate communications over the fieldbus:

DevSetHostState

Once communications is established on the fieldbus, the application should use one or more of the following calls to Receive and Send I/O data from/to devices on the fieldbus:

DevExchangeIO

DevExchangeIOEx

DevExchangeIOErr

DevReadSendData

Once communications is established on the fieldbus, the application should use one or more of the following calls to Send and Receive Messages to/from devices on the fieldbus:

DevPutMessage

DevGetMBXState

DevGetMessage

DevReadWriteDPMRaw

Depending on the Startup Behavior setup via the SyCon program, the application should make the following call to terminate communications over the fieldbus:

DevSetHostState

An application needs to make the following calls, in the order specified, to terminate an active connection to the driver.

DevExitBoard

DevCloseDriver

### 5.3 Example Program INtimeCIFTest

INtimeCIFTest.rta is an INtime example program which demonstrates access via the INtime CIF Driver of I/O data on a Hilscher GmbH CB-AB32-DPS I/O TestBoard. This program is provided in source form as a Developer Studio 6.0 project. Please feel free to use it as a starting point when developing your own application.

### 5.4 Example Program WinCIFTest

WinCIFTest.exe is a Windows example program which demonstrates access via the INtime CIF Driver of I/O data on a Hilscher GmbH CB-AB32-DPS I/O TestBoard. This program is provided in source form as a Developer Studio 6.0 project. Please feel free to use it as a starting point when developing your own application.

NOTE: WinCIFTest.exe and INtimeCIFTest.rta share a common set of source modules, thus demonstrating the ease of porting a Windows CIF driver access program to the INtime environment.

### 5.5 Programming hints

- 1) It takes approximately 1 millisecond for a write/read cycle to complete on the fieldbus. Therefore, set your polling times and thread access times to not attempt to overdrive the bus.
- 2) Send and Receive Data from Multiple CIF devices on the fieldbus is configured by SyCon to be in a contiguous array of data. If multiple threads are going to access the data, have each thread only look at the fields in the array it is going to manage. Read-only requests can be made without a timeout value, so the latest Receive Data can be obtained at any time. Send Data requires a bus cycle to insure the data is sent out on the fieldbus. The call will not return until the bus cycle completes.
- 3) Responses to messages sent via the Mailbox functions of the driver can return in an out of order fashion. Multiple threads making these calls could end up receiving a message destined for another thread. The messages themselves have transaction information built into their headers. It is recommended that a single thread manage all messaging to/from the driver so that it can route the received messages to the appropriate threads.

## 6 Additional Information

### 6.1 Hilscher GmbH Reference Manuals

Please use the listed Hilscher GmbH manuals if you are searching for additional information on use of the driver.

Manual name	Content
CIF Device Driver	General driver description, functions and API
Protocol Manual	General protocol information and definition
Protocol Interface Manual	Protocol specific definitions and functions

### 6.2 Multiple Hilscher CIF Card Support

A separate instance of the INtime CIF driver is loaded for each board in the system. Upon loading, the driver queries both the PCI and ISA buses to find all the CIF/COM boards in the system in the following order and assigns them sequential instance numbers starting with 0 (up to 4):

First Hilscher PCI device is Instance 0

Second Hilscher PCI device is Instance 1

etc.

After all the Hilscher devices on the PCI bus are enumerated and assigned an instance number, then the ISA bus (i.e. IPC Memory between 0C0000H and 0EFFFFH) is examined to find any ISA Hilscher cards present there. These are assigned instance numbers in the order found from lowest to highest memory address used.

So in a system with both an ISA and a PCI Hilscher board, the driver will discover both boards and assign them instance numbers in the following order:

Hilscher PCI device is Instance 0

Hilscher ISA device is Instance 1

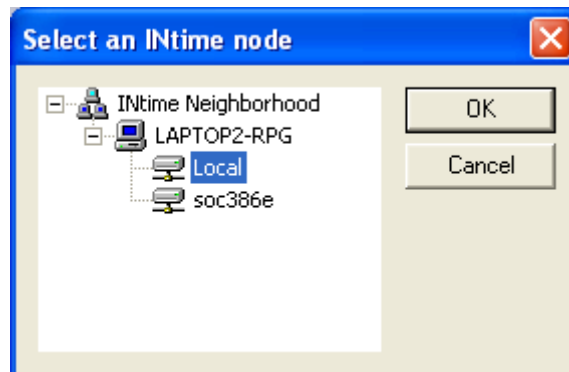
The first INtime CIF driver is loaded by starting the INtimeCIFService Windows Service using the Services Applet in Control Panel. In this 2 board system example, the driver will find the Hilscher PCI device (Instance 0) and assign it as Board 0.

To load a driver for the ISA board which is Instance 1 in the driver's view of Hilscher devices in the above system, a second instance of the driver (<INtime Install Path>\CIF\INtimeCIFDrv.rta) is loaded using the INtime Application Loader (ldrta.exe) with the following parameters:

```
<INtime Install Path>\bin\ldrta <INtime Install Path>\CIF\INtimeCIFDrv.rta -args "inst=1 board=1".
```

### 6.3 INtime Node Browser

The INtime RTOS supports the concept of a “Remote INtime Node”. This is an instance of the INtime kernel running on a separate, remote CPU without Windows. The interface to such a Remote Node is transparent to the user other than selecting the node with the INtime Node Browser as shown below:



With the INtime CIF Driver installed, whenever a Windows Application calls **DevOpenDriver()**, the INtime Cif32dll.dll will open the INtime Node Browser to allow you to select the node on which the Hilscher CIF board is installed. All active INtime nodes will show up, both local (default name is Local) and remote (in the case above, soc386e). Select the node by highlighting the desired node, and then click OK. All subsequent driver calls will be routed to the INtime CIF Driver on the node selected.

### 6.4 Limitations to CIF Driver Access from a Windows Application

The INtime CIF driver supports both INtime and Windows applications using it to communicate over the various industrial buses supported by the Hilscher line of CIF/COM communications products. However, INtime applications have precedence, and, if running (i.e. have made a *DevOpenDriver()* call to the device), will relegate Windows application calls to READ\_ONLY status.

Only one Windows application at a time can access the INtime driver. Once one Windows application has issued a *DevOpenDriver()* call, all other Windows applications will get a -10 DRV\_DEV\_DPM\_ACCESS\_ERROR when they attempt to open the driver. Only when the Windows application calls *DevCloseDriver()* will another Windows application be able to access the driver.

The following table lists the Hilscher/INtime CIF driver API, and the results if a Windows application makes the call while one or more INtime applications is running(have made a *DevOpenDriver()* call to the device):

## CALL TABLE

Function	Action if No INtime application active (i.e. has called DevOpenDriver)	Action if INtime application active
DevOpenDriver	Links an application to the device driver	Links an application to the device driver (One Windows application only. Others get a -10 DRV_DEV_DPM_ACCESS_ERROR)
DevCloseDriver	Closes a Link to the driver	Closes a Link to the driver
DevInitBoard	Links an application to the board	Links an application to the board
DevExitBoard	Closes a link to the board	Closes a link to the board
DevReset *	Resets the board	Returns error -25 DRV_DEV_FUNCTION_FAILED
DevSetHostState *	Sets/Clears information bit for Host is Running	Returns error -25 DRV_DEV_FUNCTION_FAILED if call would change the state of the Host is Running bit
DevTriggerWatchDog *	Serves the watchdog function of the board	Returns error -25 DRV_DEV_FUNCTION_FAILED
DevPutMessage *	Transfers a message to the board	Returns error -25 DRV_DEV_FUNCTION_FAILED
DevGetMessage *	Reads a message from the board	Returns error -25 DRV_DEV_FUNCTION_FAILED
DevGetMBXState	Read the actual mailbox state	Read the actual mailbox state

## CALL TABLE (Continued)

Function	Action if No INtime application active (i.e. has called DevOpenDriver)	Action if INtime application active
DevExchangeIO *	Send/Receive IO data to/from the board	Receive IO data to/from the board. If SendData SendSize is non-zero, returns error -25 DRV_DEV_FUNCTION_FAILED along with received IO data
DevExchangeIOEx *	Send/Receive IO data to/from the COM module	Receive IO data to/from the board. If SendData SendSize is non-zero, returns error -25 DRV_DEV_FUNCTION_FAILED along with received IO
DevExchangeIOErr *	Send/Receive IO data to/from the board including state information	Receive IO data to/from the board. If SendData SendSize is non-zero, returns error -25 DRV_DEV_FUNCTION_FAILED along with received IO data, including state information
DevReadSendData	Read back IO data from the send area	Read back IO data from the send area
DevPutTaskParameter *	Writes the parameters for a communications task	Returns error -25 DRV_DEV_FUNCTION_FAILED
DevGetTaskParameter	Reads the parameters for a communications task	Reads the parameters for a communications task
DevGetTaskState	Reads all task states from the board	Reads all task states from the board
DevGetBoardInfo	Read global board information	Read global board information
DevGetInfo	Reads the various information from the board	Reads the various information from the board
DevReadWriteDPMRaw *	Read/write from/to the last Kbytes of DPM data on the board	Reads from the last Kbytes of DPM data on the board. If usMode is PARAMETER_WRITE, returns error -25 DRV_DEV_FUNCTION_FAILED
DevDownload	Firmware/Configuration download	Returns error -25 DRV_DEV_FUNCTION_FAILED

## 6.5 Windows Handling of Error -25 DRV\_DEV\_FUNCTION\_FAILED

After the default installation of the INtime CIF driver, the CIF32dll.dll will respond to the API calls marked above with an asterisk (\*) by popping up a Windows Message Box stating that the Call or a portion of it was rejected. Once the Message Box is exited via a Click on the OK button, an error code of 0 is returned to the application.

Calls not marked with an asterisk (\*) also pop up a Windows Message Box stating that the Call or a portion of it was rejected.

Once the Message Box is exited via a Click on the OK button, the -25 error code is returned to the application.

You can change the DLL behavior to not pop up a message box, and to always return the -25 error code, by changing an entry in the Windows registry using the regedit Windows applet as follows:

Start=>Run (type in regedit) OK

In the User Interface of regedit, change the following value from 1 (default) to 0:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\INtimeCIFService\  
ErrorMode

## 6.6 Use of Windows-based Driver Test Program to Monitor Fieldbus systems driven by INtime Applications

The INtime CIF Driver package includes a Windows program from Hilscher GmbH called **Msg\_Dbg.exe**. This program is provided in executable form as a tool to monitor fieldbus activity instigated by an INtime application using the INtime CIF driver.

To use the tool, do the following:

Invoke Msg\_Dbg.exe

Select Board

Select Data Transfer -> ExchangeIO

This setting monitors input data and also monitors the state of the bus (Data Exchange or not). Input data will be updated at the refresh rate of Msg\_Dbg (a few times a second)

At the same time, Select Driver Function ->ReadSendData

Select the amount of send data (sent by the INtime application) to display. Click the Read Send Button when you want to update the data

Start the INtime Application and begin monitoring

## **6.7 Use of Hilscher GmbH SyCon Program with the INtime CIF Driver**

SyCon operates normally with the INtime CIF Driver **WHEN NO INtime Application** has an open connection to the driver; so use it to configure the system before starting any INtime application that communicates with the driver.

**Note: DO NOT USE SyCon when an INtime Application is accessing the driver.**

## 7 Error Numbers

### 7.1 List of Error Numbers (Courtesy of Hilscher GmbH)

The column *Hint* shows if there is additional error information available. If 'Yes' then see section *Additional Error Information*, which is the next section.

Value	Parameter	Description	Hint
0	DRV_NO_ERROR	no error	
-1	DRV_BOARD_NOT_INITIALIZED	DRIVER Board not initialized	yes
-2	DRV_INIT_STATE_ERROR	DRIVER Error in internal init state	
-3	DRV_READ_STATE_ERROR	DRIVER Error in internal read state	
-4	DRV_CMD_ACTIVE	DRIVER Command on this channel is active	
-5	DRV_PARAMETER_UNKNOWN	DRIVER Unknown parameter in function occurred	
-6	DRV_WRONG_DRIVER_VERSION	DRIVER Version is incompatible with DLL	
-7	DRV_PCI_SET_CONFIG_MODE	DRIVER Error during PCI set run mode	
-8	DRV_PCI_READ_DPM_LENGTH	DRIVER Could not read PCI dual port memory length	
-9	DRV_PCI_SET_RUN_MODE	DRIVER Error during PCI set run mode	
-10	DRV_DEV_DPM_ACCESS_ERROR	DEVICE Dual port ram not accessible (board not found)	yes
-11	DRV_DEV_NOT_READY	DEVICE Not ready (ready flag failed)	yes
-12	DRV_DEV_NOT_RUNNING	DEVICE Not running (running flag failed)	yes
-13	DRV_DEV_WATCHDOG_FAILED	DEVICE Watchdog test failed	yes
-14	DRV_DEV_OS_VERSION_ERROR	DEVICE Signals wrong OS version	yes
-15	DRV_DEV_SYSERR	DEVICE Error in dual port flags	
-16	DRV_DEV_MAILBOX_FULL	DEVICE Send mailbox is full	
-17	DRV_DEV_PUT_TIMEOUT	DEVICE PutMessage timeout	yes
-18	DRV_DEV_GET_TIMEOUT	DEVICE GetMessage timeout	yes
-19	DRV_DEV_GET_NO_MESSAGE	DEVICE No message available	
-20	DRV_DEV_RESET_TIMEOUT	DEVICE RESET command timeout	yes
-21	DRV_DEV_NO_COM_FLAG	DEVICE COM-flag not set	yes
-22	DRV_DEV_EXCHANGE_FAILED	DEVICE IO data exchange failed	
-23	DRV_DEV_EXCHANGE_TIMEOUT	DEVICE IO data exchange timeout	yes
-24	DRV_DEV_COM_MODE_UNKNOWN	DEVICE IO data mode unknown	
-25	DRV_DEV_FUNCTION_FAILED	DEVICE Function call failed/rejected	yes
-26	DRV_DEV_DPMSIZE_MISMATCH	DEVICE DPM size differs from configuration	
-27	DRV_DEV_STATE_MODE_UNKNOWN	DEVICE State mode unknown	
-30	DRV_USR_OPEN_ERROR	USER Driver not opened (device driver not loaded)	yes
-31	DRV_USR_INIT_DRV_ERROR	USER Can't connect with device	
-32	DRV_USR_NOT_INITIALIZED	USER Board not initialized (DevInitBoard not called)	

-33	DRV_USR_COMM_ERR	USER IOCTL function failed	
-34	DRV_USR_DEV_NUMBER_INVALID	USER Parameter DeviceNumber invalid	
-35	DRV_USR_INFO_AREA_INVALID	USER Parameter InfoArea unknown	
-36	DRV_USR_NUMBER_INVALID	USER Parameter Number invalid	
-37	DRV_USR_MODE_INVALID	USER Parameter Mode invalid	
-38	DRV_USR_MSG_BUF_NULL_PTR	USER NULL pointer assignment	
-39	DRV_USR_MSG_BUF_TOO_SHORT	USER Message buffer too short	
-40	DRV_USR_SIZE_INVALID	USER Parameter Size invalid	
-42	DRV_USR_SIZE_ZERO	USER Parameter Size with zero length	
-43	DRV_USR_SIZE_TOO_LONG	USER Parameter Size too long	
-44	DRV_USR_DEV_PTR_NULL	USER Device address null pointer	
-45	DRV_USR_BUF_PTR_NULL	USER Pointer to buffer is a null pointer	
-46	DRV_USR_SENDSIZE_TOO_LONG	USER Parameter SendSize too long	
-47	DRV_USR_RECVSIZE_TOO_LONG	USER Parameter ReceiveSize too long	
-48	DRV_USR_SENDBUF_PTR_NULL	USER Pointer to send buffer is a null pointer	
-49	DRV_USR_RECVBUF_PTR_NULL	USER Pointer to receive buffer is a null pointer	
-50	DRV_DMA_TIMEOUT_CH4	DMA read IO timeout	
-51	DRV_DMA_TIMEOUT_CH5	DMA write IO timeout	
-52	DRV_DMA_TIMEOUT_CH6	DMA PCI transfer timeout	
-53	DRV_DMA_TIMEOUT_CH7	DMA download timeout	
-54	DRV_DMA_INSUFF_RES_MEM	DMA Memory allocation error	
-70	DRV_ERR_ERROR	DRIVER General error	
-71	DRV_DMA_ERROR	DRIVER General DMA error	
-72	DRV_BATT_ERROR	DRIVER Battery error	
-73	DRV_PWF_ERROR	DRIVER Power failed error	
-80	DRV_USR_DRIVER_UNKNOWN	USER driver unknown	
-81	DRV_USR_DEVICE_NAME_INVALID	USER device name invalid	
-82	DRV_USR_DEVICE_NAME_UNKNOWN	USER device name unknown	
-83	DRV_USR_DEVICE_FUNC_NOTIMPL	USER device function not implemented	
-100	DRV_USR_FILE_OPEN_FAILED	USER file not opened	
-101	DRV_USR_FILE_SIZE_ZERO	USER file size zero	
-102	DRV_USR_FILE_NO_MEMORY	USER not enough memory to load file	
-103	DRV_USR_FILE_READ_FAILED	USER file read failed	
-104	DRV_USR_INVALID_FILETYPE	USER file type invalid	
-105	DRV_USR_FILENAME_INVALID	USER file name not valid	
-110	DRV_FW_FILE_OPEN_FAILED	USER firmware file not opened	
-111	DRV_FW_FILE_SIZE_ZERO	USER firmware file size zero	
-112	DRV_FW_FILE_NO_MEMORY	USER not enough memory to load firmware file	
-113	DRV_FW_FILE_READ_FAILED	USER firmware file read failed	
-114	DRV_FW_INVALID_FILETYPE	USER firmware file type invalid	
-115	DRV_FW_FILENAME_INVALID	USER firmware file name not valid	
-116	DRV_FW_DOWNLOAD_ERROR	USER firmware file download error	

-117	DRV_FW_FILENAME_NOT_FOUND	USER firmware file not found in the internal table	
-118	DRV_FW_BOOTLOADER_ACTIVE	USER firmware file BOOTLOADER active	
-119	DRV_FW_NO_FILE_PATH	USER firmware file not file path	
-120	DRV_CF_FILE_OPEN_FAILED	USER configuration file not opened	
-121	DRV_CF_FILE_SIZE_ZERO	USER configuration file size zero	
-122	DRV_CF_FILE_NO_MEMORY	USER not enough memory to load configuration file	
-123	DRV_CF_FILE_READ_FAILED	USER configuration file read failed	
-124	DRV_CF_INVALID_FILETYPE	USER configuration file type invalid	
-125	DRV_CF_FILENAME_INVALID	USER configuration file name not valid	
-126	DRV_CF_DOWNLOAD_ERROR	USER configuration file download error	
-127	DRV_CF_FILE_NO_SEGMENT	USER no flash segment in the configuration file	
-128	DRV_CF_DIFFERS_FROM_DBM	USER configuration file differs from database	
-131	DRV_DBM_SIZE_ZERO	USER database size zero	
-132	DRV_DBM_NO_MEMORY	USER not enough memory to upload database	
-133	DRV_DBM_READ_FAILED	USER database read failed	
-136	DRV_DBM_NO_FLASH_SEGMENT	USER database segment unknown	
-150	DEV_CF_INVALID_DESCRIPTOR_VERSION	CONFIG version of the descriptor table invalid	
-151	DEV_CF_INVALID_INPUT_OFFSET	CONFIG input offset is invalid	
-152	DEV_CF_NO_INPUT_SIZE	CONFIG input size is 0	
-153	DEV_CF_MISMATCH_INPUT_SIZE	CONFIG input size does not match configuration	
-154	DEV_CF_INVALID_OUTPUT_OFFSET	CONFIG invalid output offset	
-155	DEV_CF_NO_OUTPUT_SIZE	CONFIG output size is 0	
-156	DEV_CF_MISMATCH_OUTPUT_SIZE	CONFIG output size does not match configuration	
-157	DEV_CF_STN_NOT_CONFIGURED	CONFIG Station not configured	
-158	DEV_CF_CANNOT_GET_STN_CONFIG	CONFIG cannot get the Station configuration	
-159	DEV_CF_MODULE_DEF_MISSING	CONFIG Module definition is missing	
-160	DEV_CF_MISMATCH_EMPTY_SLOT	CONFIG empty slot mismatch	
-161	DEV_CF_MISMATCH_INPUT_OFFSET	CONFIG input offset mismatch	
-162	DEV_CF_MISMATCH_OUTPUT_OFFSET	CONFIG output offset mismatch	
-163	DEV_CF_MISMATCH_DATA_TYPE	CONFIG data type mismatch	
-164	DEV_CF_MODULE_DEF_MISSING_NO_SLOT_IDX	CONFIG Module definition is missing,(no Slot/Idx)	
>=1000	RCS_ERROR	Board operation system errors will be passed with this offset (e.g. error 1234 means RCS error 234). Only if a ready fault occurred during board initialization.	

## 7.2 Additional Error Information

This section contains more information (courtesy of Hilscher GmbH) about possible reasons for certain error numbers.

### **Error: -1**

The communication board is not initialized by the driver.

No or wrong configuration found for the given board.

- Check the driver configuration
- Driver function used without calling DevOpenDriver() first

### **Error: -6**

The device driver version does not corresponds to the driver DLL version. From version V1.200 the internal command structure between DLL and driver has changed.

- Make sure to use the same version of the device driver and the driver DLL

### **Error: -10**

Dual ported RAM (DPM) not accessible / no hardware found.

This error occurs, when the driver is not able to read or write to the DPM

- Check the BIOS setting of the PC
- Memory address conflict with other PC components, try another memory address
- Check the driver configuration for this board
- Check the jumper setting of the board

### **Error: -11**

Board is not ready.

This is a general error, the board has a hardware malfunction.

### **Error: -12**

At least one task is not initialized. The board is ready but not all tasks are running.

- No data base is loaded into the device
- Wrong parameter that causes that a task can't initialize. Use ComPro menu *Online-task-version*.

### **Error: -14**

No license code found on the communication board.

- Device has no license for the used operating system or customer software.
- No firmware or no data base on the device loaded.

### **Error: -17**

No message could be send during the timeout period given in the DevPutMessage () function.

- Using device interrupts

Wrong or no interrupt selected. Check interrupt on the device and in driver registration.

They have to be the same!. Interrupt already used by an other PC component.

- Device internal segment buffer full

PutMessage () function not possible, because all segments on the device are in use.

This error occurs, when only PutMessage () is used but not GetMessage () .

- HOST flag not set for the device

No messages are taken by the device. Use DevSetHostState() to signal a board an application is available.

### **Error: -18**

No message received during the timeout period given in the

DevGetMessage () function.

- Using device interrupts

Wrong or no interrupt selected. Check interrupt on the device and in driver registration. They have to be the same!. Interrupt already used by an other PC component.  
- The used protocol on the device needs longer than the timeout period given in the `DevGetMessage ()` function

**Error: -20**

The device needs longer than the timeout period given in the `DevReset ()` function  
- Using device interrupts

This error occurs when for example interrupt 9 is set in the driver registration but no or a wrong interrupt is jumpered on the device (=device in pollmode).

Interrupt already used by an other PC component.

- The timeout period can differ between fieldbus protocols

**Error: -21**

The device can not reach communication state.

- Device not connected to the fieldbus

- No station found on the fieldbus

- Wrong configuration on the device

**Error: -23**

The device needs longer than the timeout period given in the `DevExchangeIO ()` function.

- Using device interrupts

Wrong or no interrupt selected. Check interrupt on the device and in driver registration.

They have to be the same!. Interrupt already used by an other PC component.

**Error: -25**

A Windows application has attempted to change an Output Parameter in one of the supported devices on the fieldbus while an INtime application controlling the driver is active. For example, attempting to send output data via `Msg_Dbg.exe` while an INtime application has an open connection to the driver will cause this error.

– Only use Windows applications like `Sycon` or `Msg_Dbg` to modify Output data when no INtime applications have an open connection to the driver.

**Error: -30**

The device driver could not be opened.

- Device driver not installed

- Wrong parameters in the driver configuration

If the driver finds invalid parameters for a communication board and no other boards with valid parameters are available, the driver will not be loaded.

**Error: -33**

A driver function could not be called. This is an internal error between the device driver and the DLL.

- Make sure to use a device driver and a DLL with the same version.

- An incompatible old driver DLL is used.