



**Documentation for OEMs:
CoDeSys SP RTE
Realtime Runtime System for Windows NT/2000/XP**

Document Version 2.2

CONTENT

1. OVERVIEW	4
1.1 What does "real time extension" mean?	4
1.2 The System in Detail	5
2 OPERATION VIA THE SERVICE ICON	6
2.1 The System Menu	7
2.1.1 Diagnostic	7
2.1.2 Startup	9
2.1.3 Config	10
2.1.4 Ext. Config	11
2.1.5 Access	12
2.1.6 IO Drivers	13
2.2 License ...	14
3 THE 3S STANDARD IO DRIVERS	15
3.1 IO driver RTIOdrvHilscherDPM	15
3.1.1 The SysLibDpV1.lib	16
3.2 IO driver RTIOdrvFC310x.sys	17
3.2.1 The library SysLibFCDPV1.sys	17
3.2.2 The library FC_SlaveHandling.lib	18
3.2.3 The FC310x card in slave mode	18
3.3 IO driver RTIOdrvCP5613.sys	19
3.4 IO driver RTIOdrvDAMP	19
3.5 IO driver RTIOdrvApplicom	20
3.6 IO driver RTIOdrvIBS (IBS = Interbus, ibpcimp.sys)	20
3.7 IO Drivers RTIOdrvSJA and RTIOwdmPCAN	20
3.8 IO Drivers RTIOdrvAutomata and RTIOwdmAutomata	20
3.9 IO drivers RTIOdrvHMS and RTIOwdmHMS	21
3.10 IO driver RTIOdrvAPIC	21
3.11 IO Driver RTIOwdmCANAutomata	22
3.12 IO Driver RTIOwdmSofting	22
4 COMMUNICATION	23
4.1 Shared Memory Driver	23
4.2 TCP/IP Level2 Route Driver	23
5 SYSTEM DIAGNOSIS	24
5.1 Bus diagnosis of the IO drivers	24
6 THE TASK CONFIGURATION TOGETHER WITH CODESYS V2.3	26

6.1	General	26
6.2	Task specific watchdog	26
6.3	Microseconds as time base	26
6.4	Freewheeling task	26
6.5	System events	27
7	SYSTEM LIBRARIES	28
7.1	DllCall.lib together with SysLibSystemCall.lib	28
7.2	SysLibCallback.lib	28
7.3	SysLibCom.lib	28
7.4	SysLibFile.lib	29
7.5	SysLibPorts	29
7.6	SysLibTime	29
7.7	SysLibSockets	30
7.8	SysLibShm.lib	30
7.9	SysLibPciCards.lib	30
8	BEHAVIOUR IN CASE OF RUNTIMEERRORS	31
9	APPENDIX	32
9.1	The Registry Entries used by the Runtime System	32
	CHANGE HISTORY	35

1. Overview

In this document it is assumed that the reader is familiar with the fundamental behaviour and functions of a CoDeSys runtime system. Therefore this brief manual only deals with the specific features of the runtime system for a real time extension of Windows NT.

The real time system is started by executing the command 'Start CoDeSys SP Windows NT for Realtime' in the CoDeSys menu in the Windows Start menu. When started the following service icon appears in the Windows task bar:

1.1 What does "real time extension" mean?

A real time system is characterized by predictable (deterministic) time behaviour. If for example a real time system is given the task of executing specific routines (through configuration) within a predetermined time pattern this will happen within predetermined temporal ranges of tolerance. If this is not the case it is regarded as a failure of the whole system.

If this is applied to a controller such as CoDeSys this means that a task is called within predetermined (known) ranges of tolerance.

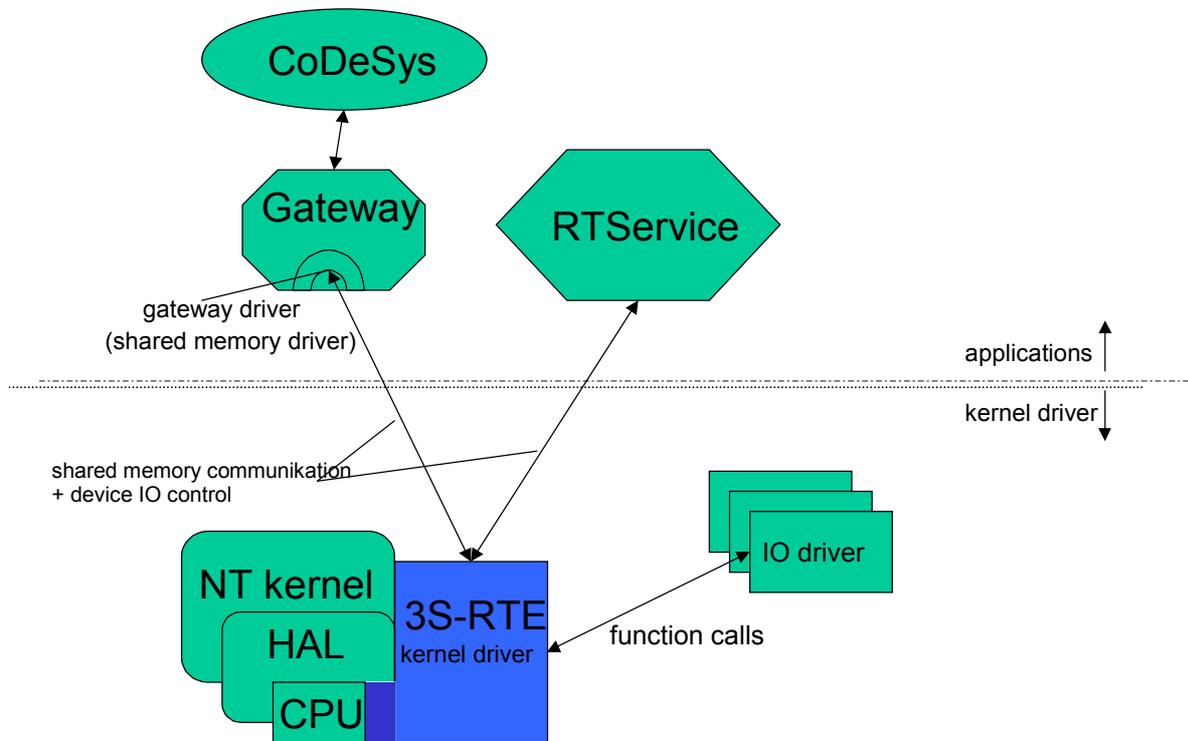
Figures drawn from past experience have shown that these ranges are not adhered to by Windows NT, which means that NT is not a genuine realtime operating system.

Yet the hardware in a PC is made in such a way that it is possible to create a reliable task scheduling by means of software.

The real time extension of Windows NT is an NT driver which installs an Interrupt-Service-Routine which is called-up cyclically by the timertick of the PC's hardware.

This routine now takes over the job of calling-up defined tasks from CoDeSys and/or of authorizing the continued or renewed execution of the operating system.

1.2 The System in Detail



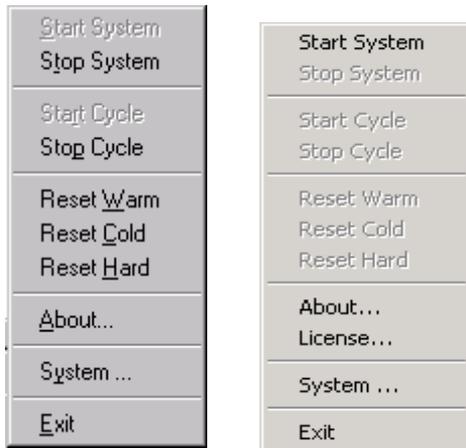
The kernel of the realtime extension (short RTE) consists of two parts: A system service, called RTService.exe and a kernelmode-driver (3SRTE.sys). In standard version the PC's timertick is used to generate 2 hardware interrupts per millisecond, and so call the scheduler of the RTE. The scheduler uses every second tick to call his own tasks and the others to return to the interrupted OS. So all PLC-tasks are interrupted once per millisecond by a time slice of OS-execution. The percentage of time for PLC-tasks is configurable (see chapter 6).

The mentioned IO-drivers are added to a list and have to offer a special interface which is defined by the IO-DriverToolkit.

The whole system does not change the operating system's kernel at runtime or installation.

2 Operation via the Service Icon

See the user menu (right mouse button) of the RTE Icon in the task bar:



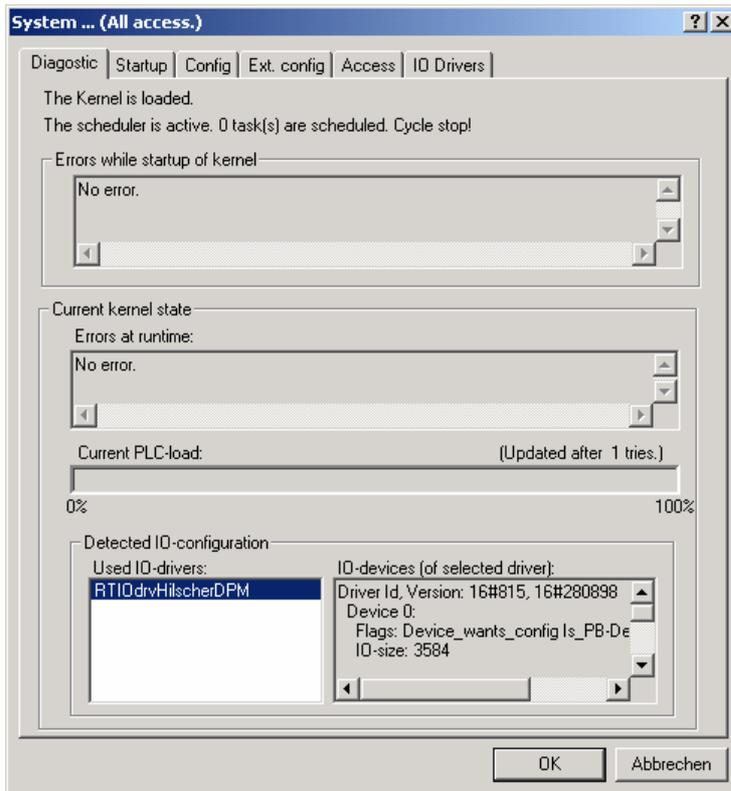
The commands which are available, depending on the current operational status, are shown black-colored (enable).

The following individual operations can be conducted:

- **Start System** loads and activates the driver. Only then is it possible to work on real time tasks and to communicate with the programming system.
- **Stop System** stops the driver and removes it from the PC's memory.
- **Start Cycle / Stop Cycle** start and stop the cyclical execution of IEC tasks. This operation corresponds with the menu item "Start"/"Stop" in the online menu of CoDeSys.
- Next are the three types of **Reset** options which also correspond with the entries in the CoDeSys menu.
- **About** shows the splash screen of the RTE which apart from the icon also includes information on the version and the copyright.
- **System** shows a dialogue with system settings. Two settings can be altered here: If there is a tick behind the option "Start when booting" the service will be started automatically by NT on the next start of the system. This happens completely regardless of the fact whether a user logs in or not. With the option "Automatic PLC Start" the user can decide whether to start the driver automatically from the service or not.
- **Exit** not only stops the driver but also the service after the driver has been unloaded. The service can be restarted with "Start – Settings– System Control – Services" or by the entry in „Start – Programs“ under CoDeSys. It is possible to start the driver in "System Control – Devices", yet this is not very sensible because it deactivates the driver. The RTE service activates it when starting.
- **License** is only available, if you own a newer version, where the hardlock (dongle) is replaced by a software license.

2.1 The System Menu

Select the item **System** in the user menu to get the following dialog:



In the following the particular sub-windows of this dialog will be described.

You have to close the dialog with OK to confirm the settings. They will be of effect as soon as the PLC gets restarted.

2.1.1 Diagnostic

The first two lines describe the current status of the system.

Errors while startup of kernel

Here the possibly occurred initialization errors are reported. Possible messages:

- "No error." : System started without errors
- "No memory available for CodeArea0." : No CodeArea0 could be created. The system RAM is too small to allocate the desired memory space. → Reduce the size of the configured code memory space, see below.
- "No memory available for CodeArea1." : See above, CodeArea0.
- "No memory available for DataArea.": See above, CodeArea0. → Reduce the size of the configured data memory space
- "No memory available for retainedata." : See above, CodeArea0. → Reduce the size of the configured retain memory space.
- "Communication Interface could not initialize." : RAM too small for communication data. → The only thing you can do is to provide additional memory devices.
- "Checksumerror when loading bootproject." : The boot project "Default.prg" is corrupt. → Re-create boot project.
- "Bootproject too big." : The boot project was generated for a configuration which provides more code memory space. → Re-create boot project.

- "Datasize of bootproject too big." : See "Bootproject too big".
- "Could not relocate bootproject." : The boot project might be corrupt.
- "Error reading bootprojectfile." : The boot project might be corrupt or currently opened.
- "Error reading taskconfig from bootproject." : The boot project might be corrupt.
- "Error reading IOconfig from bootproject." : The boot project might be corrupt.
- "No IO-driver found but specified." : A configured IO driver could not be found or could not get loaded by the system. -> Install the driver.
- „An IO-driver failed in his initfunction." : An IO driver has detected an error in the installation routine, although it was already loaded. → Check the IO configuration, where required contact the manufacturer of the IO driver. (See below a description of the 3S IO drivers.)
- „An IO-driver must handle at least 1 device." : An IO driver has reported that it could not find a device to work with. → Check IO configuration.
- „The PC-timer could not be calibrated." : The PC has not sent periodic interrupts on the expected address. May be that it is not an AT-compatible PC. → Contact 3S or redo the setup.
- „The PC-timertick is inexact." : If the PLC at starting detects that there are no matching values in more than 10 measurements, then the system will not be started. The PC has no exact time base.

Errors at runtime

Here the errors are displayed which occurred during processing the PLC program.

The errors which might be displayed, correspond to the runtime errors which can be reported by each runtime programmable with CoDeSys.:

- Watchdog: A task has exceeded its cycle time by a bigger time span than defined in the watchdog configuration. -> Might be a programming bug, otherwise change configuration.
- Field bus error : Error during configuration of an IO card.
- IO-update error: Error during reading or writing of inputs or outputs. Example: You tried to write to an address or to read an address, for which no driver is available. Only possible, if in the target settings the address check option is deactivated .
- Illegal instruction: The processor has found invalid code. Only possible, if a file is erroneous or if parts of the program have been overwritten by data.
- Access violation: Error at memory access: the application has tried to access an invalid memory address. -> Probably caused by a program bug -> fix it.
- The FPU errors „Division by zero“, „Inexact result“, „Invalid operation“, „Denormal operand“ and „Overflow“ get displayed. All these errors normally are programming bugs.

All these runtime error messages (besides IO-update error) can be localized in CoDeSys in online mode with the aid of the „Show Call Stack“ function.

Current PLC load

Here the current load of the PLC is shown: If the load gets to high or if baseless watchdog errors occur, the time slice of the PLC can be extended. See below: „Config“. The load always refers to the total processing time of the PLC, not to the total computer performance.

Detected IO Configuration

Here the window on the left side shows the configured drivers. The window on the right side will show the devices which are supported by the driver which is currently marked on the left side. For each driver a version identification is displayed and a list of the devices, consecutively numbered (starting with 0).

2.1.2 Startup

Configure here the startup behaviour of the PLC.

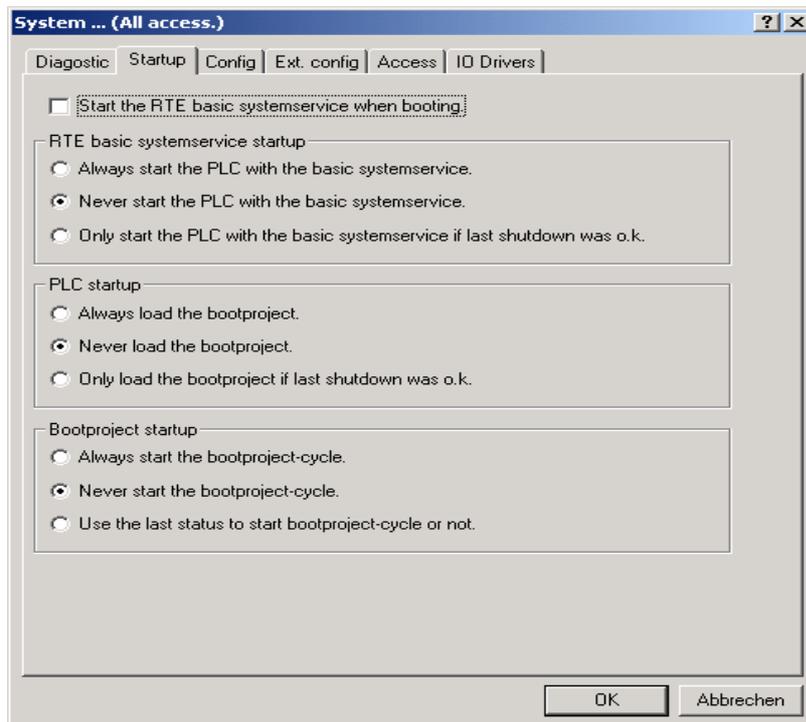
The options:

Start the RTE basic systemservice when booting: The service will be set to "Automatic start" and due to this will be started by the system during boot-up.

RTE basic systemservice startup: Startup behaviour of the system service. Define here whether the service should start the PLC in any case, never, or only if the PLC has been terminated correctly before.

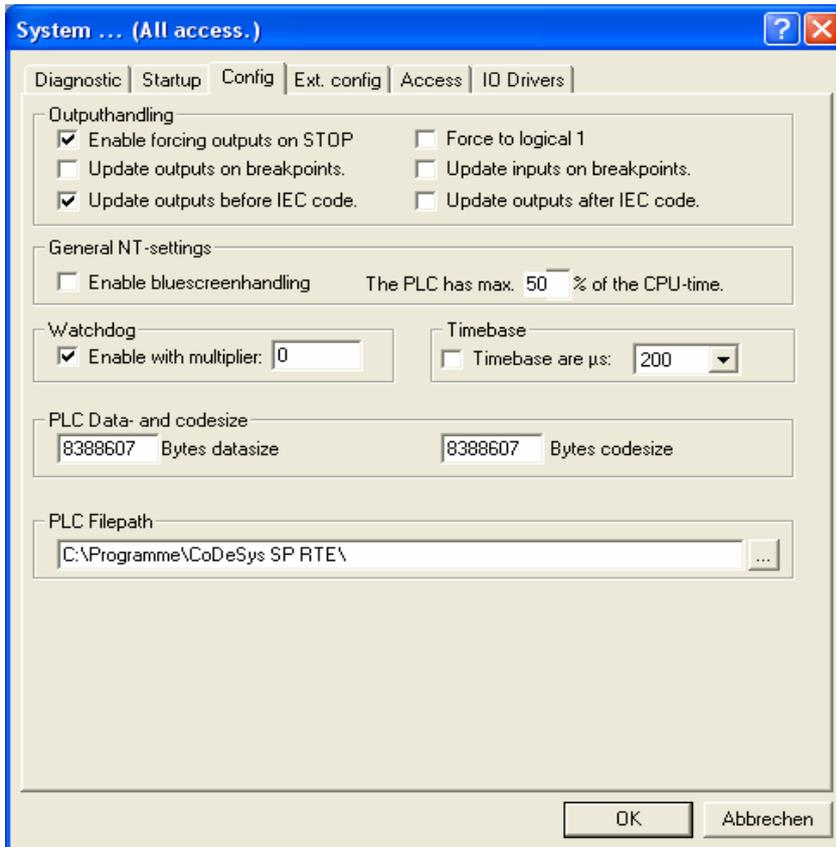
PLC startup: Startup behaviour of the PLC. Define here, whether the PLC should load the boot project in any case, never, or only if the PLC has been terminated correctly before.

Bootproject startup: Startup behaviour of the boot project. Define here, whether the PLC should start the boot project in any case, never, or only if the PLC has been terminated correctly before.



2.1.3 Config

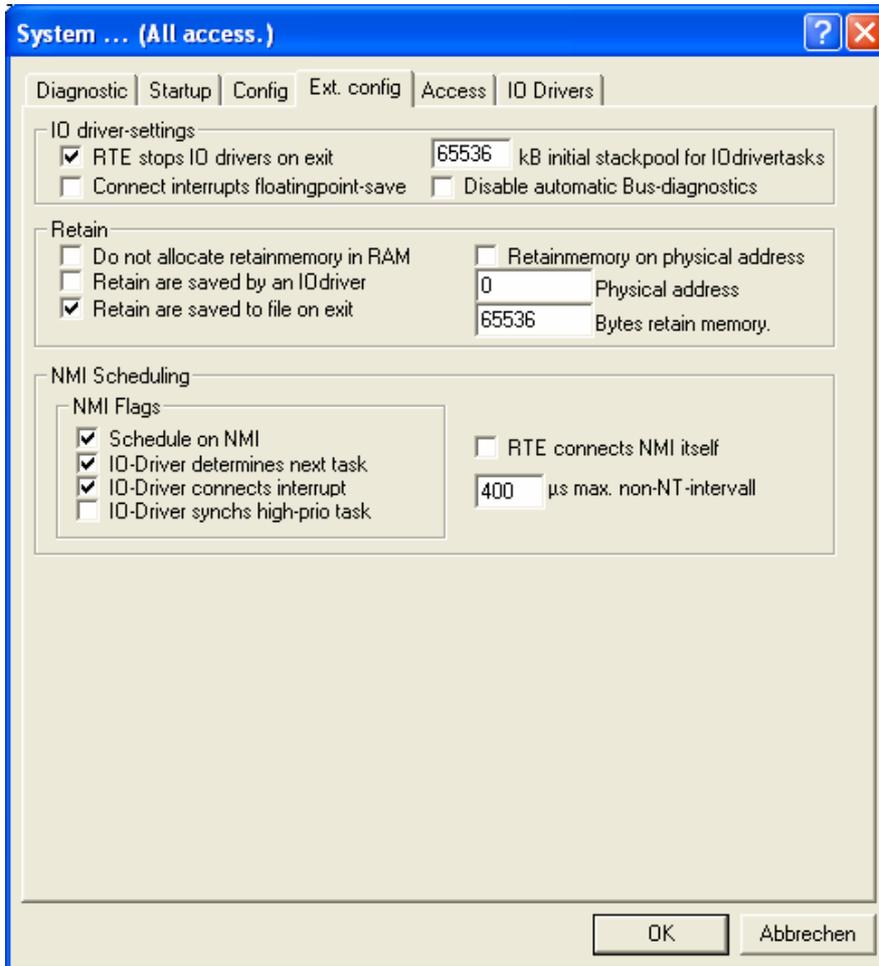
Configuration of some basic settings:



- **Outpuhandling:** If option **Enable forcing outputs on STOP** is activated, then at cycle stop the outputs will be set to 0 once. (Exception: option **Force to logical 1** is also activated; then the outputs are set to 1.) With the options **Update inputs/outputs on breakpoints** one can decide if IOs are updated when debugging - including necessary buscycles - even if the update-task is halted on a breakpoint. By default IOs are updated before the IEC-code is executed. With the use of the options **Update outputs before/after IEC code** this behaviour can be changed. This might be useful, if the task which is defined as update-task, is slow or triggered by event.
- **General NT-settings:** Set here two parameters, which also can affect the operating system: If „Enable bluescreenhandling“ is activated, then the operating system will not be able to show a „bluescreen“ in case of a severe system error (system stop, displaying some error messages), but the system will be "frozen" (for the user), while the SPS keeps running. Of course in this case all functions, which need a running operating system (like file or network access operations), cannot be executed any longer.
Set also the percentage of the CPU-time which will max. be available for the PLC.
- **Watchdog :** Here you can activate/deactivate the time control for the tasks and you can assign it with a multiplicator. The multiplicator defines how many multiples of the configured cycle time will be tolerated in case of a time overrun, before the watchdog will produce an error.
- **Timebase:** If needed the time base (default setting 1ms) can be modified. By that a higher resolution can be reached and, for CoDeSys version 2.3 and higher, also cycle times < 1ms are possible. (In the versions lower than 2.3.1 also cycle times < 1ms have been possible, but the times given in the task configuration are now analyzed in µs ! (even if T#1ms is entered in the task configuration))
- **PLC data- and codesize:** Here you can set that size of the code and data memory, which will be allocated by the PLC when starting. The values must match with those configured in the target settings (memory layout).
- **PLC Filepath:** Here the PLC will store files like boot project, retain data etc. The path must exist and must have write access.

2.1.4 Ext. Config

Extended configuration options. These settings are system specific and should not be modified without exact knowledge on the PC hardware !



IO driversettings:

- At terminating the PLC the IO drivers will only be terminated by the RTE kernel if option **RTE Stops IOdrivers on exit** is activated.
- If hardware interrupts supporting floating points should be connected, activate option **Connect interrupts floatingpoint-save**. This option only should be used, if floating point operations are really needed in the interrupt handler, otherwise the resulting loss of performance would be senseless. Inversely an interrupt handler, which executes floating point operations while the option is deactivated, will cause sporadic system crashes.
- Activate option **Initial stackpool for IO drivers** if you want to get the size of the stack for tasks, which are created by IO drivers (not by CoDeSys).
- With the option **Disable automatic bus diagnostic** the system is prevented from writing diagnostic data into the diagnostic address which is used in the PLC configuration of CoDeSys.

Retain:

- If both of the options **Do not allocate retainmemory in RAM** and **Retainmemory on physical address** are activated, then all variables, which have got the flag "RETAIN" in CoDeSys, will be stored directly in the static RAM. For this purpose the PC hardware must have a SRAM-area on a fixed physical bus address.
- **Retain are saved by IOdriver** normally will not be used and only serves to allow the specific drivers to manage SRAM on their own.

- The setting **Retain are saved to file on exit** does not require a special hardware. But this option presumes that the PLC always will be terminated correctly resp. the PC gets shut down correctly. If this should be guaranteed, especially in industrial environment, using an USP (Uninterruptable Power Supply) is unavoidable.

NMI Scheduling:

- If option **Schedule on NM** is activated, the PLC will not install an interrupt handler for the timer chip of the PC. It will be assumed, that an IO driver is available which can generate or at least can acknowledge cyclic interrupts.
- The option **RTE connects NMI itself** will cause the PLC itself to set an interrupt handler on the NMI vector.
- If option **IO driver determines next task** is activated, the IO driver, which activates and acknowledges the NMI, will determine what the scheduler should do after each interrupt. If it cannot designate this, then the PLC itself will determine, when the operating system and the PLC tasks should be called. For this purpose the PLC will use the value given in **max. non-NT-Intervall** to check in case of asymmetrical time distribution, which interval should be used for the operating system and which for the PLC task. If possible, this method should not be used, because it assumes asymmetrical interrupt generation and is not flexible.
- If an IO driver itself wants to designate an scheduling interrupt, then option **IO driver connects interrupt** must be activated.
- Activate option **IO driver synchs high-prio-task** if you want to allow an IO driver (that which also defines the scheduler interrupt) to call the top priority task at a certain time which is defined by the driver itself.

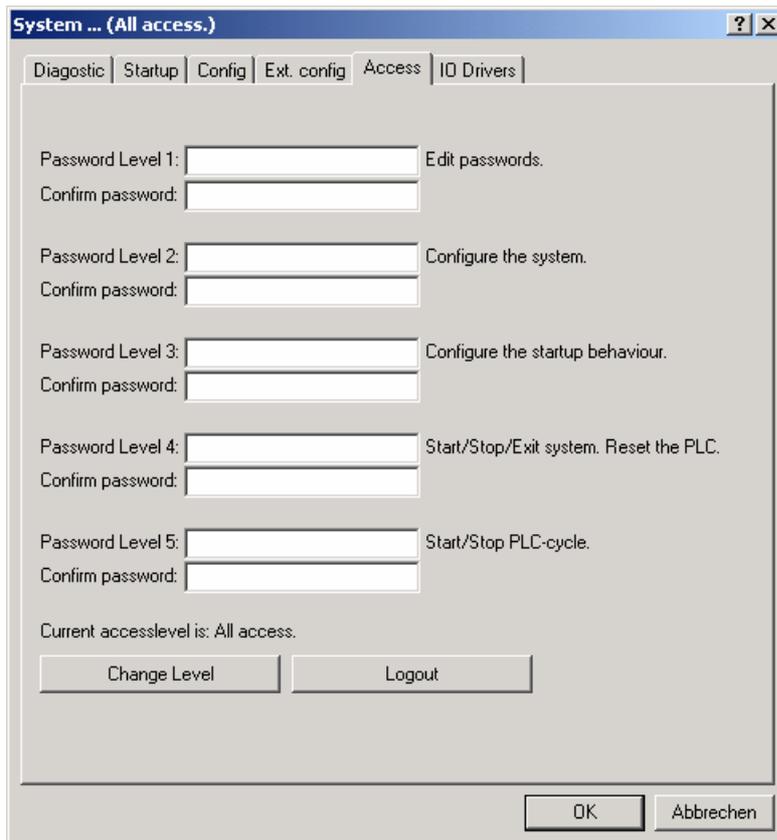
2.1.5 Access

Here passwords for 5 access levels can be defined to protect the particular operating actions against unauthorized or inadvertent usage. The system always will start automatically with the highest numbered level for which a password is defined, that means the level with the least rights. Only in level 1 passwords can be changed. The rights of the higher priority levels always automatically include those of the lower level groups and additionally the special access right for the function which they are responsible for.

To log in on a certain level, simply press button **Change Level** and insert the appropriate password. The level will be kept until you have logged out with **Logout**.

The following table lists the access rights concerning the particular actions, for the levels 1 to 5:

Level	Start/Stop Cycle	Start/Stop/Exit System	Startup/ Configuration	Extended Configuration/ IO-Driver config.	Change passwords
1	x	x	x	x	x
2	x	x	x	x	-
3	x	x	x	-	-
4	x	x	-	-	-
5	x	-	-	-	-



In **Password Level <n>** you must enter a password and confirm it in **Confirm password**.

2.1.6 IO Drivers

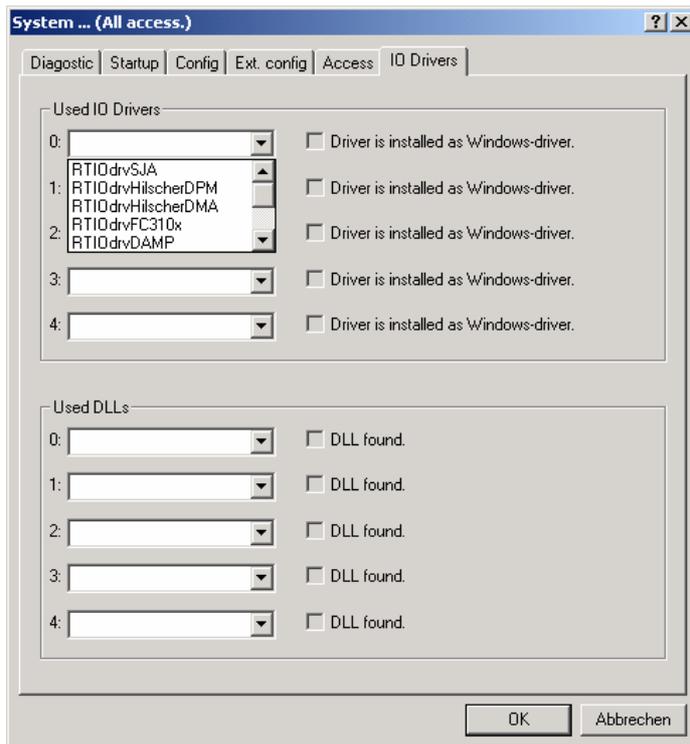
Here you can connect IO drivers (**Used IO Drivers**) and customer specific DLLs (**Used DLLs**). These entries can be made manually or selected from a list. The selection list will be filled by the setup.

In the registry the list of drivers and DLLs will be stored in \HKEY_LOCAL_MACHINE\SOFTWARE\3S Smart Software Solutions GmbH\CoDeSys SP\RTPLC\IO Driverpool resp. External DLL Pool. The names of the subkeys correspond to the entries in the selection list.

Drivers and DLLs must be configured in an ascendant order, starting with 0.

When a driver is entered, then a grey check will indicate that it is already registered as a Windows driver (**Driver is installed as Windows-driver**). If this is not the case, you have to install it, otherwise you will get an error message at system start.

Attention (for IO driver developers): If the object, which creates a driver, has the same name like the .sys-file (without extension), then 1 word will be enough to be entered here. Otherwise the driver must be entered by 2 words, separated by a space: <object name> <driver name>.



The assignment of the IO addresses used in the PLC program is either done in the PLC configuration in CoDeSys (only possible for CAN and Profibus) or directly in the address definition:

Each driver allocates a certain space of the address area. The size of this space is displayed in the **Diagnostic** tab (see above). In the same order in which the devices are listed there, their address spaces will be arranged one after the other in the address area.

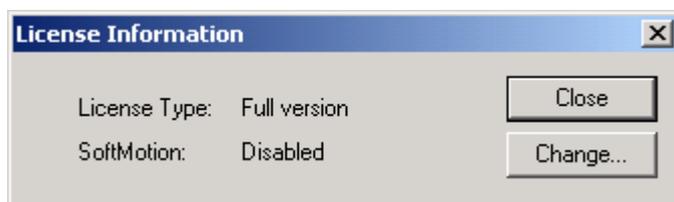
Example:

In the computer there are 2 Hilscher cards, one (Device0) of them allocating 512 Bytes, the other 3584. Both cards have been configured with the aid of SyCon, all modules are addressed without any gaps in the SyCon, starting with 0.

So in CoDeSys the first card will start with address %QB0 resp. %IB0, the second with %QB512 bzw. %IB512.

2.2 License ...

To license your version of CoDeSys SP RTE, choose the menu-item "License...". A dialog appears, where the existing licenses are displayed:



With „Change...“ the licenses can be edited. Therefore a wizard, which will guide you through the licensing procedure, will appear. You edit licenses using telephone or e-mail.

3 The 3S Standard IO Drivers

Regard: As soon as during setup the PC has been identified as a non-AT-compatible one, but the hardware working according to the UP-MPS specification, the RTIOdrvAPIC driver will be installed automatically. You neither can configure this driver nor you may remove it.

3.1 IO driver RTIOdrvHilscherDPM

The IO driver RTIOdrvHilscherDPM uses the values found in the registry entry (HKEY_LOCAL_MACHINE\System\CurentControlSet\Services\RTIOdrvHilscherDPM\Params) Device 0 to 4, to find its cards resp. to tell which cards it has found.

Valid entries are:

	Possible values	Comment
Device0..4\Type (String)	8K_DPM or 2K_DPM	The Hilscher-CIF/COM cards are available with 2K(1k IO data) and 8K(7k IO data).
Device0..4\Address (DWORD)	32-bit-value	The physical bus address of the card.
CreateComTask	0 or 1, default: 1	If this value is available and =1, then a communication task will be created, which will constantly check the Hilscher cards for any communication messages. If the value =0 or does not exist then this task will not be created.
Device0..4\DoNotConfigure	0 or 1	If this value is 1, then the Hilscher card will be ignored by CoDeSys in the PLC configuration
Detected	String „Not found” or “Card found: <Information on the card>	This entry is used to show which cards have been detected by the driver.
in Params: AutoDetection	0 or 1	This entry can be used to activate (1) or deactivate (0) the automatic search for Hilscher cards. If a card is not searched automatically, it has to be entered manually by Address/Type.
In Params: UseInterrupt	Bitwise	Bit 0 = 1: The driver uses interrupts for PCI-Pbmastercards. Bit 1 = 1: The driver will use the 2. interrupt channel of the PCI controller. This is only necessary for special hardware implementations. Bit 2 = 1: The driver will use negative interrupt polarity when initializing the PCI-controller. This is only necessary for special hardware implementations.
In Params: WaitFlags	Bit coded (exclusive, set only one of the bits!)	Bit0: With this flag the driver is configured to wait before updating the IO data until the last PB-cycle is finished. Bit1: With this flag the driver is configured to skip the update for this cycle, if the last PB-cycle is not finished.

If you are using PC bus versions and manual configuration (AutoDetection = 0), consider that the address of the card can change as soon as new PCI devices get added.

The Hilscher card driver works with all Hilscher cards, even with such which cannot be configured in CoDeSys. Nevertheless the process data transmission mode "user controlled, buffered" can be used in the Hilscher configurator (e.g. Interbus). By doing so data inconsistencies within a task can be avoided.

The RTIOdrvHilscherDPM usually searches for its cards independently, except there is an entry "AutoDetection"=0 in (HKEY_LOCAL_MACHINE\System\CurentControlSet\Services\RTIOdrvHilscherDPM\, which will prevent the automatic detection of cards.

The automatic detection mechanism will enter the found cards in each "Device" key in HKEY_LOCAL_MACHINE\System\CurentControlSet\Services\RTIOdrvHilscherDPM\Params.

The access on the process image of the card will be done cycle-(task)synchronously, if in the SyCon the „user controlled, buffered“ access has been set. The driver will do a handshake with all Hilscher cards via the PdCom-Flag.

When using Profibus- or CANopen cards the PLC configurator of CoDeSys can be used. There any desired IO addresses can be assigned on the channel level. From the assignment in the PLC configuration the system will know, which card has to be addressed.

When using Hilscher cards, the message box interface can directly be accessed via the library *Hilscher.lib* (See for details in the documentation delivered by Hilscher with the cards). In extension to that for CANopen cards of Hilscher the *Hilscher_SDOtransfer.lib* can be used to build up a SDO communication with CAN nodes. Both libraries are part of the setup.

Together with the latest version of the driver, the HilscherPBInfo.lib can be used, if interrupts are enabled. The actual PB-scan time can be determined.

3.1.1 The SysLibDpV1.lib

The library SysLibDPV1.lib can be used together with the RTIOdrvHilscherDPM and an appropriate PB-card. The library contains 2 function blocks:

DPV1_Read

DPV1_Write

With these FBs the acyclic communication services of class 1 can be used (MSAC_C1). The function blocks have the input parameters:

ENABLE:BOOL;	With this input the FB is activated, a service is started.
Device:INT;	With "Device" the card is chosen. If only 1 card is present in the system, always pass 0 here.
StationAddr:INT;	The station address of the card.
Slot:INT;	Slot and Index are defined within the specification of the class 1 DPV1-services and depend on the slave.
Index:INT;	
Len:INT;	The number of bytes to write (when calling WRITE), respectively the size of the local buffer (when calling READ).
buffer:DWORD;	The address of the local buffer. It has to be determined by using ADR(). Typically this is the address of a byte array.

When calling the FB, there has to be a rising edge on the ENABLE-input. Then the FB has to be called with ENABLE=TRUE until the READY-output signalizes TRUE.

With the STATE-output the correct execution of the service can be checked.

At the Size-output the FB shows, how much data actually has been transferred.

In case of an error during the execution the first byte of the read-buffer (only when using READ) the error code from the Hilscher-Card is passed. See the appropriate documentation of the card for more information.

When Hilscher cards are used, the message box interface can directly be accessed via the library *Hilscher.lib* (See for details in the documentation delivered by Hilscher with the cards). In extension to that for CANopen cards of Hilscher the *Hilscher_SDOtransfer.lib* can be used to build up a SDO communication with CAN nodes. Both libraries are part of the setup.

3.2 IO driver RTIOdrvFC310x.sys

The RTIOdrvFC310x works with the Beckhoff Profibus master cards FC3101 and FC3102.

The cards are detected automatically and only can be configured with the PLC configurator of CoDeSys.

In the 2-channel version of the card (FC3102) 2 devices will be detected.

The driver will read the software status of the firmware from the card; but you will be able to use all functions only if the card at least contains the firmware version 2.0. (available from approx. August 2002).

The driver uses a configuration entry in the registry:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvFC310x\Params contains the following entry: Debug, coded bitwise. (If the value is missing, 0 will be assumed)

Bit0: If "1", then at termination of the driver (that means at termination of the RTE) in this registry key error statistics for each configured slave will be stored. By that you can get information on the quality of the PB network.

Bit1: If "1" then the task synchronous mode of the cards will not be used. So the PB cycles will run asynchronously to the PLC tasks.

Together with the FC310x the *SysLibFCDPV1.lib* can serve to use the DPV1 communication services Read and Write.

3.2.1 The library SysLibFCDPV1.sys

The library SysLibFCDPV1.lib can only be used together with the RTIOdrvFC310x and a corresponding PB-card. The library contains 2 function blocks:

- FC_DPV1_Read
- FC_DPV1_Write

Using these blocks, asynchronous data transfer via class 1 services can be used. (MSAC_C1). The function blocks both use the following input parameters:

ENABLE:BOOL;	Setting this input from FALSE to TRUE, a job will be started.
Device:INT;	Here the card, which is used for the transfer, is chosen. If there's only one FC3101 in the system, always pass 0 here.
StationAddr:INT;	The station address of the slave to communicate with.
Slot:INT;	Slot and index are defined within the PB-specification and differ in content from slave to slave. See the corresponding documentation for the specific slave.
Index:INT;	
Len:INT;	The number of bytes to write (for a WRITE-job), or the size of the local read-buffer (for a READ-job).
buffer:DWORD;	The address of the local buffer. Use ADR()-operator to determine it. Typically this is an address of an ARRAY OF BYTE.

To use the FBs, put a rising edge to the enable-input. Then call the FB cyclic, with enable set to true, until the READY-output of the FB is set to TRUE.

Check the success of the job using the output STATE of the FB.

When executing a READ-job, the error codes are written to the READ-buffer in case of an error.

These codes refer to the documentation of the FC3101/FC3102.

The function FC_ClearMessageBox is used to clear the message-interface of the card and should be called after a RESET or after a complete download. The function should be used as follows (from a cyclic task):

```

IF ClearFCMessageBox THEN
    dwMsgBoxState := FC_ClearMessageBox(0);
    IF dwMsgBoxState = 0 THEN
        ClearFCMessageBox := FALSE;
    END_IF
END_IF
    
```

The return value of the function is 0, when no more jobs are to be deleted. The return value is 1 with the jobID of the deleted job in the upper 16 bits. 16#FFFFFFFF is the return value in case of an error.

The only input param is the index of the card on which the jobs are to be deleted.

3.2.2 The library FC_SlaveHandling.lib

Together with the FC310x (from FW-Version 2.17 on) the system library FC_SlaveHandling.lib can be used. The library is used to deactivate/reactivate dedicated slaves at runtime.

The function block has the following input parameters:

ENABLE : BOOL;	The function block is activated on a rising edge.
iSlaveAddress : BYTE;	The station address of the slave to activate/deactivate.
iNewState : SETSTATE;	The state the slave should be set to.

After deactivating a slave, the FC3101/FC3102 will no longer communicate with this slave, the same as it was never been configured. After reactivating the slave, the master will restart the communication.

After a download of a CoDeSys-application, all configured slaves are active. The library is only needed if the application wants to deactivate dedicated slaves at runtime.

3.2.3 The FC310x card in slave mode

The card can appear as a PB-slave on the bus, too. (A 2-channel card (FC3102) also mixed as a master and a slave, in 2 nets). Therefore you have to tell the driver, which card should appear as slave and what baudrate it should use. This is done via 3 registry-entries. These entries have to be set up manually, for example using regedit:

(The entries are all located within the driver's configuration-key:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvFC310x\Params)

The DWORD-value "DeviceXIsSlave" has to be set to 1, where X is the index of the card. (So, if you're using a single FC3101 the name resolves to "Device0IsSlave".)

The DWORD-value "SlaveXBaudRate" has to be set to the desired baudrate (the card is not able to detect the baudrate!). (Again an example with only one FC3101: the name is "Slave0BaudRate" and it should be set to 12000000 to set the baudrate to 12MBaud.)

It is absolutely necessary to set the DWORD-value "Debug" to 2.

Attention: These values are all read when starting the driver. If they get modified, the system must be restarted once via the RTE menu by commands **Stop System** and subsequently **Start System**. (The driver also will be restarted by doing so.)

3.3 IO driver RTIOdrvCP5613.sys

The RTIOdrvCP5613 works with the Siemens Profibus master CP5613.

The driver needs a firmware file, which comes with the card. The file must be available in the runtime directory of the RTE. The file is named „FW_5613.bin“.

The path of the runtime directory is stored in the registry in

HKEY_LOCAL_MACHINE\SOFTWARE\3S Smart Software Solutions GmbH\CoDeSys SP\RTPLC\
in "Path".

The driver evaluates 2 configuration switches of its registry key. The key:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvCP5613\Params

- „Debug“: If this is not 0, then the LDB, which is created from the CoDeSys configuration data not only will be written on the card, but also as a file on the hard disk. This file also will be created in the runtime directory and gets the name „out_x ldb“, whereby x will be replaced by the index of the card.
- „ReadFile“: If this is not 1, then the CoDeSys configuration will be created and – if "Debug" is 1 – also stored on the disk, but on the card a LDB file will be written, which must have the name "in.ldb" and must be available in the runtime directory.

If you want to work with the card in the RTE, you in each case have to create a PB configuration in CoDeSys, even if the configuration should be read from a file. The driver only will start the card, if configuration data have been loaded with the program.

3.4 IO driver RTIOdrvDAMP

This driver works with any card offering a shared (with the PC) memory area on a fix physical address (Memory mapped dualport RAM).

This memory area must be segmented in an input and an output data area and the inputs and outputs of the card must be clearly allocated to the start addresses of these areas by the offsets.

The driver must be manually configured in the registry with the following entries:

In HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvDAMP\Params,

and below 'Params' for each possible card there is a "DeviceX" key. X is the index of the card (0 to 4).

In 'Params' there is a value which will be evaluated: If 'Simulation' is 1, then no IOs will be read or written, the driver can be used for testing the program.

In each "DeviceX" subkey there are the values

- „Flags“: If there is a card with the given index, flags must be "7", otherwise "0".
- „PhysAddrIn“: The physical bus address of the input data area on the card.
- „PhysAddrOut“: The physical bus address of the output area of the card.
- „InLen“: The length of the input data area in Bytes.
- „OutLen“: The length of the output data area in Bytes.

Example:

A digital IO-card with 64 Bytes input and output data should be used by the PLC.

The card is on the ISA bus and has address 0xC0000. At the beginning of the address area the input data are stored, subsequently the output data. In this case the configuration values must be set to (in Device0):

- Flags = 7
- InLen = 64

- OutLen = 64
- PysAddrIn = C0000
- PhysAddrOut = C0040

3.5 IO driver RTIOdrvApplicom

This driver can work with Applicom IO cards (ApplicomIO). Any bus system can be used, the interface between cards and PLC will always be the same. The cards can only be configured with the configuration tool provided by Applicom together with the cards.

When detecting the IO addresses of the particular modules the programmer has to consider the configured modules as lying one after the other in the process image of the PLC.

The driver will independently search and start all ApplicomIO cards available in the system.

The driver will not evaluate any registry configuration keys.

3.6 IO driver RTIOdrvIBS (IBS = Interbus, ibpcimpm.sys)

This driver works with the InterbusS master card IBS PCI SC/I-T delivered by Phoenix Contact.

The RTIOdrvIBS is always installed by the setup. To use an IBS PCI SC/I-T under Windows 2000 or XP, another driver from the w2k_Xp Drivers directory of the setup-CD has to be installed, using the .inf-file there. A possibly already installed driver of the company Phoenix has to be uninstalled.

The entry in the System..., IO drivers tab is, as mentioned in the pulldown-list also, „lbsisasc1N1D ibpcimpm“.

HINT: The driver is not able (still) to setup device-instances, means only one card per PC can be used.

The cards must be configured with the configuration tool delivered by Phoenix (CMD G4). Thereafter the RTE can exchange IO data with the InterbusS network, which was configured for the card. The IO-addresses in the PLC program correspond to those which have been set in the CMD tool.

In order to work with driver and card, first the CMD tool provided by Phoenix Contact must be installed. This will later be needed to load the configuration onto the card and to save it there in the parameterizing memory. If you want to configure the card remote via the serial interface, then this step can be left out. But anyway regard that, if you install the CMD tool later, the 3S driver might be overwritten. In this case you have to re-install the 3S system after the CMD installation (another remedy: save the driver ibpcimpm.sys from System32\drivers and put it back there after the CMD-Installation has been finished.)

To make possible a restart of the card after a bus error without using the configuration tool CMD G4, you have to use „Write parametrizing memory“ in the CMD tool after configuration and start of the card. This is necessary to save the configuration and the startup configuration parameters on the card. So in case of a bus error the bus will be restarted by „Online->Reset“ (in CoDeSys Online menu) or by a „Reset Warm/Cold/Hard“ in the user menu of the RTE.

3.7 IO Drivers RTIOdrvSJA and RTIOwdmPCAN

To use the CAN-Card PCAN-PCI of Peak-Systems, choose the entry RTIOdrvSJA from the list in IO Drivers, on an NT4.0 system.

To use such a card on an XP/Win2000 system please first install the card then install the .inf file from our Win2K_XP\Peak Systems PCAN directory.

3.8 IO Drivers RTIOdrvAutomata and RTIOwdmAutomata

To use a SERCOS-Card of the company Automata choose the IO-driver-entry RTIOdrvAutomata on an NT4.0 system.

To use such a card on an XP/Win2000-System, first install the card physically and then install the .inf-file from the Win2K_XP\Sercos Automata of the installation CD resp. the installation directory of the RTE. Then choose the entry RTIOwdmAutomata.

The driver is only able to work with one single card.

On an NT4.0 system the ISA/PC104-version of the card can be used. Therefore the driver has to be configured.

With the registry entries "Address" and "Interrupt" within the key
\\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RTIOdrvAutomata\Params
the physical bus address and the interrupt vector to be used are supplied to the driver.

The PCI-version of the card can be used without changing anything after executing RTE setup.

Under Windows XP and 2K only the PCI card is useable at the moment.

3.9 IO drivers RTIOdrvHMS and RTIOwdmHMS

With the RTIOdrvHMS the Profibus-PCI-cards of the company HMS can be used. The master and slave versions of the card can both be used.

The driver will recognize its cards himself. It is not configurable and can work with up to 5 cards master and slave, where master and slave can be mixed.

To use a HMS card under Win2K/XP, install the card physically into the PC and then use the hardware assistant to install the .inf-file from Win2K_XP\PBMasterSlave HMS.

In the IO Driver tab of the system...-menu always use the entry RTIOdrvHMS, under all Window-systems.

Hint: Before doing that, uninstall a possibly already installed HMS-driver under all circumstances! Make sure, the hardware assistant installed the driver supplied by 3S.

3.10 IO driver RTIOdrvAPIC

As mentioned above the driver will be installed automatically by the system as soon as this is required. The driver will configure himself. In case one of the values in the registry-key from below

- SetClocksPerUS or
- SetCountsPerMS

Is missing in the registry, the driver will measure the frequency of the interrupt counter. In that case the start of the system can last up to 30 seconds. The driver then writes the measured values to these registry values, to get them on the next startup. With these values the driver can be configured manually.

It will also write an error code to the registry for diagnostic purposes.

In \\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvAPIC the following bits can be set in the entry "ErrLog" :

Bit 0: NO_ERROR	Set as soon as the initialization has been finished successfully.
Bit 1: ERR_APIC_NOT_ENABLED	The needed interrupt controller has not been initialized.
Bit 2: ERR_MISSING_CPU_FEATURE	The CPU of the PC does not have the required interrupt controller.
Bit 3: ERR_APIC_NOT_MAPPED	like before
Bit 4: ERR_IOAPIC_NOT_MAPPED	like before
Bit 5: ERR_INTTEST_TIMEOUT	No interrupt source could be initialized.
Bit 6: ERR_NO_CPU_CLOCKFRQ	The clock frequency of the PC could not be evaluated.

Bit 7: ERR_MY_INT_ISUSED No free interrupt vector could be found..

Bit 8: ERR_NO_FREE_INT_FOUND like 7

Bit 9: ERR_TIMERINT_NOT_DISABLED This is not a APIC platform.

If you get one of the described errors, please contact the 3S Support Team.

3.11 IO Driver RTIOwdmCANAutomata

To work with a CAN-card of the manufacturer Automata, you use this driver. The driver is implemented for Windows 2000/XP. If you want to use the card under NT4.0, just ask at 3S.

To work with one of the cards CAN PCI 1N, 2N or 2NNV (described as 70034500, 70062810 and 70062800 in Automata's documents) first install the card physically and then, on the next start of the PC, install the driver from the directory "CAN Automata" with the hardware manager of windows.

After the card has been installed, there's an entry to be chosen in the IO Driver's-tab of the System....-Menu.

If you are using the card with two CAN-channels and NV-ram, on startup of the system the driver will automatically change the system's retain-settings to use direct retain variables on the physical address of the NV-ram on the card. You can verify this by opening the Ext.config-tab in the System...-Menu

To prevent the driver from changing the retain-settings, there's the possibility to create a registry entry: Create a new DWORD-value within

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOwdmCANAutomata\Params with the name „NoRetain" and set it to 1.

3.12 IO Driver RTIOwdmSofting

To work with a Profibus Master card „PROFIboard PCI" of the company Softing and the RTE, the card has to be installed with the file RTIOwdmSofting.inf.

The driver will work on Windows 2000 and XP.

In HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOwdmSofting\Params an optional value "SyncMode" of type DWORD can be set to 0 explicitly. Then the card will run its own profibus update cycle. Normally it is better to let the PLC trigger the PB-cycle. This is the default behaviour.

4 Communication

The programming system CoDeSys can communicate with the RTE in two ways:

- via the Shared Memory driver
- via the TCP/IP Level2 Route driver

Other communication ways are possible (e.g. Profibus FMS via two Hilscher PB-cards), but they are not part of the RTE setup and have to be requested at 3S GmbH.

4.1 Shared Memory Driver

To get the driver available, a 3S-Gateway also must run on that computer where the PLC is running. In the CoDeSys programming system select in dialog „Online → Communication parameters“ that gateway which is running on the PLC computer and choose 'Shared Memory Driver' as communication channel (if necessary, create with „New...“).

Hereupon this the programming system should be able to communicate with the PLC.

4.2 TCP/IP Level2 Route Driver

If you want to use this driver, it doesn't matter on which computer the 3S-gateway is running.

Select in the CoDeSys programming system in the dialog „Online -> Communication parameters“ the desired gateway, e.g. localhost. Then select the communication channel: "TCP/IP Level2 Route" and enter the target address: Host name or IP address of the target computer.

After that the programming system should be able to communicate with the PLC.

5 System Diagnosis

5.1 Bus diagnosis of the IO drivers

All IO drivers are cyclically requested to fill the following diagnosis structure *GetBusState*, as soon as an IEC task has written its process data to the drivers:

```

TYPE GETBUSSTATE:
STRUCT
  BOLDENABLE : BOOL;
  ENABLE: BOOL;
  DRIVENAME: POINTER TO STRING;
  DEVICENUMBER: INT;
  READY: BYTE;
  STATE: INT;
  EXTENDEDINFO: ARRAY[0..129] OF BYTE;
END_STRUCT
END_TYPE

```

This structure is defined in library **BusDiag.lib**.

The IO drivers write this structure to the diagnosis address which is defined in the configuration for each bus master. This functionality will only be available for bus masters which are configured in CoDeSys.

For each potential participant in the bus a byte will be reserved in EXTENDEDINFO, where the bits 0 – 2 will be used as follows:

Bit 0: The participant is available in the configuration.

Bit 1: The participant is available on the bus.

Bit 2: The participant returns an error.

As soon as a participant which is available on the bus returns an error, the corresponding diagnosis information can be read with the help of the function block *DiagGetState* (also defined in BusDiag.lib).

DiagGetState will be called for a certain participant of the bus.

```

FUNCTION_BLOCK DiagGetState
VAR_INPUT
  ENABLE: BOOL ;
  DRIVENAME: POINTER TO STRING ;
  DEVICENUMBER: INT ;
  BUSMEMBERID: DWORD ;
END_VAR
VAR_OUTPUT
  READY: BOOL ;
  STATE: INT ;
  EXTENDEDINFO: ARRAY[0..99] OF BYTE ;
END_VAR

```

The function block will be executed as soon as a rising edge is detected at the ENABLE input.

DRIVENAME is the name of the driver (address of the name) to which the diagnosis request should be sent. If "0" is inserted here, the diagnosis request will be sent to all available drivers.

DEVICENUMBER identifies the bus which is managed by the driver. (The Hilscher card driver e.g. can handle up to 5 cards (busses).) The index is based on 0.

BUSMEMBERID identifies the participant of the bus. For a CANopen card BUSMEMBERID e.g. will consist of the NodeID, for a PB-DP card it will render the station address of the participant etc. (in general: the BUSMEMBERID allows a clear identification of the participant). How a bus participant is identified is bus and driver specific.

READY is an output parameter which will be set as soon the diagnosis request has been completely processed. In STATE you will then find one of the values of

```
VAR_GLOBAL CONSTANT
  NDSTATE_INVALID_INPUTPARAM:INT:=-1;
  NDSTATE_NOTENABLED:INT:=0;
  NDSTATE_GETDIAG_INFO:INT:=1;
  NDSTATE_DIAGINFO_AVAILABLE:INT:=2;
  NDSTATE_DIAGINFO_NOTAVAILABLE:INT:=3;
END_VAR
```

The byte array EXTENDEDINFO can contain up to 100 bytes of customer specific diagnosis data of the bus participant.

6 The Task Configuration together with CoDeSys V2.3

CoDeSys V2.3 for CoDeSys SP RTE supports an extended task configuration, where additionally the following features are available:

- watchdog supervision specific to each task: a maximum time span can be specified for each task, which cannot be violated.
- Microseconds can be specified as cycletime.
- A „free wheeling“ task can be created.
- System events can be connected with IEC-functions (callbacks) by picking them from a list.

6.1 General

Within the task configuration window of CoDeSys up to 16 tasks can be specified. The scheduler performs a pre-emptive, prioritized multitasking on them, where tasks, specified as “cyclic” are scheduled periodically with the projected cycletime.

As usual, the unambiguity of the task priorities has to be obtained. (More than one task with the same priority leads to malfunction of the scheduling process. If you need this, just add more than one entry point under one task. In newer versions of CoDeSys a compile error is generated in this case.)

6.2 Task specific watchdog

If the new task configuration is used, the entry “watchdog -> enable with multiplier” from the “Config”-register of the “System...”-menu is ignored.

For every single task a maximum execution time (at least the cycle time) can be specified. In case this execution-time is exceeded by the task the normal behaviour is a PLC-stop. The PLC-stop can be avoided by specifying a callback function on the system event “escpt_watchdog”. The return value determines, whether the PLC should be stopped (return value = 0) or whether the task should be executed further on (return value <> 0). See further information on system events in chapter 6.5).

(Hint: The sensitivity-parameter of the task configuration is not used within this target system. It is set to be always 1 on the PLC.)

6.3 Microseconds as time base

In the new task configuration the user can switch from “ms” to “ μ s” as time base. (Just type a number and then change the unit from “ms” to “ μ s”.) But as long as in “Config”-register of the “System...”-menu the time base is not set to microseconds, the effective task cycles will be a multiple of 1 ms.

As soon as the time base is changed to smaller values, the effective cycletimes can become an even multiple of the chosen time base.

For example, to create a 100 μ s-task, the smallest value that is currently possible, choose the time base 50 μ s in the “Config”-register. Configure a task with 100 μ s cycle time and you can watch (if you programmed a simple counter $a:=a+1;$) that the task is executed 10000 times per second.

6.4 Freewheeling task

A freewheeling task is a task, that starts from the beginning at once when it finishes.

The cycle time of a task like that is determined by the execution time of the application, not by configuration.

Be aware that a freewheeling task can exist only once and must have the lowest priority of all the configured tasks, because no other task with a lower priority will ever be called.

6.5 System events

Within the task configuration, system events can be connected with function calls (callbacks) to IEC-functions (application defined functions). This means, the system (RTE) itself calls an application defined function on some system events. Thus the application can react on certain events.

Remember that the functionality of such callback functions is restricted: You mustn't create local variables, you can't use asynchronous functionalities, like filesystem-accesses, you cannot set breakpoints to debug such functions. But you can call other application-defined functions from there.

To use this feature, the system library **SysLibCallback.lib** must be included in the project.

The system events simply call the respective functions. The only event, for which the return value has effect on the behaviour, is the WATCHDOG_EXCEPTION (see above, chapter 6.2: task-specific watchdog).

7 System libraries

Together with CoDeSys SP RTE some system libraries can be used. The use of them is shown in the following sections. System libraries not mentioned there cannot be used together with CoDeSys SP RTE.

7.1 **DIICall.lib together with SysLibSystemCall.lib**

Together with the DIICall.lib you must always include SysLibSystemCall.lib into the project, otherwise compile errors will result. The library makes the asynchronous call of user defined functions, located in Windows®-DLLs. To create such a DLL, refer to our toolkit “CustomDLL Toolkit”, shipped together with the RTE-setup.

With the function block DIICall, to which a function name, an input- and an output buffer as well as a timeout is passed, a function in normal userspace surroundings can be called up.

After the function returned, the output parameter of this FB is set to STATUS_READY or STATUS_ERROR, according to the return value of the DLL-function. If the function doesn't return within the timeout, the status is set to STATUS_TIMEOUT and the execution of the function is stopped.

The STATUS_EXCEPTION is only set, if the DLL-function itself generated an exception.

7.2 **SysLibCallback.lib**

If you want to connect special events to application-defined function calls, you can use SysLibCallback.lib. There's a function SysCallbackRegister, to connect a function with an event, and one (SysCallbackUnregister) to disconnect the event.

The events which are currently supported are shown within the extended task configuration of CoDeSys V2.3.

See chapter 6.5, Task Configuration, section “Systemevents” .

7.3 **SysLibCom.lib**

The SysLibCom.lib is used to access the serial interface(s) of the PC from the application-program.

To use a serial interface, the application must perform the following steps :

- Open the interface with **SysComOpen**, passing one interface name out of COM1...8.
- Set the settings of the interface using **SysComSetSettings** or **SysComSetSettingsEx**. If SysComSetSettings is used, the interface is configured to 8 databits, no hardware handshake. To set more detailed the settings of an interface, use SysComSetSettingsEx. See the comments of the structure definition “COMSETTINGSEX” of SysLibCom. Be aware, you can only set the settings of an interface once after you opened it. To change the settings of an opened interface, first close it and then reopen it.
- Now you can use **SysComRead** and **SysComWrite** to read data from the interface to a buffer, resp. write data from a buffer to the interface.
- If you do no longer need the interface, free the resource by closing the interface, using **SysComClose**.

7.4 SysLibFile.lib

To work with the PC's file system, use SysLibFile. The following functions from SysLibFile are currently supported:

- SysFileOpen
- SysFileClose
- SysFileRead
- SysFileWrite
- SysFileDelete.

All other functions contained in SysLibFile are currently not supported.

To open a file, the function **SysFileOpen** is used. Pass a fully qualified filename and the mode, in which the function should be opened to the function:

- Mode "w" opens the file for write access. Be careful: Just opening a file with "w" and then subsequently closing the file overwrites any existing file and creates a file of 0 bytes length.
- Mode "r" opens a file for read access. The file handle returned by SysFileOpen will be invalid (value -1 resp. 16#FFFFFFFF) if the file does not exist. The file is opened for reading in sequential mode, means every read access moves the read position the read number of bytes.
- Mode "a" opens a file for write access and appends the written data to the end of a probably existing file.

The functions **SysFileRead** and **SysFileWrite** read/write to/from files. They each get as inputs a buffer and a data handle (returned by function SysFileOpen).

When the access to the file is no longer needed, close the resource using **SysFileClose**.

7.5 SysLibPorts

With the library SysLibPorts.lib, the application is able to access the IO-ports of the PC directly.

Therefore the accesses are distinguished by the size of the desired access: There's a function for byte-, word- and doubleword-size.

Example: To read a byte from the parallel port of a standard PC, call **SysPortIn(16#378)**. To write to this port (useable to check the task's timing with an oscilloscope) call **SysPortOut(16#378,value)**.

To use SysLibPorts, a very detailed knowledge about the used PC-hardware is necessary.

Port addresses must fit the range 0...65535.

7.6 SysLibTime

The SysLibTime is used for exact time measurements and for retrieving the current system time. Be aware that **CurTime** is a very fast function, returning immediately. But the function **CurTimeEx**, that additionally returns the current system time, possibly needs one millisecond to return.

A short example on the use of SysLibTime:

Declare two variables:

```
VAR
    ct : CurTime; (*The instance of CurTime is created.*)
    syst : SysTime64; (*An instance to return the values is created.*)
END_VAR
```

Implementation:

```
ct(SystemTime := syst);
```

After this call, syst contains a 64-bit value that represents the time in μs since booting.

(Remark: To compute time differences up to 4294967295 μs , it is enough to use the Low-parts of the SysTime64-structure.)

7.7 SysLibSockets

The SysLibSockets is used mainly by the network variables, transmitted over UDP.

This library can also be used by the application for own transmissions and protocols.

See the **Networkvariables_UDP.lib**-library for details on the use of the functions. The networkvariables-library is shipped with the CoDeSys programming system.

7.8 SysLibShm.lib

This library provides functions for accessing a memory area which is used in common by several processes resp. referencing a physical address (Shared-Memory, shortcut ShM).

If the target system is supporting the functionality, the library functions can be used to open and to close the ShM and to read and write from it. The reading, writing and closing functions need the handle which is returned by the opening function. The execution is synchronous.

The execution may last some 10ms. It is not recommended to call this function within a cyclic control task, only during initialization.

The functionality of the library is described in detail in the online documentation of CoDeSys.

The library can only be used for mapping of physical memory, not for creating shared memory areas to share with other windows processes.

7.9 SysLibPciCards.lib

The library is described in the online help of CoDeSys.

8 Behaviour in case of runtimeerrors

There are some different reasons, which lead to a cycle-stop of the user application.

These errors are:

Number	Name	Reason
16	Watchdog	The application needed more than the defined cycle time to complete the current cycle. The cycle time, which is controlled, is computed by the watchdog-factor (see "Config" in the chapter "Operation via the Service icon") If the CoDeSys-version 2.3 is used, it may support the definition of the watchdog factor within the task configuration. Then the factor there is used.
20	Fieldbus error Error in configuration data	An IO-driver could not initialize himself. Can only happen when starting the PLC. One of the used fieldbus cards did not accept the data coming from the PLC-configuration.
21	Error update IOs	Can only happen after loading the program. Happens, if an IO-address, which no IO-driver can handle, is used.
81	Access violation	The application tried to access a memory address, which is invalid. Maybe a pointer is used and not correctly initialized?
258	Division by zero	The application tried to compute a division by zero.
336-343	FPU-Error	An invalid or impossible floatingpoint-operation was executed..

All these errors lead to the same behaviour of the PLC:

- The outputs are, if configured, set to 1 or 0.
- The PLC-cycle is stopped. (Remember, only the task that caused the error is left immediately. The other tasks, that may be active, will complete their current cycle and not begin a new one.)
- The IO-update of all tasks will continue. (Only the PIO may have been set to 1 or 0.)

The error that occurred is shown within "Diagnostic"-tab of the service menu.

The PLC will store the error with (if possible) the callstack. After logging in with CoDeSys, the line within the application can be found. (Online -> callstack)

9 APPENDIX

9.1 The Registry Entries used by the Runtime System

The registry key RegKeyRuntime will be found at:

\\HKEY_LOCAL_MACHINE\SOFTWARE\3S-Smart Software Solutions GmbH\CoDeSys SP\RTPLC

(All these entries normally are handled by the „System...“-Configuration dialog and need not to be edited manually!)

With this key the runtime system will look for the following parameters:

Name	Possible Values	Meaning
LastExit, DWORD	0 or 1	Before one tries to load a bootproject, LastExit=0 will be set. If the PLC crashes (is not terminated properly) this value will not change to 1 as normal. If it is 0, then no boot project can be loaded.
NmiConnect, DWORD	0 or 1	If this value is set to 1, an interrupt-handler of the runtime system will be put to the Int-vector "2". Currently only useful in combination with NmiSchedule.
NmiSchedule, DWORD	Bitwise analysis	Bit0: If this bit is set, the interval will not be modified and no own routine will be put to the interrupt vector of the timer chip. The scheduler will be called by the Nmi-Handler routine. Bit1: If this bit is set, the IO driver, which handles the NMI, can define the time at which NT must be scheduled. Bit2: If this bit is set, the RTE kernel cannot install an interrupt routine for the cyclic interrupt on its own. This must be done by an appropriate IO driver, which on request will create a cyclic interrupt and call the routine, which normally serves as NMI handler (or connect it with its cyclic interrupt)
MaxNonNtInt, DWORD	50 – 950	If the scheduler is working on an NMI, which is created asymmetrically (with the aim to allocate less CPU time to NT than to the PLC), then this value is used to determine at which interrupt and after which IEC tasks NT will be served, whereby NT will always lose out.
MaxPLCTime, DWORD	10 – 90	If the scheduling is conventional (by timer tick), this number determines the time ratio of NT and the runtime system. "90" thus means that the PLC will get 90% of the CPU time, NT only 10%.
Path, STRING	\??\<valid directory name, starting with a drive letter>	The path, where all files concerning the runtime system will be searched and stored. Max. 128 characters
RetainFlags, DWORD	all 32 Bits used singly	Default: Retain memory will be allocated dynamically in the RAM, Save to file at terminating. Bit0 = 1: Retain memory will not get allocated dynamically. Bit1 = 1: Retain memory will be assigned to a physical address. If only bits 0 and 1 are set (3), it will be presumed that the retain area is located in an SRAM. The retain data will not be saved explicitly. Bit2 = 1: Retain will be saved in file at terminating, like default. Bit3 = 1: Retains will be saved via IO driver, cyclic.
RetainAddr	valid physical memory address	Must exist if Bit1 has been set in RetainFlags. This value will be used as a physical memory address for storage of the retain data. The length you find in RetainSize.

Name	Possible Values	Meaning
RetainSize	Total size of the retain data (in physical address, resp. size of area which should be allocated)	Must exist, if Bit1 is set in RetainFlags.
HandleBlueScreen	1 or 0	If this value exists and equals 1, an NT-Bluescreen handler will be started as soon as the system crashes. In this case the PLC will continue running (if not responsible for the crash itself). If this value does not exist or equals 0 then the computer will "freeze" with the common Bluescreen behaviour in case of a crash.
StopIODrivers	0 or 1	If the value is 1, then the RTService will automatically stop the drivers as soon as the system terminates. (The RTService also starts all listed drivers automatically.)
OutputFlags	all 32 Bits used individually	Bit0: 0: Outputs will not be forced as soon as the PLC stops. Bit0: 1: All Outputs will be set to the value of Bit1 (of this value) as soon as the PLC status is in STOP mode.
StackpoolSize	Reasonable size of the pool (bytes)	Size of the pool, from which the tasks created by external drivers get their stack during creation of the task. This pool gets allocated at system start. If a value of 512 is entered here, the external task handling will be deactivated, that means that IO drivers will not be able to create any tasks. If no value is entered, then by default 64 kB will be allocated.
CodeSize	Size of the code area (bytes)	The PLC will reserve this area twice for the code generated by CoDeSys (IEC POU's). If the entry is missing, 4MB are assumed.
DataSize	Size of the data area (bytes)	The PLC will reserve the given data area for the IEC variables, memory locations and PA. If the entry is missing, 2MB are assumed.
WdgMultiplier	0 – 0xffffffffe , 0xffffffff : disable	The number of task cycles during which the execution of a task may still continue before the watchdog is activated. 0 (default): the watchdog will be activated as soon as the configured task cycle time is exceeded. 0xffffffff: the watchdog gets deactivated
Resolution	50,100,200	Resolution of the time scale (only for PLCs with target Id 44 and 45) in µs.
InterruptFlags	0 or 1 (currently)	If this value is 1, the interrupt functions will be executed in an environment where floating point calculation is allowed.
AutoStartPLC	coded bitwise	Bit0-2 exclude each other. Bit0: The service always starts the kernel automatically. Bit1: The service never starts the kernel. Bit2: The service only will start the kernel, if the last time it has been terminated correctly. Bit3-5 exclude each other. Bit3: The PLC always loads the boot project. Bit4: The PLC never loads the boot project. Bit5: : The PLC will load the boot project, if the last time it has been terminated correctly. Bit6-8 exclude each other. Bit6: The boot project will be started always. Bit7: The boot project will never be started. Bit8: The boot project only will be started, if the last time the PLC has been terminated correctly.

In the subkey RegKeyRuntime "IO Drivers" you will find a list of IO drivers. The names of the drivers can be given in the form of <driver name> or <object name><space><driver name>. The second form is used, if the driver creates a driver object, whose name is not the same as the driver. The values must be named "Driver<running number>", whereby the running number can be 0 to 4. The values are given as strings. These strings can be used to address the drivers by the usual functions (e.g. CreateFile).

You will find more information on the IO drivers in the RTIODrv-Toolkit and in the documentation on the runtime interface.

The subkey RegKeyRuntime "External Dlls" may contain a list of DLLs, which are loaded by the service during the start of the system. The list contains entries in the following form:

Dllx <optional path><file name>

x is a running number, starting with 0. The functions in the DLLs can be called in the IEC program using the library function "DllCall".

Only those registry entries which are directly used by the runtime system are mentioned above. The complete system might of course use additional entries, because the IO drivers will create and manage own entries for their special needs.

Change History

Version	Description	Editor	Date
	Issued	AF	11.10.2000
	Update versions	AF	20.11.2000 – 16.12.2004
	changes for V2.3.4.0	AF,	05.08.2005
2.0	Start of doc version control: V2.0; up-to-date doc template; Release	MN	17.08.2005
2.1	Chap.2.1.3. Config: “NT settings” instead of “OS settings”, description of “The PLC has max __ %CPU time” added. Chap.3.12: spelling corrected.	MN	22.09.2005
2.1	Chap.3.1: Description of the wait flags for the Hilscherdriver inserted (#5399)	AF	13.10.2005
2.1	Formal Review, Release	MN	13.10.2005
2.2	According the version 2.3.5.1 of the RTE the description of the SysLibShm and the SysLibPciCards was added (#5271)	AF	1.6.2006
2.2	Description of the new config dialog added. (#6078)	AF	21.6.2006
2.2	Rework according to Review documented in German version, Release	MN	21.6.2006