



Dokumentation für OEMs:
CoDeSys SP RTE
Echtzeitlaufzeitsystem für Windows NT/2000/XP

Dokument Version 2.2


INHALT

1	ÜBERBLICK	4
1.1	Was heißt Echtzeiterweiterung?	4
1.2	Das System im Detail	5
2	DIE BEDIENUNG ÜBER DAS SERVICE-ICON	6
2.1	Das Systemmenü	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	DIE STANDARD-IO-TREIBER VON 3S	15
3.1	IO-Treiber RTIOdrvHilscherDPM	15
3.1.1	Die SysLibDPV1.lib	17
3.2	IO-Treiber RTIOdrvFC310x.sys	17
3.2.1	Die Bibliothek SysLibFCDPV1.sys	18
3.2.2	Die Bibliothek FC_SlaveHandling.lib	19
3.2.3	Die FC310x Karte im Slavebetrieb	19
3.3	IO-Treiber RTIOdrvCP5613.sys	19
3.4	IO-Treiber RTIOdrvDAMP	20
3.5	IO-Treiber RTIOdrvApplicom	20
3.6	IO-Treiber RTIOdrvIBS (IBS = Interbus, ibpcimpm.sys)	21
3.7	IO-Treiber RTIOdrvSJA und RTIOwdmPCAN	21
3.8	IO-Treiber RTIOdrvAutomata und RTIOwdmAutomata	21
3.9	IO-Treiber RTIOdrvHMS und RTIOwdmHMS	22
3.10	IO-Treiber RTIOdrvAPIC	22
3.11	IO-Treiber RTIOwdmCANAutomata	23
3.12	IO-Treiber RTIOwdmSofting	23
4	KOMMUNIKATION	24
4.1	Shared Memory Treiber	24
4.2	TCP/IP Level2 Route Treiber	24
5	SYSTEMDIAGNOSE	25
5.1	Busdiagnose der IO-Treiber	25
6	DIE TASKKONFIGURATION ZUSAMMEN MIT CODESYS 2.3	27

6.1	Allgemein	27
6.2	Taskspezifischer Watchdog	27
6.3	Microsekunden als Zeitbasis	27
6.4	Freilaufende Task	28
6.5	Systemereignisse	28
7	DIE SYSTEMBIBLIOTHEKEN	29
7.1	DllCall.lib mit SysLibSystemCall.lib	29
7.2	SysLibCallback.lib	29
7.3	SysLibCom.lib	29
7.4	SysLibFile.lib	30
7.5	SysLibPorts.lib	30
7.6	SysLibTime.lib	30
7.7	SysLibSockets.lib	31
7.8	SysLibShm.lib	31
7.9	SysLibPciCards.lib	31
8	VERHALTEN IM FEHLERFALL	32
9	ANHANG	33
9.1	Die vom Laufzeitsystem verwendeten Registry-Einträge.	33
	ÄNDERUNGSHISTORIE	37

1 Überblick

In diesem Dokument wird angenommen, dass der Leser mit dem prinzipiellen Verhalten und der Funktion eines CoDeSys-Laufzeitsystems vertraut ist. Hier wird nur auf die Besonderheiten des Laufzeitsystems zur Echtzeiterweiterung von Windows NT eingegangen.

Das Echtzeitsystem wird gestartet, indem im Windows-Startmenü aus dem CoDeSys-Menü heraus 'Start CoDeSys SP Windows NT for Realtime' gewählt wird. Das Icon  erscheint in der Taskleiste.

1.1 Was heißt Echtzeiterweiterung?

Ein Echtzeit-System ist gekennzeichnet durch ein voraussagbares (deterministisches) Zeitverhalten. Wird also etwa einem Echtzeitsystem die Aufgabe erteilt (durch Konfiguration), bestimmte Routinen innerhalb eines vorgegebenen Zeitrasters auszuführen, so wird dies innerhalb vorgegebener zeitlicher Toleranzen passieren, ansonsten wird das als Versagen des Gesamtsystems angesehen.

Auf eine Steuerung, im Sinne von CoDeSys bezogen, heißt das, dass eine Task innerhalb gegebener (vorher bekannter) Toleranzen aufgerufen wird.

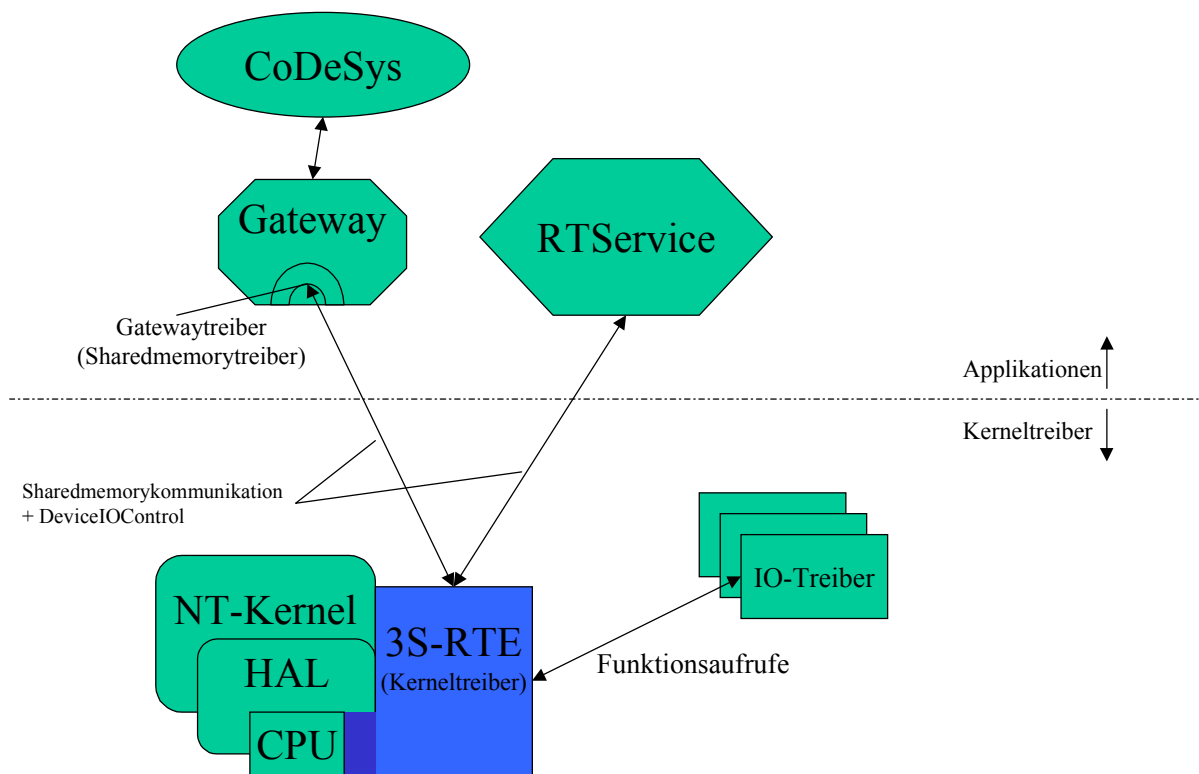
Die Erfahrung hat gezeigt, dass diese Grenzen von Windows NT nicht eingehalten werden, NT also kein echtes Echtzeitbetriebssystem ist.

Die Hardware in einem PC ist aber so beschaffen, dass es mit Mitteln der Software möglich ist, ein verlässliches Task-Scheduling zustande zu bringen.

Die Echtzeiterweiterung von Windows NT ist ein NT-Treiber, der eine Interrupt-Service-Routine installiert, die zyklisch vom Timertick der PC-Hardware aufgerufen wird.

Diese Routine übernimmt nun die Aufgabe, von CoDeSys definierte Tasks anzusprechen und/oder die Ausführung des Betriebssystems weiter/wieder zuzulassen.

1.2 Das System im Detail



Der Kernel der Echtzeiterweiterung (kurz RTE) von 3S besteht aus einem Applikationsteil und einem Kernelmode-Treiber. In der Standardausführung wird der Timerchip (über den jeder AT-kompatible PC verfügt), dazu benutzt, 2 Interrupts pro ms zu erzeugen und damit den Task-Scheduler der RTE aufzurufen. Dieser benutzt jeden 2-ten Interrupt dazu, eigene Tasks aufzurufen und jeden 2-ten um das Betriebssystem aufzurufen. Die Tasks der SPS werden also einmal pro Millisekunde vom Betriebssystem unterbrochen, wobei die Aufteilung konfigurierbar ist (siehe Kap. 6).

Die IO-Treiber werden per Konfiguration in eine Liste eingetragen und müssen einer Schnittstellenkonvention genügen (siehe dazu RTIOdrv-Toolkit). Sie tauschen mit der RTE beim Starten Funktionszeigertabellen aus und können dann mit der RTE kommunizieren.

Es ist nicht notwendig, den Betriebssystem-Kernel zu verändern.

2 Die Bedienung über das Service-Icon

Hier das Kontextmenü (rechte Maustaste, linke Maustaste auf das Icon oder die Tastenkombination Ctrl-Shift-Alt-P) des RTE-Icons in der Taskleiste:



Die jeweils sinnvollen Bedienungsmöglichkeiten sind schwarz (enabled) dargestellt, je nach aktuellem Betriebszustand.

Hier die Bedienungsmöglichkeiten im einzelnen:

Mit **Start System** wird der Treiber geladen und aktiviert. Erst dann ist eine Bearbeitung von Echtzeittasks und eine Kommunikation mit dem Programmiersystem möglich.

Mit **Stop System** wird der Treiber gestoppt und aus dem PC-Speicher entfernt.

Mit **Start/Stop Cycle** wird die zyklische Bearbeitung von IEC-Tasks gestartet / gestoppt. Diese Bedienung entspricht den Menüpunkten Start/Stop im Online-Menü von CoDeSys.

Es folgen die drei Rücksetz-(Reset-)arten, die ebenfalls den entsprechenden Menüeinträgen in CoDeSys entsprechen.

Mit **About** wird der Splashscreen der RTE angezeigt. Hier sieht man außer dem Icon die Versions- und Copyright-Information.

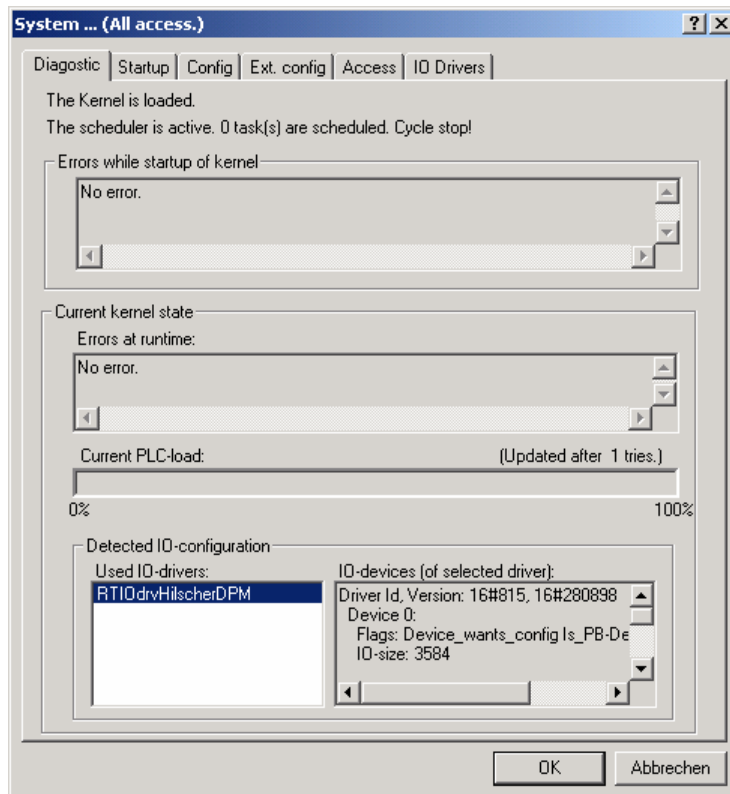
Mit **System** wird ein Dialog mit Systemeinstellungen angezeigt. Die Diagnose- und Konfigurationsmöglichkeiten sind unten näher erläutert.

Mit **Exit** wird nicht nur der Treiber gestoppt, sondern auch der Service beendet sich, nachdem der Treiber entladen wurde. Der Service kann über „Start – Einstellungen – Systemsteuerung – Dienste“ oder über den Eintrag unter CoDeSys in „Start – Programme“ wieder gestartet werden. Den Treiber über „Systemsteuerung – Geräte“ zu starten ist zwar möglich, macht aber keinen Sinn, da der Treiber dann inaktiv ist. Der RTE-Service aktiviert ihn beim Starten.

Mit **License** kann das System lizenziert werden, dieser Punkt ist nur in neueren Versionen vorhanden, bei denen so der Hardwarekopierschutz (Dongle) ersetzt wurde.

2.1 Das Systemmenü

Nach Auswahl des Menüpunktes „System ...“ erscheint der folgende Dialog:



Im folgenden werden die einzelnen Kacheln beschrieben.

Der Dialog muss mit OK verlassen werden, bevor die Einstellungen wirksam werden. Die Konfigurationseinstellungen werden erst mit dem nächsten Neustart der SPS übernommen.

2.1.1 Diagnostic

Die oberen beiden Textzeilen beschreiben den aktuellen Zustand des Systems.

In „**Errors while startup of kernel**“ werden evtl. auftretende Initialisierungsfehler eingetragen. Mögliche Einträge zur Zeit sind:

- "No error." : Keine Fehler beim Starten des Systems aufgetreten.
- "No memory available for CodeArea0." : Es konnte keine CodeArea0 erzeugt werden. Das System verfügt evtl. über zu wenig RAM, um die gewünschte Speichermenge anzulegen. → Abhilfe: weniger Codespeicher konfigurieren, siehe unten.
- "No memory available for CodeArea1." : Wie CodeArea0.
- "No memory available for DataArea.": Wie CodeArea0, → weniger Datenspeicher konfigurieren, siehe unten.
- "No memory available for retaindata." : Wie CodeArea0, → weniger Retain-Speicher konfigurieren.
- "Communication Interface could not initialize." : Zu wenig RAM für Kommunikationsspeicher vorhanden. → Keine Abhilfe möglich, mehr Speicher zur Verfügung stellen.
- "Checksumerror when loading bootproject." : Das Bootprojekt "Default.prg" ist korrupt. → Bootprojekt neu erzeugen.
- "Bootproject too big." : Das Bootprojekt wurde für eine Konfiguration mit mehr Codespeicher erstellt. → Bootprojekt neu erzeugen.
- "Datasize of bootproject too big." : Wie vorheriger Fehler.
- "Could not relocate bootproject." : Das Bootprojekt ist evtl. zerstört.

- "Error reading bootprojectfile." : Das Bootprojekt ist evtl. zerstört oder geöffnet.
- "Error reading taskconfig from bootproject." : Das Bootprojekt ist evtl. zerstört.
- "Error reading IOconfig from bootproject." : Das Bootprojekt ist evtl. zerstört
- "No IO-driver found but specified." : Ein konfigurierter IO-Treiber ist nicht vorhanden oder konnte vom System nicht geladen werden. Den Treiber installieren.
- „An IO-driver failed in his initfunction." : Ein IO-Treiber stellte einen Fehler in seiner Initialisierungsroutine fest, obwohl er bereits geladen wurde. Die IO-Konfiguration muss überprüft werden, gegebenenfalls den Hersteller des IO-Treibers kontaktieren. (Die IO-Treiber von 3S sind unten beschrieben.)
- „An IO-driver must handle at least 1 device." : Ein IO-Treiber hat gemeldet, dass er kein Gerät gefunden hat, um damit zu arbeiten. IO-Konfiguration überprüfen.
- „The PC-timer could not be calibrated." : Der PC lieferte keine periodischen Interrupts auf der erwarteten Adresse. Es ist möglich dass es sich um einen nicht AT-kompatiblen PC handelt. Setzen Sie sich mit 3S in Verbindung oder führen Sie das Setup erneut aus.
- „The PC-timertick is inexact." : Wenn die PLC beim starten feststellt, dass in mehr als 10 Messungen keine übereinstimmenden Werte gefunden werden konnten, verweigert sie den Systemstart. Der PC verfügt über keine genaue Zeitbasis.

Errors at runtime

Hier werden Fehler angezeigt, die bei der Bearbeitung des SPS-Programms aufgetreten sind. Die Fehler, die hier auftreten können, entsprechen den Laufzeitfehlern, die jedes, mit CoDeSys programmierbare Laufzeitsystem anzeigen kann:

- Watchdog: Eine Task hat ihre Zykluszeit um mehr als den eingestellten Watchdog-Faktor überschritten. Programmierfehler oder Konfiguration ändern.
- Feldbusfehler : Bei der Konfiguration einer IO-Karte ist ein Fehler aufgetreten.
- IO-updateerror: Beim Lesen oder Schreiben von Ein-/Ausgängen ist ein Fehler aufgetreten. Z.B. haben Sie versucht, auf eine IO-Adresse zu schreiben / von einer Adresse zu lesen, zu der es keinen Treiber gibt. Ist nur möglich, wenn in den Zielsystemeinstellungen von CoDeSys die Adressprüfung ausgeschaltet ist.
- Illegal instruction: Der Prozessor ist auf ungültigen Code gestoßen. Kann nur passieren, wenn eine Datei fehlerhaft ist oder Programmteile mit Daten überschrieben wurden.
- Accessviolation: Speicherzugriffsfehler, die Anwendung hat versucht, auf eine ungültige Speicheradresse zuzugreifen. Evtl. Programmierfehler beseitigen.
- Die FPU-Fehler „Division by zero“, „Inexact result“, „Invalid operation“, „Denormal operand“ und „Overflow“ werden angezeigt. Bei allen diesen Fehlern handelt es sich normalerweise um Programmierfehler.

Alle diese Laufzeitfehlermeldungen (außer IO-updateerror) können in CoDeSys, im Onlinebetrieb, mit der Funktion „Aufrufhierarchie“ lokalisiert werden.

Current PLC load

Hier wird die momentane Auslastung der SPS angezeigt. Wenn die Auslastung zu hoch wird, oder unmotiviert Watchdog-Fehler auftreten, kann die Zeitscheibe der SPS vergrößert werden. Siehe dazu unten „Config“. Die Auslastung bezieht sich immer auf die Gesamtzeit der SPS, nicht auf die gesamte Rechnerleistung.

Detected IO Configuration

Hier werden links die konfigurierten Treiber und rechts die von den Treibern gefundenen Geräte angezeigt. (Um die Gerätekonfiguration eines Treibers zu sehen, einfach den Treiber markieren.) Dabei wird für jeden Treiber eine Versionskennung und eine Id angezeigt und mit 0 beginnend seine Geräte aufgelistet.

2.1.2 Startup

Hier kann das Startverhalten der SPS konfiguriert werden.

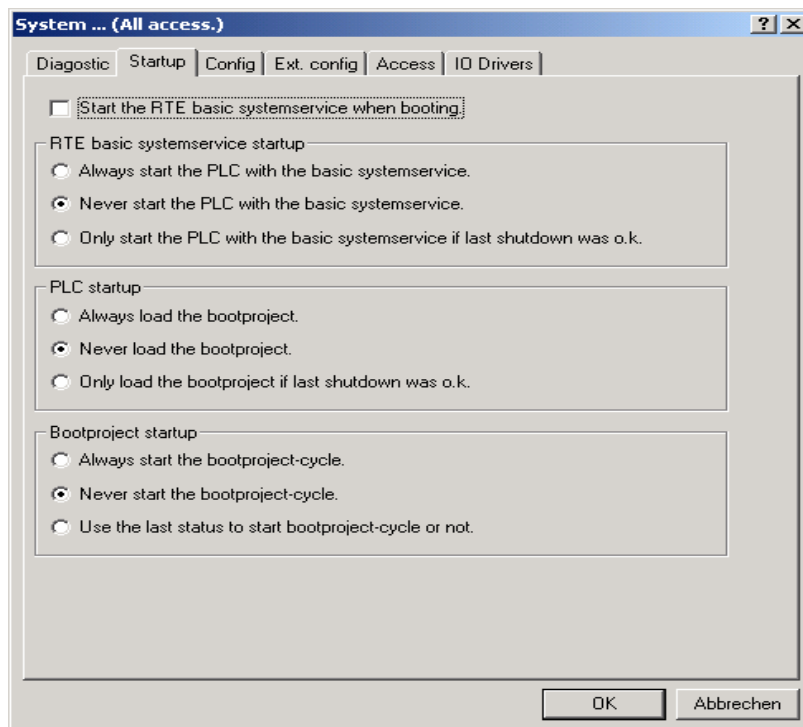
Die Konfigurationsmöglichkeiten im einzelnen:

Start the RTE basic systemservice when booting: Der Dienst wird auf "Automatisch starten" gesetzt und wird damit vom System beim Hochlaufen gestartet.

RTE basic systemservice startup: Hier wird das Startverhalten des Systemdienstes konfiguriert. Es kann festgelegt werden, ob der Dienst die SPS immer, nie oder nur wenn die SPS beim letzten Mal korrekt beendet wurde, startet.

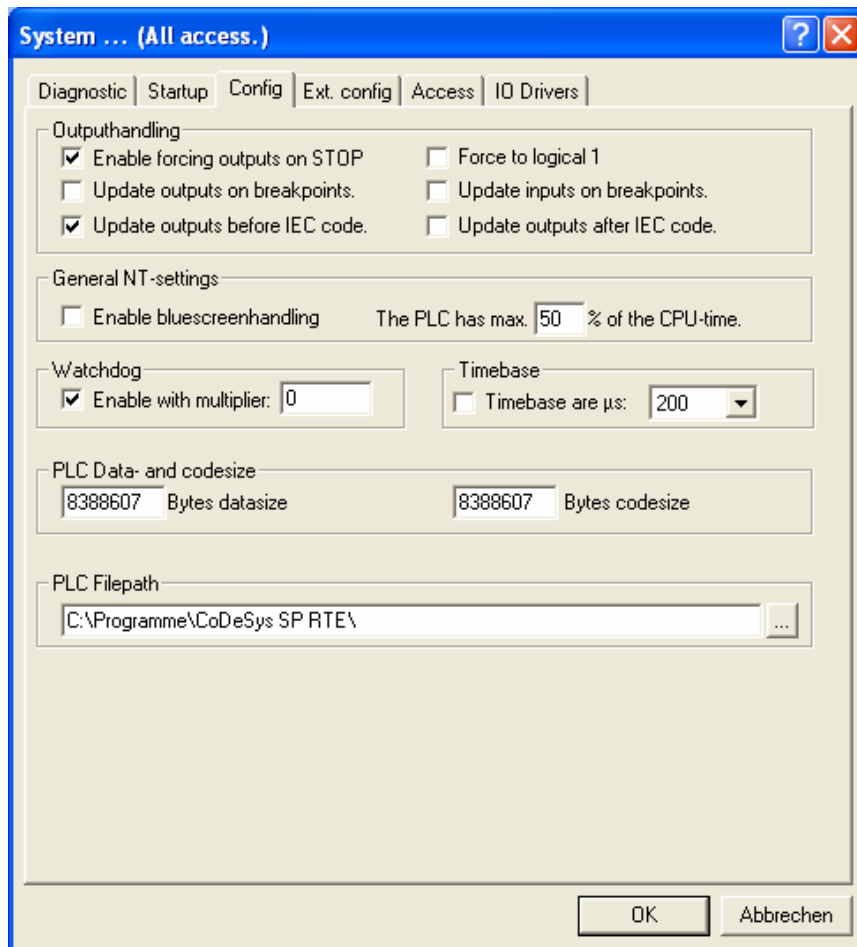
PLC startup: Das Startverhalten der SPS wird hier konfiguriert. Es kann festgelegt werden, ob die SPS ihr Bootprojekt immer, nie oder nur wenn die SPS beim letzten Mal korrekt beendet wurde, lädt.

Bootproject startup: Hier wird das Startverhalten des Bootprojekts festgelegt. Es kann festgelegt werden, ob die SPS ihr Bootprojekt immer, nie oder nur wenn die SPS beim letzten Mal korrekt beendet wurde, startet.



2.1.3 Config

Hier können einige Grundeinstellungen festgelegt werden.



Outputhandling: Wenn die Option **Enable forcing outputs on STOP** aktiviert ist, werden die Ausgänge bei Zyklus-Stop einmal auf 0 gesetzt. (Außer die Option **Force to logical 1** ist ebenfalls aktiviert, dann werden die Ausgänge auf 1 gesetzt.) Mit den Optionen **Update outputs/inputs on breakpoints** wird festgelegt, ob der IO-Update einschließlich evtl. anzustoßender Buszyklen auch dann erfolgt, wenn die Task, die das normalerweise erledigt, auf einem Breakpoint angehalten ist. Normalerweise werden die Ein- und Ausgänge immer vor der IEC-Task aktualisiert. Durch die Optionen **Update outputs before/after IEC code** kann man dieses Verhalten verändern. Wenn z.B. die Task, die als Update-Task definiert ist, eine langsame Zykluszeit oder eine Ereignis-Task ist, kann es sinnvoll sein, auch nach dem IEC-Code die Ausgänge zu aktualisieren.

General NT-settings: Hier werden die beiden Parameter, die auch das Betriebssystem beeinflussen können, festgelegt. Wenn „Enable bluescreenhandling“ aktiviert ist, dann ist das Betriebssystem nicht mehr in der Lage, bei einem schwerwiegenden Systemfehler den so genannten „bluescreen“ (Systemstop mit Ausgabe einiger Fehlermeldungen) anzuzeigen, sondern das System wird dann (für den Benutzer) eingefroren, die SPS läuft jedoch weiter. (Alle Funktionen, die ein funktionierendes Betriebssystem voraussetzen, wie Datei oder Netzwerkzugriffe, können natürlich dann nicht mehr ausgeführt werden.) Weiterhin wird der Prozentsatz der CPU-Zeit festgelegt, der der SPS maximal zur Verfügung steht.

Watchdog : Hier kann die Zeitüberwachung der Tasks ein- und ausgeschaltet werden und mit einem Multiplikator versehen werden. Der Multiplikator gibt an, um das wie vielfache der konfigurierten Zykluszeit die Laufzeitüberwachung ein Zeitüberschreitung toleriert, bevor ein Fehler ausgelöst wird (Watchdog).

Timebase : Bei Bedarf kann die Zeitbasis (Grundeinstellung 1ms) umgestellt werden. Damit erreicht man höhere Auflösung und, ab der Version 2.3 von CoDeSys, sind dann auch Zykluszeiten kleiner als 1ms möglich. (In den Version vor 2.3.1 sind auch Zykluszeiten < 1ms möglich, aber es ist zu beachten, dass die in der Taskkonfiguration angegebenen Zeiten nicht mehr in ms, sondern in µs -

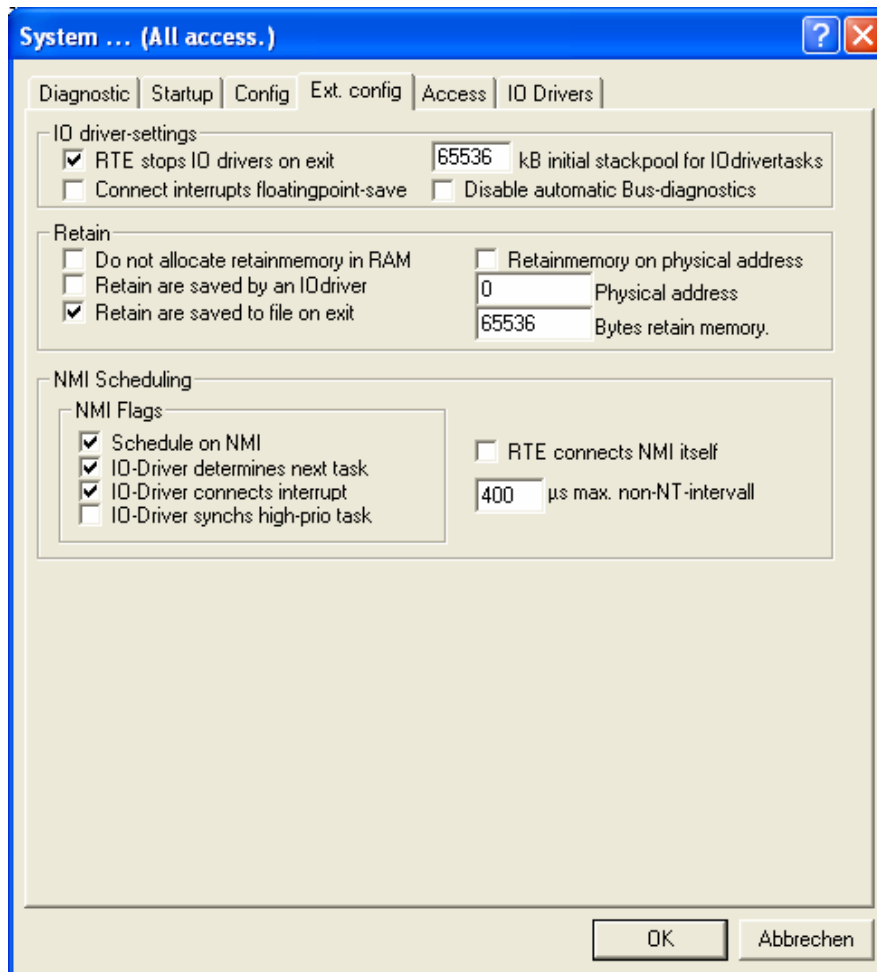
auch wenn T#1ms, also eine Millisekunden-Angabe, in der Taskkonfiguration steht - ausgewertet werden.)

PLC data- and codesize : Hier kann die Größe des Code- und Datenspeichers eingestellt werden, den die SPS beim Starten anlegt. Diese Größen müssen mit den in „Zielsystemeinstellungen -> Speicheraufteilung“ eingestellten übereinstimmen.

„**PLC Filepath**“: Hier speichert die SPS ihre Dateien, wie Bootprojekt, Retain-Daten usw. Der Pfad muss existieren und beschreibbar sein.

2.1.4 Ext. config

Hier können erweiterte Einstellungen gemacht werden. Diese Einstellungen sind systemspezifisch und sollten nicht ohne genaue Kenntnis der PC-Hardware verändert werden.



IO driversettings:

- Die IO-Treiber werden vom RTE-Kernel beim Beenden der SPS nur beendet, wenn die Option **RTE Stops IOdrivers on exit** aktiviert ist.
- Sollen Hardware-Interrupts mit Fließkommaunterstützung eingehängt werden, so muss die Option **Connect interrupts floatingpoint-save** aktiviert werden. Diese Option sollte nur benutzt werden, wenn im Interrupthandler wirklich Fließkommaoperationen gebraucht werden, da der resultierende Performanceverlust sonst sinnlos ist. Umgekehrt verursacht ein Interrupthandler, der Fließkommaoperationen ohne die aktivierte Option ausführt, sporadische Systemabstürze.
- Mit **Initial stackpool for IO drivers** wird die Größe des gesamten Stacks für Task, die von IO-Treibern (nicht CoDeSys) angelegt werden, angegeben.

- Mit der Option „**Disable automatic bus-diagnostics**“ wird verhindert, dass Treiber für IO-Karten, die mit CoDeSys konfiguriert wurden, im Hintergrund die Diagnosedaten in die in der Steuerungskonfiguration angegebene Merkeradressen schreiben.

Retain:

- **Do not allocate retainmemory in RAM** zusammen mit **Retainmemory on physical address** führt dazu, dass Variablen, die das Flag “RETAIN” in CoDeSys besitzen, direkt im statischen RAM angelegt werden. Dazu muss die PC-Hardware über einen SRAM-Bereich (**Bytes retain memory**) auf einer fixen physikalischen Busadresse (**Physical address**) verfügen.
- **Retain are saved by IOdriver** wird normalerweise nicht benutzt und dient nur dazu, spezifischen Treibern die Möglichkeit zu geben, SRAM selbst zu verwalten.
- Die Einstellung **Retain are saved to file on exit** ist die Einstellung, für die es keiner speziellen Hardware bedarf. Allerdings setzt diese Einstellung voraus, dass die SPS immer richtig beendet wird, bzw. der PC korrekt heruntergefahren wird. Soll dieses sichergestellt werden, speziell in der Industrie, ist der Einsatz von USVs unvermeidlich. (USV = unterbrechungsfreie Stromversorgung)

„NMI Scheduling“:

- Mit Option **Schedule on NMI** wird die SPS veranlasst, keinen Interrupthandler für den Timerchip des PCs zu installieren. Es wird dann vorausgesetzt, dass ein IO-Treiber vorhanden ist, der in der Lage ist, selbst zyklische Interrupts zu erzeugen oder wenigstens zu quittieren.
- Mit der Option **RTE connects NMI itself** wird die SPS dazu veranlasst, selbst einen Interrupthandler auf den NMI-Vektor zu setzen.
- Ist die Option **IO driver determines next task** aktiviert, bestimmt der IO-Treiber, der den NMI aktiviert und quittiert, was der Scheduler nach jedem Interrupt tun soll. Kann er das nicht, bestimmt die SPS selbst, wann das Betriebssystem und wann die SPS-tasks aufgerufen werden sollen. Dazu benutzt die SPS den Zahlenwert aus **max. non-NT-Intervall**, um bei asymmetrischer Zeitverteilung festzustellen, welches Intervall für das Betriebssystem und welches für SPS-task benutzt werden soll. Wenn möglich sollte diese Methode nicht benutzt werden, da sie asymmetrische Interrupt-Erzeugung voraussetzt und unflexibel ist.
- Wenn ein IO-Treiber selbst einen Interrupt zum Scheduling bestimmen möchte, muss **IO driver connects interrupt** aktiviert werden.
- Die Option **IO driver synchs high-prio-task** ermöglicht einem IO-Treiber (der, der auch den Scheduler-Interrupt bestimmt) die höchstprioritäre Task nur zu einem von ihm bestimmten Zeitpunkt aufzurufen.

2.1.5 Access

Hier können für 5 verschiedene Stufen Passwörter eingerichtet werden, um die einzelnen Bedienungsmöglichkeiten vor unbefugter oder unbeabsichtigter Benutzung zu schützen. Das System startet immer im zahlenmäßig größten, d.h. mit dem Zugriffslevel, der über die wenigsten Berechtigungen verfügt, und für den ein Passwort konfiguriert ist. Nur im Level 1 können Passwörter geändert werden. Die höher berechtigten Passwortlevel haben immer alle Zugriffsrechte der weniger berechtigten und jeweils das Recht, für das sie zuständig sind.

Wenn auf ein bestimmtes Level eingeloggt werden soll, einfach den Schalter **Change Level** benutzen und das zugehörige Passwort eingeben. Der Level bleibt erhalten, bis der Benutzer sich mit **Logout** ausloggt.

Für die Zugriffsrechte ergibt sich folgende Tabelle (senkrecht Passwortlevel, waagrecht Rechte):

Level	Start/Stop Zyklus	Start/Stop/Exit System	Startverhalten /Konfiguration	Erweiterte Konfiguration/ IO-Treiberkonfig.	Passwörter ändern
1	x	x	x	x	x
2	x	x	x	x	-
3	x	x	x	-	-
4	x	x	-	-	-
5	x	-	-	-	-

System ... (All access.)

Diagnostic Startup Config Ext. config Access IO Drivers

Password Level 1: Edit passwords.
Confirm password:

Password Level 2: Configure the system.
Confirm password:

Password Level 3: Configure the startup behaviour.
Confirm password:

Password Level 4: Start/Stop/Exit system. Reset the PLC.
Confirm password:

Password Level 5: Start/Stop PLC-cycle.
Confirm password:

Current accesslevel is: All access.

Change Level Logout

OK Abbrechen

In **Password Level <n>** muss jeweils ein Passwort und das identische in **Confirm password** eingetragen werden.

2.1.6 IO Drivers

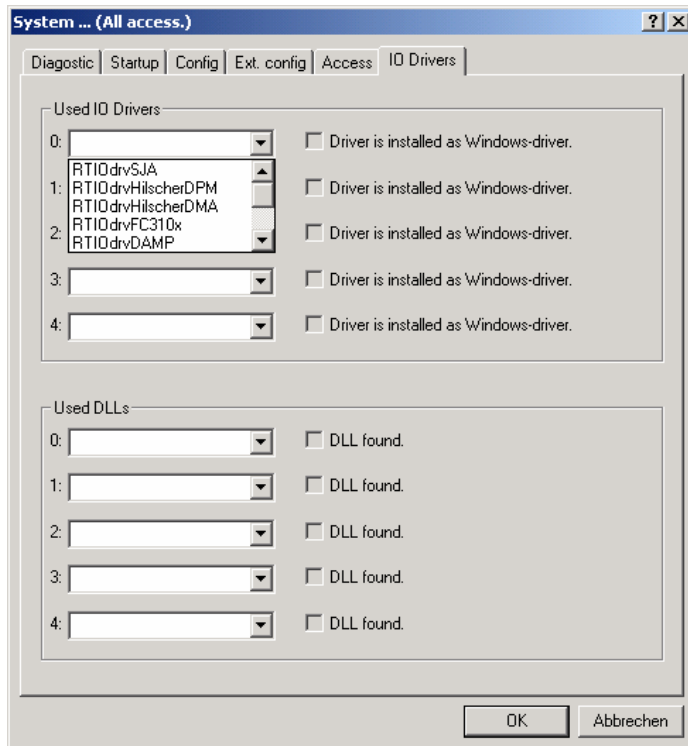
Hier können IO-Treiber (**Used IO Drivers**) und kundenspezifische DLLs (**Used DLLs**) eingehängt werden. Die Einträge können manuell eingetragen oder aus der Liste ausgewählt werden. Welche Einträge in der Liste erscheinen wird normalerweise vom Setup konfiguriert.

Die Liste findet sich in der Registry unter \HKEY_LOCAL_MACHINE\SOFTWARE\3S Smart Software Solutions GmbH\CoDeSys SP\RTPLC\IO Driverpool bzw. External Dll Pool wieder. Die Namen der Unterschlüssel dieser beiden Schlüssel sind die Listeneinträge die angezeigt werden.

Die Treiber und die DLLs müssen jeweils bei 0 beginnend aufsteigend konfiguriert werden.

Wird ein Treiber eingetragen, zeigt das graue Häkchen an, ob der Treiber bereits als Windowstreiber in der Registrierung vorhanden ist (**Driver is installed as Windows-driver**). Wenn nicht, muss er installiert werden, sonst erscheint beim Starten des System eine Fehlermeldung.

Achtung (für Entwickler von IO-Treibern): Wenn das Objekt, das ein Treiber anlegt, genauso heißt wie die .sys-Datei (ohne Endung), dann genügt hier ein Wort, sonst muss der Treiber mit 2 Worten, durch Leerzeichen getrennt, eingetragen werden: <Objektnamen> <Treibername>.



Die Zuordnung der IO-Adressen aus dem SPS-Programm erfolgt entweder über die Steuerungskonfiguration von CoDeSys (nur mit CAN und Profibus) oder über die Adressen selbst:

Jeder Treiber belegt im Adressraum eine bestimmte Größe. Diese ist in **Diagnostic** (Size eines Devices, siehe oben) ablesbar. In der Reihenfolge, in der die Devices (Geräte) hier auftauchen, ist ihr Adressraum aneinandergehängt.

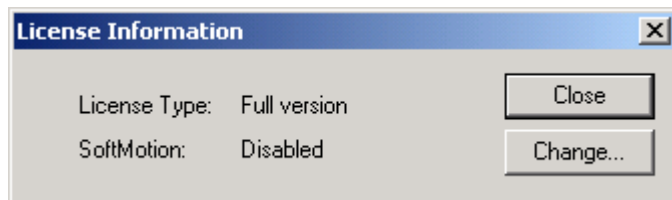
Beispiel:

Im Rechner stecken 2 Hilscher-Karten, eine (Device0) belegt 512 Bytes, die andere 3584. Beide wurden mit dem SyCon konfiguriert, alle Module sind im SyCon lückenlos von 0 beginnend adressiert.

Dann beginnt in CoDeSys die erste Karte bei %QB0 bzw %IB0, die zweite Karte bei %QB512 bzw. %IB512.

2.2 License ...

Um Ihre Version der CoDeSys SP RTE zu lizenzieren, wählen Sie diesen Menüpunkt. Es erscheint ein Dialog, in dem die bestehenden Lizenzen angezeigt werden:



Mit **Change...** können diese Lizenzen geändert bzw. erweitert werden. Dazu erscheint ein Wizard, der Sie durch den Lizenzierungsvorgang führt. Die Lizenzierung kann wahlweise über Telefon oder via e-mail erfolgen.

3 Die Standard-IO-Treiber von 3S

(Der RTIOdrvAPIC wird vom Setup immer dann installiert, wenn detektiert wurde, dass es sich nicht um einen AT-kompatiblen PC handelt, sondern die Hardware nach der UP-MPS-Spezifikation funktioniert. Der Treiber kann weder konfiguriert noch darf er entfernt werden.)

3.1 IO-Treiber RTIOdrvHilscherDPM

Der IOTreiber RTIOdrvHilscherDPM benutzt die Werte in seinem Treibereintrag (HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RTIOdrvHilscherDPM\Params) Device 0 bis 4, um seine Karten zu finden bzw. auszugeben, welche Karten er gefunden hat..

Gültige Einträge sind:

	Mögliche Werte	Kommentar
Device0..4\Type (String)	8K_DPM oder 2K_DPM	Die Hilscher-CIF/COM-Karten gibt es mit 2K(1k IO-Daten) und 8K(7k IO-Daten).
Device0..4\Address (DWORD)	32-bit-Wert	Die physikalische Busadresse der Karte.
CreateComTask	0 oder 1, der Defaultwert ist 1	Ist dieser Wert vorhanden und 1 wird eine Kommunikationstask, die die Hilscher-Karten dauernd nach Kommunikationsnachrichten durchsucht, angelegt. Bei 0 oder fehlendem Eintrag wird diese Task nicht erzeugt.
Device0..4\DoNotConfigure	0 oder 1	Dieser Wert dient dazu, damit einzelne Hilscher-Karten von CoDeSys in der Konfiguration übergangen werden. (Da die Konfiguration der FMS-Kommunikation der Hilscher-Karte nur durch den SyCon möglich ist, darf die Karte, über die kommuniziert werden soll, nicht über
Detected	Stringwert „Not found“ oder „Card found: <Informationen über die Karte>“	Dieser Eintrag dient zur Kontrolle, was der Treiber für Karten detektiert.
in Params: AutoDetection	0 oder 1	Mit diesem Schalter kann die automatische Suche nach Hilscher-Karten verhindert werden. Dann muss die Karte, die benutzt werden soll, manuell über Address/Type eingetragen werden.
In Params: UseInterrupt	BitCodiert	Bit 0 = 1: Der Treiber verwendet für eine PCI-PB-Masterkarte die Interrupt-Steuerung der Karte. Bit 1 = 1: Der Treiber verwendet für die Interrupt-Steuerung den 2. Interrupt der Hilscher-Karte. Dies ist abhängig von der Hardware. Normalerweise, für original Hilscher-Karten ist

	Mögliche Werte	Kommentar
		dieses Bit 0. Bit 2 = 1: Ist dieses Bit = 1, dann wird die Interrupt-Polarität negiert, das ist abhängig von der Anschaltung der Hilscher-Karte. Muss nur benutzt werden, wenn die Karte von einem Hardwarehersteller als Modul eingesetzt wird.
In Params: WaitFlags	Bit codiert (exklusiv, nur ein Bit setzen!)	Bit0: Mit diesem Flag wird dem Treiber mitgeteilt, dass vor dem Update der IOs solange gewartet werden soll, bis der vorhergehende Updatezyklus fertig ist. Bit1: Mit diesem Bit wird dem Treiber mitgeteilt, dass der IOUpdate immer dann übersprungen werden soll, wenn der vorhergehende Profibus-Zyklus noch nicht abgeschlossen war.

Bei Verwendung der PCI-Bus-Varianten und manueller Konfiguration (AutoDetection = 0) ist darauf zu achten, dass sich beim Einbau neuer PCI-Geräte die Adresse der Karte verändern kann.

Der Hilscher-Kartentreiber arbeitet mit allen Hilscher-Karten, auch solchen, die nicht von CoDeSys konfiguriert werden können. Trotzdem kann mit dem Konfigurator (z.B. Interbus) von Hilscher als Prozessdatenübergabeverfahren „anwendergesteuert, gepuffert“ benutzt werden. So vermeidet man innerhalb einer Task Dateninkonsistenzen.

Der RTIOdrvHilscherDPM sucht normalerweise seine Karten selbständig, außer in

(HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RTIOdrvHilscherDPM\Params)

gibt es den Wert „AutoDetection“, der 0 sein muss, um die automatische Kartenerkennung zu verhindern.

In jeden „Device“-Schlüssel unterhalb von

KEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RTIOdrvHilscherDPM\Params

trägt die automatische Erkennung ein, welche Karten gefunden wurden.

Der Zugriff auf das Prozessabbild der Karte erfolgt dann Zyklus-(Task-)synchron, wenn im SyCon der „anwendergesteuerte, gepufferte“ Zugriff eingestellt wurde. Der Treiber führt mit allen Hilscher-Karten ein Handshake über das PdCom-Flag durch.

Bei der Verwendung von Profibus- oder CANopen-Karten kann der Steuerungskonfigurator von CoDeSys benutzt werden. Hier können auf Kanalebene beliebige IO-Adressen vergeben werden, durch die Zuordnung in der Steuerungskonfiguration weiß das System, welche Karte angesprochen werden soll.

Zusammen mit der neuesten Version des Treibers kann die HilscherPBInfo.lib benutzt werden, wenn die Interrupts aktiviert sind. Damit kann die aktuelle PB-Zykluszeit ermittelt werden.

3.1.1 Die SysLibDPV1.lib

Die Bibliothek SysLibDPV1.lib kann zusammen mit dem RTIOdrvHilscherDPM und einer entsprechenden PB-Karte verwendet werden. Die Bibliothek enthält 2 Bausteine:

DPV1_Read

DPV1_Write

Mit diesen Bausteinen können azyklische Datenübertragungsdienste der Klasse 1 benutzt werden (MSAC_C1). Die Bausteine haben beide die Eingangsparameter:

ENABLE:BOOL;	Mit diesem Eingang wird der Baustein aktiviert, ein Auftrag gestartet.
Device:INT;	Mit Device wird die Karte ausgewählt. Bei nur einer FC310x im System immer 0.
StationAddr:INT;	Die Stationsadresse des angesprochenen Slaves.
Slot:INT;	Slot und Index sind in der Definition der DPV1-Dienste beschrieben und haben je nach Slave unterschiedliche Bedeutungen.
Index:INT;	
Len:INT;	Die Anzahl zu schreibender Bytes (bei WRITE), bzw. die Größe des lokalen Lesepuffers (bei READ).
buffer:DWORD;	Die Adresse des Lesepuffers. Sie muss mit ADR() ermittelt werden. Typischerweise die Adresse eines ARRAY OF BYTE.

Beim Aufruf der FBs muss einmal am ENABLE-Eingang eine steigende Flanke angelegt werden (Wechsel von FALSE auf TRUE) und dann müssen die Bausteine mit ENABLE gleich TRUE solange zyklisch aufgerufen werden, bis der READY-Ausgang TRUE meldet.

Mit dem Ausgang STATE des Bausteins kann die erfolgreiche Ausführung überprüft werden.

An seinem Size-Ausgang zeigt der Baustein an, wie viele Bytes wirklich geschrieben/gelesen werden konnten.

Bei Ausführung des Lesebausteins wird im Fehlerfall in das erste Byte von Buffer der Fehlercode eingetragen.

Dieser Fehlercode entspricht dem Fehlercode der Hilscher-Karte und kann in der Dokumentation der Karte nachgeschlagen werden.

Bei der Verwendung der Hilscher-Karten kann über die mitgelieferte Hilscher.lib direkt die Messsagebox-Schnittstelle (näheres dazu findet sich in der Dokumentation der Fa. Hilscher zu den einzelnen Karten) angesprochen werden.

Bei der Verwendung der CANopen-Karten der Fa. Hilscher kann aufbauend darauf die Hilscher_SDOtransfer.lib benutzt werden, um mit CANopen-Nodes eine SDO-Kommunikation aufzubauen. Die Bibliothek ist ebenfalls im Setup enthalten.

3.2 IO-Treiber RTIOdrvFC310x.sys

Der RTIOdrvFC310x arbeitet mit den Profibusmasterkarten FC3101 und FC3102 der Fa. Beckhoff.

Die Karten werden automatisch erkannt und können nur mit dem Steuerungskonfigurator von CoDeSys konfiguriert werden.

In der 2-Kanal-Version der Karte (FC3102) werden 2 Geräte (Devices) erkannt.

Der Treiber liest den Softwarestand der Firmware von der Karte aus, alle Funktionen können aber nur genutzt werden, wenn die Karte mindestens über die Firmwareversion 2.0 verfügt. (Ab ca. August 2002).

Der Treiber benutzt einen Konfigurationseintrag in der Registry:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvFC310x\Params beinhaltet bei Bedarf den folgenden Eintrag: Debug, bitweise codiert. (Fehlt der Wert, wird 0 angenommen.)

Bit0: Wenn 1 werden beim Beenden des Treibers (also beim Beenden der RTE) in diesem Registry-Key Fehlerstatistiken über jeden konfigurierten Slave gespeichert. Damit kann über die Güte des PB-Netzes Informationen gewonnen werden.

Bit1: Wenn 1 wird der task-synchrone Modus der Karten nicht benutzt. Die PB-Zyklen werden also asynchron zu den SPS-Tasks laufen.

Mit der FC310x zusammen kann die SysLibFCDPV1.lib benutzt werden, um die DPV1-Kommunikationsdienste Read und Write zu benutzen.

3.2.1 Die Bibliothek SysLibFCDPV1.sys

Die Bibliothek SysLibFCDPV1.lib kann nur zusammen mit dem RTIOdrvFC310x und einer entsprechenden PB-Karte verwendet werden. Die Bibliothek enthält 2 Bausteine:

FC_DPV1_Read

FC_DPV1_Write

Mit diesen Bausteinen können azyklische Datenübertragungsdienste der Klasse 1 benutzt werden (MSAC_C1). Die Bausteine haben beide die Eingangsparameter:

ENABLE:BOOL;	Mit diesem Eingang wird der Baustein aktiviert, ein Auftrag gestartet.
Device:INT;	Mit Device wird die Karte ausgewählt. Bei nur einer FC310x im System immer 0.
StationAddr:INT;	Die Stationsadresse des angesprochenen Slaves.
Slot:INT;	Slot und Index sind in der Definition der DPV1-Dienste beschrieben und haben je nach Slave unterschiedliche Bedeutungen.
Index:INT;	
Len:INT;	Die Anzahl zu schreibender Bytes (bei WRITE), bzw. die Größe des lokalen Lesepuffers (bei READ).
buffer:DWORD;	Die Adresse des Lesepuffers. Sie muss mit ADR() ermittelt werden. Typischerweise die Adresse eines ARRAY OF BYTE.

Beim Aufruf der FBs muss einmal am ENABLE-Eingang eine steigende Flanke angelegt werden (Wechsel von FALSE auf TRUE) und dann müssen die Bausteine mit ENABLE gleich TRUE solange zyklisch aufgerufen werden, bis der READY-Ausgang TRUE meldet.

Mit dem Ausgang STATE des Bausteins kann die erfolgreiche Ausführung überprüft werden.

Bei Ausführung des Lesebausteins werden im Fehlerfall in Buffer die einige Fehlercodes eingetragen.

Diese Fehlercodes entsprechen den Fehlercodes der FC310x und können in der Dokumentation der Karte nachgeschlagen werden.

Die Funktion FC_ClearMessageBox kann benutzt werden, um das MessageBox-Interface der FC-Karte auszuräumen. Das wird z.B. nach einem Reset der Steuerung benötigt, wenn keine gültigen Job-Ids dem Programm mehr bekannt sind. Die Funktion sollte (z.B.) wie folgt benutzt werden (aus einem zyklisch aufgerufenen Programm heraus, für die erste FC-Karte):

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

Der Rückgabewert der Funktion ist 0, wenn keine Job, der gelöscht werden hätte können vorhanden ist (MessageBox ist leer). Der Rückgabewert ist 1 (untere 16 bit des Rückgabewerts) mit der ID des gerade gelöschten Jobs in den oberen 16 bit. 16#FFFFFFFF ist der Rückgabewert im Fehlerfall.

Der Eingabeparameter der Funktion ist die Nummer der Karte, auf die die Funktion arbeiten soll.

3.2.2 Die Bibliothek FC_SlaveHandling.lib

Zusammen mit der FC310x (ab FW-Stand 2.17) kann die Systembibliothek FC_SlaveHandling.lib benutzt werden, um einzelne Slaves zur Laufzeit an- und abschalten zu können.

Der Baustein hat folgende Eingangsparameter:

ENABLE : BOOL;	Der Baustein wird mit einer steigenden Flanke aktiviert.
iSlaveAddress : BYTE;	Die Stationsadresse des Slaves.
iNewState : SETSTATE;	Der Status auf den der Slave gesetzt werden soll.

Nachdem ein Slave abgeschaltet wurde, behandelt die FC ihn nicht mehr, so als wäre er nicht konfiguriert worden. Erst wenn ein Slave wieder eingeschaltet wurde, wird er vom Master wieder angesprochen.

Nach dem Laden des Programms, also nach der Konfiguration der Karte sind alle Slaves eingeschaltet und werden angesprochen. Die Bibliothek ist nur notwendig, wenn zur Laufzeit der Applikation Slaves an- und abgeschaltet werden sollen.

3.2.3 Die FC310x Karte im Slavebetrieb

Die Karte kann (im 2 Kanalbetrieb auch gemischt) als Slave betrieben werden. Dazu muss man dem Treiber mitteilen, welcher Kanal als Slave betrieben werden soll. Das geschieht mit Hilfe von 3 Registry-Einträgen, die manuell vorgenommen werden müssen (z.B. mit Regedit):

(Die Einträge müssen im Konfigurationsschlüssel des Treibers angelegt werden: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvFC310x\Params)

Der DWORD-Wert „DeviceXIsSlave“ muss auf 1 gesetzt werden, wobei X durch den Index des Kanals ersetzt wird. (Wenn Sie also eine FC3101 im PC gesteckt haben, kommt nur der Name „Device0IsSlave“ in Frage.)

Der DWORD-Wert „SlaveXBaudRate“ muss auf die entsprechende Baudrate gesetzt werden. (Also wieder für das Beispiel mit nur einer FC3101 kommt nur der Name „Slave0BaudRate“ in Frage. Dieser wird für 12MBaud auf dezimal 12000000 gesetzt.)

Der DWORD-Wert „Debug“ muss zwingend auf 2 gesetzt werden.

ACHTUNG: Diese Werte werden nur einmal beim Starten des Treibers gelesen. Ändert man diese Werte, muss einmal über das Menu der RTE mit **Stop System** und anschließendem **Start System** das System neu gestartet werden. (Und damit auch der Treiber.)

3.3 IO-Treiber RTIOdrvCP5613.sys

Der RTIOdrvCP5613 arbeitet mit dem Profibusmaster CP5613 der Fa. Siemens.

Der Treiber benötigt eine Firmwaredatei, die mit der Karte geliefert wird. Die Datei muss in folgendem Verzeichnis vorhanden sein: Im Runtime-Verzeichnis der RTE. Die Datei heißt „FW_5613.bin“.

Das Runtime-Verzeichnis kann unter

HKEY_LOCAL_MACHINE\SOFTWARE\3S Smart Software Solutions GmbH\CoDeSys SP\RTPLC\ im Wert Path in der Registry nachgeschlagen werden.

Der Treiber wertet 2 Konfigurationsschalter aus seinem Registrierungsschlüssel aus. Der Schlüssel heißt: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvCP5613\Params

- „Debug“: Wenn ungleich 0, dann wird die LDB, die aus den CoDeSys-Konfigurationsdaten gebildet wird, nicht nur auf die Karte, sondern auch als Datei auf die Festplatte geschrieben. Die Datei wird ebenfalls im Runtime-Verzeichnis angelegt und hat den Namen „out__x.ldb“, wobei x durch den Index der Karte ersetzt wird.

- „ReadFile“: Wenn ungleich 1, dann wird die CoDeSys-Konfiguration zwar erzeugt und u.U., wenn Debug auf 1 steht, auf der Platte abgelegt, aber auf die Karte wird eine LDB-Datei geschrieben, die den Namen in.ldb tragen und sich im Runtime-Verzeichnis befinden muss.

Um mit der Karte in der RTE arbeiten zu können, muss immer in CoDeSys eine PB-Konfiguration angelegt werden, auch wenn die Konfiguration aus einem File gelesen werden soll. Der Treiber startet die Karte nur, wenn Konfigurationsdaten mit dem Programm geladen werden.

3.4 IO-Treiber RTIOdrvDAMP

Dieser Treiber arbeitet mit jeder Karte, die auf einer festen physikalischen Busadresse einen gemeinsamen Speicherbereich mit dem PC anbietet (Memory mapped dualport RAM).

Dabei muss dieser Speicherbereich in Ein- und Ausgangsbereich aufgeteilt sein und die Ein- und Ausgänge der Karte müssen über die Offsets zu den Startadressen dieser Bereiche eindeutig zugeordnet werden können.

Der Treiber muss manuell in der Registry konfiguriert werden. Dabei gibt es folgende Einträge:

Unter HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvDAMP\Params

und unterhalb von Params für jede mögliche Karte einen „DeviceX“-Schlüssel. Dabei ist X der Index der Karte, kann also von 0 bis 4 laufen.

In Params gibt es einen Wert der ausgewertet wird: Wenn Simulation“ auf 1 steht, werden keine IOs gelesen oder geschrieben, der Treiber kann zum Testen des Programms benutzt werden.

In jedem „DeviceX“ Unterschlüssel gibt es die Werte

- „Flags“: Wenn die Karte mit diesem Index vorhanden ist, muss Flags auf 7 stehen, sonst auf 0.
- „PhysAddrIn“: Die physikalische Busadresse des Eingangsdatenbereichs auf der Karte.
- „PhysAddrOut“: Die physikalische Busadresse des Ausgangsbereichs auf der Karte.
- „InLen“: Die Länge in Bytes des Eingangsdatenbereichs.
- „OutLen“: Die Länge in Bytes des Ausgangsbereichs.

Beispiel:

Es soll eine digitale IO-Karte mit 64 Bytes Ein- und Ausgangsdaten von der SPS benutzt werden.

Die Karte liegt auf dem ISA-Bus und hat die Adresse 0xC0000. Vorne im Adressbereich der Karte liegen die Ein- und dahinter die Ausgänge. Dann müssen die Konfigurationswerte folgendermaßen gesetzt werden (in Device0):

Flags = 7

InLen = 64

OutLen = 64

PhysAddrIn = C0000

PhysAddrOut = C0040

3.5 IO-Treiber RTIOdrvApplicom

Der Treiber kann mit IO-Karten der Fa.Applicom arbeiten (ApplicomIO). Dabei kann es sich um jedes Bussystem handeln, die Schnittstelle der Karten zur SPS bleibt immer gleich. Die Karten können nur mit dem, mit den Karten mitgelieferten, Konfigurationstool von Applicom konfiguriert werden.

Bei der Ermittlung der IO-Adressen der einzelnen Module muss der Programmierer alle konfigurierten Module als hintereinander liegend im Prozessabbild der SPS betrachten.

Der Treiber sucht und startet selbständig alle ApplicomIO-Karten, die sich im System befinden.

Der Treiber wertet keine Konfigurationsschalter der Registry aus.

3.6 IO-Treiber RTIOdrvIBS (IBS = Interbus, ibpcimp.sys)

Der Treiber arbeitet mit der InterbusS-Masterkarte IBS PCI SC/I-T der Fa. Phoenix Contact.

Der RTIOdrvIBS wird vom Setup immer mitinstalliert. Um eine IBS PCI SC/I-T unter Windows 2000 oder XP zu nutzen, muss aus dem Verzeichnis w2k_Xp Drivers der Setup-CD die Karte mit der entsprechenden .inf-Datei installiert werden. Ein evtl. bereits installierter Treiber der Fa. Phoenix muss vorher deinstalliert werden.

Der Treibereintrag in der Registerkarte IO Drivers des System....-Menu lautet (siehe Pull-Down-Liste dort) „lbsisasc1N1D ibpcimp“.

Achtung: Der Treiber kann (noch) keine Geräteinstanzen anlegen, d.h. er wird nur für eine Karte pro PC arbeiten.

Die Karten müssen mit dem Konfigurationstool der Fa. Phoenix konfiguriert werden (CMD G4). Danach kann die RTE mit dem InterbusS-Netzwerk, für das die Karte konfiguriert wurde, IO-Daten austauschen. Die IO-Adressen im SPS-Programm entsprechen dabei denen, die mit dem CMD-Tool eingestellt wurden.

Um den Treiber und die Karte in Betrieb zu nehmen, muss zunächst das CMD-Tool von Phoenix Contact installiert werden, um später die Konfiguration auf die Karte zu laden und im Parametrierungsspeicher der Karte abzulegen. Wenn die Karte über die serielle Schnittstelle von einem anderen PC aus konfiguriert und bedient werden soll, kann dieser Schritt übersprungen werden. Zu beachten ist allerdings, dass bei einem späteren installieren des CMD-Tools der 3S-Treiber u. U. überschrieben wird. Das 3S-System muss dann nach der CMD-Installation nochmals installiert werden (oder einfach den Treiber aus System32\drivers sichern und nach der CMD-Installation wieder dahin zurückschreiben, der Treiber heißt ibpcimp.sys).

Um den Wiederanlauf der Karte nach einem Busfehler ohne das Konfigurationstool CMD G4 zu ermöglichen, muss nach der Konfiguration und dem Starten der Karte, im CMD-Tool „Parametrierungsspeicher schreiben...“ benutzt werden, um die Konfiguration und das Anlaufverhalten auf der Karte zu speichern. Im Falle eines Busfehlers wird dann mit „Online->Reset“ (von CoDeSys aus im Onlinebetrieb) oder mit „Reset Warm/Cold/Hard“ des Bedienmenüs der SoftSPS der Bus neu gestartet.

3.7 IO-Treiber RTIOdrvSJA und RTIOwdmPCAN

Um die CAN-Karte PCAN-PCI der Fa. Peak-Systems zu nutzen, wählen Sie auf einem NT4.0-System den Eintrag RTIOdrvSJA in der IO-Treiberliste aus.

Um eine solche Karte auf einem XP/Win2000-System zu nutzen, installieren Sie zuerst die Karte und installieren mit dem Hardwareassistenten von Windows dann nicht den mitgelieferten Treiber von Peak-Systems, sondern aus dem Verzeichnis Win2K_XP\Peak Systems PCAN der Installations-CD bzw. des Installationsverzeichnisses der RTE die .inf-Datei.

3.8 IO-Treiber RTIOdrvAutomata und RTIOwdmAutomata

Um eine SERCOS-Karte der Fa. Automata zu nutzen wählen Sie auf einem NT4.0-System den Eintrag RTIOdrvAutomata aus.

Um eine solche Karte auf einem XP/Win2000-System zu nutzen, installieren Sie zuerst die Karte und dann mit dem Hardwareassistenten von Windows aus dem Verzeichnis Win2K_XP\Sercos Automata der InstallationsCD bzw. des Installationsverzeichnisses der RTE die .inf-Datei. Dann wählen Sie den Eintrag RTIOwdmAutomata aus.

Der Treiber kann nur mit einer Karte arbeiten.

Auf einem Windows NT-System kann der Treiber mit der ISA/PC104-Version der Karte benutzt werden. Dazu muss der Treiber entsprechend konfiguriert werden.

Mit den beiden Registry-Einträgen „Address“ und „Interrupt“ in
\\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RTIOdrvAutomata\Params
wird dem Treiber die Busadresse der Karte und der Interruptvektor, der benutzt werden soll, mitgeteilt.

Die PCI-Version der Karte kann ohne Konfigurationsänderungen unter Windows NT4.0 mit dem RTIOdrvAutomata und unter Win2K/XP mit dem RTIOwdmAutomata benutzt werden.

Unter Windows XP und 2K kann im Moment nur die PCI-Karte benutzt werden.

3.9 IO-Treiber RTIOdrvHMS und RTIOwdmHMS

Mit dem RTIOdrvHMS können die Profibus-PCI-Karten der Fa. HMS benutzt werden. Die Karte kann in der Master- und in der Slavevariante benutzt werden.

Der Treiber erkennt seine Karten selbst (er kann mit bis zu 5 Karten Master/Slave gemischt arbeiten) und kann nicht konfiguriert werden.

Um eine HMS-Karte unter Win2K/XP zu nutzen, die Karte in den PC einbauen und dann mit dem Hardwareassistenten von Windows aus dem Verzeichnis Win2K_XP\PBMasterSlave HMS die .inf-Datei installieren. Damit wird der RTIOwdmHMS installiert.

Im IO Drivers-Tab des System...-Menu wird immer der RTIOdrvHMS angegeben, unter allen Windows-Systemen.

Hinweis: Deinstallieren Sie einen evtl. bereits installierten Treiber der Fa. HMS unter allen Umständen und vergewissern Sie sich, dass der Treiber von 3S installiert wurde.

3.10 IO-Treiber RTIOdrvAPIC

Wie oben erwähnt, wird der Treiber bei Bedarf vom System selbst installiert. Der Treiber ist selbstkonfigurierend. Der Treiber misst immer dann, also vor allem beim ersten Start, die Frequenz des Interruptcounters, wenn er einen der Einträge

SetClocksPerUS oder

SetCountsPerMS

nicht in u.g. Registry-Key findet. Wenn der Treiber eine Messung durchführen muss, kann der Systemstart einmalig bis zu 30 Sekunden lang dauern. Der Treiber schreibt die Ergebnisse der Messung in diese Werte in die Registrierungsdatenbank. Bei folgenden Starts stehen sie dann zur Verfügung. Mit Hilfe dieser Werte kann der Treiber auch von Hand skaliert werden.

Er schreibt zu Diagnosezwecken eine Fehlernummer in die Registry.

Unter \HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOdrvAPIC im Wert ErrLog können folgende Bits gesetzt sein:

Bit 0: NO_ERROR	Wird gesetzt, wenn die Initialisierung erfolgreich war.
Bit 1: ERR_APIC_NOT_ENABLED	Der notwendige Interrupt-Controller wurde nicht initialisiert.
Bit 2: ERR_MISSING_CPU_FEATURE	Die CPU des PCs verfügt nicht über den notwendigen Interrupt-Controller.
Bit 3: ERR_APIC_NOT_MAPPED	wie vorher
Bit 4: ERR_IOAPIC_NOT_MAPPED	wie vorher
Bit 5: ERR_INTTEST_TIMEOUT	Es konnte keine Interrupt-Quelle initialisiert werden.
Bit 6: ERR_NO_CPU_CLOCKFRQ	Die Taktfrequenz des PCs konnte nicht ermittelt werden.
Bit 7: ERR_MY_INT_ISUSED	Es konnte kein freier Interrupt-Vektor gefunden werden.
Bit 8: ERR_NO_FREE_INT_FOUND	wie vorher
Bit 9: ERR_TIMERINT_NOT_DISABLED	Es handelt sich nicht um eine APIC-Plattform.

Bei Fehlern dieser Art kontaktieren Sie bitte den 3S Support.

3.11 IO-Treiber RTIOwdmCANAutomata

Um eine CAN-Karte der Fa. Automata zu nutzen, wird dieser Treiber eingebunden. Bisher wurde der Treiber nur für Win2000/XP implementiert, auf Anfrage bei 3S auch für NT4.0 erhältlich.

Um eine der Karten CAN PCI 1N, 2N oder 2NNV (Bezeichnungen 70034500, 70062810 bzw. 70062800) zu betreiben, installieren Sie die Karte zuerst physikalisch dann mit dem Hardwaremanager von Windows, der beim nächsten Einschalten des Rechners erscheint, dann den Treiber aus dem Verzeichnis „CAN Automata“.

Nachdem die Karte installiert wurde, befindet sich in der IO Drivers Registerkarte des System ...-Menus der Eintrag „RTIOwdmCANAutomata“.

Wenn die 2-kanalige Karte mit dem Zusatz NV (für NV-Ram) benutzt wird, wird beim Starten des Treibers automatisch die Einstellung des RTE-Kernels auf diesen Retain-Bereich. (Es wird beim Starten des Systems die physikalische Adresse und die Größe des NV-Rams auf der Karte eingetragen.)

Wenn der Treiber das automatische Umkonfigurieren des RTE-Kernels unterlassen soll, kann ihm in seinem Registry-Key das vorgegeben werden:

In HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOwdmCANAutomata\Params einen neuen DWORD-Wert mit dem Namen „NoRetain“ erzeugen und den Wert zu 1 setzen.

3.12 IO-Treiber RTIOwdmSofting

Um eine Profibus Master Karte „PROFIboard PCI“ der Fa. Softing mit der RTE betreiben zu können, muss die Karte mit der Datei RTIOwdmSofting.inf installiert werden.

Der Treiber ist auf Windows 2000 und XP funktionsfähig.

In HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RTIOwdmSofting\Params kann man optional einen Wert „SyncMode“ vom Typ DWORD angeben und diesen explizit auf 0 setzen. Damit wird die Karte ihren eigenen Profibusupdatezyklus fahren, ohne an den Taskzyklus der SPS gekoppelt zu sein. Normalerweise ist es besser, den Profibuszyklus von der SPS-Task aus zu triggern.

Konfiguriert wird der Profibus mit dem CoDeSys-Konfigurator. Die Konfigurationsdaten werden zusammen mit dem Programm auf die Steuerung geladen und damit wird vom Treiber die Karte konfiguriert.

4 Kommunikation

Das Programmiersystem CoDeSys kann mit der RTE auf zwei Arten kommunizieren:

- über den Shared Memory Treiber
- über den TCP/IP Level2 Route Treiber

Andere Kommunikationswege sind möglich (z.B. Profibus FMS über zwei Hilscher-PB-Karten), jedoch nicht Teil des Setups und müssen bei 3S GmbH angefragt werden.

4.1 Shared Memory Treiber

Um den Treiber benutzen zu können, muss auf dem gleichen Rechner, auf dem auch die SoftSPS läuft, der Gateway der Fa. 3S laufen. Wählen Sie dann auf dem Programmiersystemrechner unter „Online -> Kommunikationsparameter“ als Gateway den auf dem Zielsystemrechner aus und als Kommunikationskanal den Shared Memory Treiber. (U.U. mit „Neu...“ erzeugen.)

Danach kann das Programmiersystem mit der PLC kommunizieren.

4.2 TCP/IP Level2 Route Treiber

Um den Treiber zu benutzen, ist es gleichgültig, auf welchem Rechner der Gateway der Fa. 3S läuft.

Wählen Sie im Programmiersystem „Online -> Kommunikationsparameter“ und dann als Gateway z.B. den lokalen. Als Kommunikationskanal wählen Sie TCP/IP Level2 Route und tragen als Zieladresse den Hostname oder die IP-Adresse des Zielsystemrechners ein.

Danach kann das Programmiersystem mit der PLC kommunizieren.

5 Systemdiagnose

5.1 Busdiagnose der IO-Treiber

Alle IO-Treiber werden zyklisch, nachdem eine IEC-Task ihre Prozessdaten an die Treiber geschrieben / von den Treibern gelesen hat, aufgefordert die folgende Diagnosestruktur zu füllen:

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

Die Struktur ist in der Bibliothek **BusDiag.lib** definiert.

Die IO-Treiber schreiben diese Struktur an die in der Konfiguration für jeden Busmaster angegebene Diagnoseadresse. Diese Funktion ist nur für solche Busmaster aktiv, die auch mit CoDeSys konfiguriert werden.

Für jeden möglichen Busteilnehmer wird in EXTENDEDINFO ein Byte reserviert, in dem die Bits 0 – 2 wie folgt genutzt werden:

Bit 0: Busteilnehmer ist in der Konfiguration vorhanden.

Bit 1: Busteilnehmer ist auf dem Bus verfügbar.

Bit 2: Busteilnehmer meldet Fehler.

Meldet ein auf dem Bus verfügbarer Teilnehmer einen Fehler, kann seine spezifische Diagnoseinformation mit dem Baustein DiagGetState (ebenfalls definiert in BusDiag.lib) gelesen werden.

Der Baustein DiagGetState wird für einen bestimmten Busteilnehmer aufgerufen.

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

Der Baustein beginnt mit einer steigenden Flanke an seinem ENABLE-Eingang zu arbeiten.

DRIVERNAME ist der Name des Treibers (Adresse des Namens), an den der Diagnoseauftrag gehen soll. Wird hier 0 eingetragen, wird der Diagnoseauftrag an alle vorhandenen Treiber weitergereicht.

DEVICENUMBER identifiziert den Bus, der von diesem Treiber verwaltet wird. (Der Hilscher-Kartentreiber kann z. B. bis zu 5 Karten (Busse) verwalten.) Der Index ist 0-basiert.

BUSMEMBERID identifiziert den Busteilnehmer. Bei einer CANopen-Karte ist BUSMEMBERID z. B. die NodeID, bei einer PB-DP-Karte die Stationsadresse des Teilnehmers usw. (Allgemein gesprochen, erlaubt die BUSMEMBERID eine eindeutige Zuordnung des Busteilnehmers. Die Art und Weise der Zuordnung ist Bus- bzw. Treiberspezifisch.)

READY ist ein Ausgangsparameter, der gesetzt wird, wenn der Diagnoseauftrag bearbeitet wurde.

In STATE befinden sich dann einer der Werte aus

```
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
```

In dem Byte-Array EXTENDEDINFO befinden sich bis zu 100 Bytes herstellerspezifischer Diagnosedaten des Busteilnehmers.

6 Die Taskkonfiguration zusammen mit CoDeSys 2.3

CoDeSys V2.3 unterstützt für CoDeSys SP RTE eine erweiterte Taskkonfiguration, mit der zusätzlich folgende Features verfügbar sind:

- Taskspezifische Watchdog-Überwachung: Es kann für jede Task eine maximale Ausführungszeit angegeben werden, die sie nicht überschreiten darf. Mikrosekunden als Zykluszeit einstellbar.
- Eine freilaufende Task kann angelegt werden.
- Systemereignisse können mit Funktionsaufrufen dem IEC-Programm zur Verfügung gestellt werden.

6.1 Allgemein

In der Taskkonfiguration können verschiedene (maximal 16) Tasks angelegt werden. Die RTE führt ein priorisiertes, preemptives Multitasking durch, bei dem Tasks mit der projektierten Zykluszeit periodisch aufgerufen werden.

Dabei muss nach wie vor auf eine eindeutige Vergabe der Prioritäten geachtet werden. (Mehrere Tasks mit der gleichen Priorität führt zu Fehlfunktionen des Systems, sollen mehrere Bausteine von einer Task aufgerufen werden, können unter einer Task auch mehrere Bausteinaufrufe projektiert werden. In neueren Versionen von CoDeSys führen zwei gleiche Prioritäten auch zu einem Fehler beim Übersetzen des Projekts.)

6.2 Taskspezifischer Watchdog

Wird die erweiterte Taskkonfiguration benutzt, so wird die Einstellung „Watchdog – Enable with multipliert“ aus dem Register „Config“ des „System ...“-Menus ignoriert.

Zu jeder Task kann eine maximale Ausführungszeit (mindestens jedoch die Zykluszeit) angegeben werden. Benötigt eine Task länger als die hier eingestellte Zeit, wird eine Watchdog-Exception ausgelöst. Die Reaktion auf ein solches Ereignis ist im Normalfall ein Programmstop, außer es wurde das Systemereignis „excpt_watchdog“ mit einem Funktionsaufruf verknüpft. Dann entscheidet der Rückgabewert der IEC-Funktion, ob ein Programmstop erfolgt (Rückgabewert = 0), oder die Task einfach weiterbearbeitet wird (Rückgabewert \neq 0). Mehr zu Systemereignissen lesen Sie im Abschnitt 6.5.

(Hinweis: Der Parameter „Empfindlichkeit“ in der Taskkonfiguration wird von diesem Zielsystem nicht ausgewertet und ist immer 1.)

6.3 Mikrosekunden als Zeitbasis

In der erweiterten Taskkonfiguration können auch Mikrosekunden als Einheit der Taskzykluszeit verwendet werden. (Dazu einfach eine Zahl im Feld „Zykluszeit“ eintragen, dann kann von ms auf μ s umgeschaltet werden.) Solange aber die RTE nicht im „System...“-Menu im Register „Config“ auf eine kleinere Zeitbasis umgestellt ist, sind die tatsächlichen Zykluszeiten immer Vielfache einer Millisekunde.

Erst wenn die Zeitbasis der RTE auf kleinere Werte eingestellt ist, können Zykluszeiten, die einem geradzahligen Vielfachen der Zeitbasis entsprechen, auch ausgeführt werden.

Um also z.B. eine 100 μ s-Task anzulegen, muss im Feld „Zykluszeit“ der Taskkonfiguration 100 eingegeben werden, dann die Einheit auf μ s umgestellt werden. Im „Config“-Register des „System...“-Menus wird die Zeitbasis („Timebase“) auf 50 μ s gestellt und anschließend das System gestartet.

Eine in dieser Task programmierte Zählvariable ($a := a + 1$;) wird nun 10000 Mal pro Sekunde inkrementiert.

6.4 Freilaufende Task

Unter einer freilaufenden Task wird eine Task verstanden, die, sobald sie mit ihrer Arbeit fertig ist wieder von vorn beginnt. Das Zeitraster in dem diese Task abgearbeitet wird, hängt also nicht von der Konfiguration ab, sondern von ihrer eigenen Ausführungszeit.

Die freilaufende Task, falls verwendet, sollte immer die niedrigste Priorität (höchster Zahlenwert) aller konfigurierten Tasks haben, da Tasks mit niedrigerer Priorität praktisch nicht aufgerufen werden.

6.5 Systemereignisse

In der Taskkonfiguration können Systemereignisse mit Funktionsaufrufen verknüpft werden. Auf bestimmte Ereignisse hin ruft also die RTE in IEC programmierte Funktionen auf, so dass das Anwenderprogramm noch auf bestimmte Ereignisse reagieren kann.

Allerdings ist die Funktionalität dieser Funktionen begrenzt. In einer so genannten „Callback-Funktion“ könne z.B. keine Dateioperationen ausgeführt werden. Alle asynchronen Funktionalitäten, die das System bietet, können nicht genutzt werden. Es dürfen auf keinen Fall in Callback-Funktionen Breakpoints gesetzt werden (Systemabsturz).

Innerhalb der Callback-Funktion selbst können andere Funktionen aufgerufen werden, jedoch dürfen auf gar keinen Fall lokale Variablen angelegt werden.

Um Systemereignisse benutzen zu können, muss die **SysLibCallback.lib** ins Projekt eingebunden sein.

Die Systemereignisse rufen die angemeldeten Funktionen einfach auf. Das einzige Ereignis, bei dem der Rückgabewert das Verhalten beeinflusst ist die WATCHDOG_EXCEPTION (siehe dazu oben 6.2, Taskspezifischer Watchdog).

7 Die Systembibliotheken

Zusammen mit CoDeSys SP RTE können verschiedene Systembibliotheken benutzt werden. Die Benutzung dieser Systembibliotheken wird im Folgenden beschrieben. Alle hier nicht beschriebenen Systembibliotheken können auch nicht benutzt werden.

7.1 DllCall.lib mit SysLibSystemCall.lib

Mit der DllCall.lib zusammen muss immer die SysLibSystemCall.lib ins Projekt eingebunden werden, sonst resultieren Übersetzungsfehler. Die Bibliothek ermöglicht den asynchronen Aufruf von Funktionen, die in kundenspezifischen DLLs (Dynamic Link Library) vorhanden sind. Zur Erstellung einer solchen DLL benutzen Sie bitte das, im Setup von CoDeSys SP RTE enthaltene Toolkit.

Mit dem FB „DllCall“, dem ein Funktionsname, ein Timeout und ein Aus- und ein Eingabepuffer übergeben werden, können Funktionen in normalen Windows-DLLs aufgerufen werden.

Nachdem diese Funktionen aufgerufen wurden und zurückgekehrt sind, wird der Ausgabeparameter des FB normalerweise auf STATUS_READY oder STATUS_ERROR gesetzt, Je nach Rückgabewert der DLL-Funktion.

Kehrt die Funktion nicht innerhalb des vorgegebenen Timeouts zurück (das Timeout 16#FFFFFFFF schaltet die Timeout-Überwachung ab), wird der Status auf STATUS_TIMEOUT gesetzt.

Der STATUS_EXCEPTION wird nur gesetzt, wenn die DLL-Funktion eine Exception verursacht hat.

7.2 SysLibCallback.lib

Sollen Systemereignisse zur Laufzeit an bestimmte IEC-Funktionen gekoppelt werden, kann dazu die Funktion **SysCallbackRegister** benutzt werden. Um die Kopplung wieder aufzuheben wird die Funktion **SysCallbackUnregister** benutzt.

Die Bibliothek enthält eine Enumeration von allen Events, die vorgesehen wurden.

Tatsächlich benutzt werden können die, die in der Taskkonfiguration von CoDeSys, nachdem das Zielsystem auf CoDeSys SP RTE umgestellt wurde, sichtbar sind.

Innerhalb der Callback-Funktion selbst können andere Funktionen aufgerufen werden, jedoch dürfen auf gar keinen Fall lokale Variablen angelegt werden.

7.3 SysLibCom.lib

Die SysLibCom.lib wird zur Benutzung der seriellen Schnittstellen eines PCs vom Anwenderprogramm aus. Um eine Schnittstelle zu verwenden muss folgendermaßen vorgegangen werden:

- Schnittstelle öffnen mit **SysComOpen**, wobei die Schnittstelle aus dem Enum der Bibliothek COM1..8 gewählt werden muss.
- Die Einstellungen der Schnittstelle setzen, mit der Funktion **SysComSetSettings** oder **...SetSettingsEx**. Wird die Funktion **SysComSetSettings** benutzt, werden automatisch 8 Datenbits, kein Hardwarehandshake eingestellt. Zur Benutzung der erweiterten Einstellungen lesen Sie bitte die Kommentare in der Strukturdefinition von COMSETTINGSEX. Beachten Sie, dass die Funktionen zum Einstellen der Schnittstellenparameter nur ein Mal nach dem Öffnen der Schnittstelle benutzt werden können. Um die Schnittstellenparameter einer Schnittstelle zu ändern, muss diese zunächst geschlossen werden.
- Danach können mit dem sog. Handle (Rückgabewert von **SysComOpen**, muss ungleich 0 und –1 sein, da ansonsten die Schnittstelle nicht geöffnet werden konnte.) als Eingabeparameter die Funktionen **SysComRead** und **SysComWrite** aufgerufen werden. Die Funktion SysComWrite schreibt den Inhalt eines Puffers auf die Schnittstelle und die Funktion SysComRead liest von der Schnittstelle Zeichen ein, so viele wie im Lesepuffer der Schnittstelle vorhanden sind, maximal jedoch die übergebene Puffergröße.

- Wird die Schnittstelle nicht mehr gebraucht, kann sie mit **SysComClose** wieder freigegeben werden.

7.4 SysLibFile.lib

Die SysLibFile bietet dem Anwenderprogramm Zugriff auf das Dateisystem des PCs.

Unterstützt werden derzeit die Funktionen

- SysFileOpen,
- SysFileClose,
- SysFileRead,
- SysFileWrite
- SysFileDelete.

Alle anderen Funktionen der SysLibFile werden derzeit nicht unterstützt.

Um eine Datei zu öffnen, wird die Datei **SysFileOpen** benutzt. Die Funktion bekommt den Dateinamen, der ein vollständiger Dateipfad sein muss, übergeben, sowie den Modus, in dem die Datei geöffnet werden soll:

- Modus „w“ öffnet die Datei im Schreibmodus, d.h. eine evtl. vorhandene Datei mit diesem Namen wird überschrieben. Achtung: Bereits das Öffnen der Datei mit „w“ und wieder Schließen genügt, um eine Datei zu überschreiben und eine Datei mit 0 Byte Länge zu erzeugen.
- Modus „r“ öffnet eine Datei zum Lesen. Das sog. Datei-Handle, das die Funktion SysFileOpen zurückgibt ist ungültig (Wert -1 bzw. 16#FFFFFFFF), wenn die Datei nicht existiert. Die Datei wird zum sequentiellen Lesen geöffnet, d.h. mit jedem Lesezugriff wird die Lese-Position um die Anzahl gelesener Bytes weiter geschoben.
- Modus „a“ öffnet eine Datei zum anhängenden Schreiben (engl. append). Dabei wird eine vorhandene Datei zum Schreiben geöffnet, schreiben auf diese Datei hängt Daten ans Ende der Datei an.

Die Funktionen **SysFileRead** und **SysFileWrite** lesen/schreiben aus/in Dateien. Ihnen wird jeweils ein Puffer und ein Datei-handle (Rückgabewert der Funktion SysFileOpen) übergeben.

Um die Datei wieder freizugeben, also nach der Benutzung zu schließen, wird **SysFileClose** aufgerufen.

7.5 SysLibPorts.lib

Mit der Bibliothek SysLibPorts.lib kann das Anwenderprogramm direkt auf IO-Ports des PCs lesend und schreibend zugreifen. Dabei werden die Portzugriffe nach Datenbreite des Zugriffs unterschieden, es gibt also jeweils eine Funktion für Byte-, Wort- und Doppelwortzugriff.

Beispiel:

Der Aufruf SysPortIn(16#378) liest z.B. auf einem Standard-PC vom 1. Byte der parallelen Schnittstelle.

Die Benutzung dieser Funktionen erfordert genaue Kenntnisse der verwendeten Hardware.

Die Portadressen müssen im Bereich 0 ... 65535 liegen.

7.6 SysLibTime.lib

Die SysLibTime.lib kann zur genauen Zeitmessung und zur Abfrage der Systemzeit verwendet werden. Dabei ist zu beachten, dass der FB **CurTime** eine sehr schnelle Funktion ist, die sofort zurückkehrt und die Zeit in Mikrosekunden seit dem letzten Systemstart liefert.

Der FB **CurTimeEx** liefert zwar zusätzlich dazu noch die aktuelle Systemzeit zurück, kann aber u.U. 1ms brauchen, bis er zurückkehrt.

Beispiel zur Benutzung des CurTime-FB:

```
VAR
    ct : CurTime;
    syst : SysTime64;
END_VAR
ct(SystemTime := syst);
```

Jetzt steht in syst ein 64-bit Wert, der die Zeit in μs seit dem letzten Systemstart enthält.

(Anmerkung: Um damit Zeitdifferenzen zu berechnen (bis zu 4294967295 μs) genügt es, mit den Low-Anteilen der SysTime64-Struktur zu rechnen.)

7.7 SysLibSockets.lib

Die SysLibSockets.lib wird vor allem von Netzwerkvariablen (über UDP) benutzt.

Die Bibliothek kann aber natürlich auch vom Anwenderprogramm für eigene UDP-Datenübertragungen verwendet werden.

Die **NetVarUDP_Lib.lib** kann als Beispiel zur Verwendung der Funktionen herangezogen werden. Es sind nur die dort verwendeten Funktionen auch tatsächlich implementiert.

7.8 SysLibShm.lib

Diese Bibliothek dient zum Anlegen und für die Zugriffe auf einen Speicherbereich, der mit anderen Prozessen gemeinsam genutzt werden kann bzw. auf eine physikalische Adresse verweist (Shared-Memory, abgekürzt ShM).

Wenn das Zielsystem die Funktionalität beinhaltet, können die Bibliotheksfunktionen verwendet werden, um das ShM zu öffnen, zu schließen und lesend und schreibend darauf zuzugreifen. Die Funktionen zum Lesen, Schreiben und Schließen benötigen das Handle, das beim Öffnen des ShM erhalten wird. Die Abarbeitung erfolgt synchron.

Die Abarbeitung kann u.U. mehrere 10 ms lang sein. Es wird empfohlen, diese Funktion nicht in der zyklischen Steuer-Task oder nur einmal bei der Initialisierung zu benutzen.

Die Funktionalität der Bibliothek im Einzelnen entnehmen Sie bitte der Online-Hilfe von CoDeSys.

Die Bibliothek kann auf der RTE nur zum Einblenden (Mappen) von physikalischem Speicher benutzt werden, nicht zum Anlegen gemeinsamer Speicherbereiche mit anderen Prozessen.

7.9 SysLibPciCards.lib

Die Bibliothek wird in der Online-Hilfe von CoDeSys beschrieben.

8 Verhalten im Fehlerfall

Es gibt verschiedene Ursachen, die zu einem Zyklus-Stop des Anwenderprogramms führen.

Die Fehler sind:

Nummer	Name	Ursache
16	Watchdog	Das Anwenderprogramm hat die projektierte Zykluszeit um den in der Konfiguration (siehe „Config“ in „Bedienung über Serviceicon“) festgelegten Faktor überschritten. Für die Taskkonfiguration der CoDeSys V2.3, die die Festlegung des Faktors in der Programmieroberfläche unterstützt, gilt der dort eingestellte Faktor.
20	Fieldbus error Error in configuration data	Ein IO-Treiber konnte sich nicht richtig initialisieren. Kann nur beim Starten der PLC auftreten. Bei der Konfiguration einer Feldbuskarte wurden die Daten des Konfigurators nicht akzeptiert.
21	Error update IOs	Kann nur nach dem Programmladen auftreten. Wenn ein Ein- oder Ausgang auf eine Ein- oder Ausgangsadresse, zu der es keinen IO-Treiber gibt, gelegt wurde.
81	Access violation	Zur Programmlaufzeit hat das Anwenderprogramm versucht, auf eine ungültige Speicheradresse zuzugreifen. (Üblicherweise durch Benutzung eines nicht initialisierten Zeigers.)
258	Division by zero	Eine Division durch null wurde versucht.
336-343	FPU-Fehler	Eine ungültige Gleitkommaoperation wurde ausgeführt.

Alle diese Fehler führen zum gleichen Verhalten der PLC:

- Die Ausgänge werden, je nach dem, ob es in den Konfigurationsdaten gewünscht wurde (siehe „Config“ in „Bedienung über Serviceicon“), auf 0 oder 1 gesetzt, oder nicht verändert.
- Der Zyklus aller Tasks wird angehalten. Die PLC wird auf STOP gesetzt. (Nur die Task, die den Fehler verursacht hat, wird sofort verlassen. Evtl. weitere aktive Tasks werden zu Ende bearbeitet. Sie werden keinen neuen Zyklus mehr beginnen.)
- Das IO-Update aller Tasks läuft weiter. (Nur das PAA wurde evtl. auf 0 gesetzt.)

Der aufgetretene Fehler wird im „Diagnostic“-Tab des Bedienmenüs angezeigt.

Die PLC speichert den Fehler mit (falls möglich) der Programmstelle ab. Nach dem Einloggen mit CoDeSys kann dann die Fehlerstelle abgefragt werden (Online -> Aufrufhierarchie).

9 ANHANG

9.1 Die vom Laufzeitsystem verwendeten Registry-Einträge.

Der hier als RegKeyRuntime bezeichnete Schlüssel in der Registry hat den Pfad:

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

(Alle diese Einträge werden normalerweise vom „System...“-Konfigurationsdialog verwaltet und brauchen nicht manuell bearbeitet werden!)

Unter diesem Schlüssel sucht das Laufzeitsystem folgende Werte:

Name	Mögliche Werte	Bedeutung
LastExit, DWORD	0 oder 1	Bevor versucht wird ein Bootprojekt zu laden, wird LastExit 0 gesetzt. Stürzt die PLC ab (wird nicht normal beendet), wird dieser Wert nicht mehr zu 1. Ist dieser Wert 0, wird kein Bootprojekt mehr geladen.
NmiConnect, DWORD	0 oder 1	Wenn dieser Wert vorhanden und 1 ist, wird ein Interrupt-Handler des Laufzeitsystems auf den Int-vektor 2 gelegt. Z.z. nur sinnvoll zusammen mit NmiSchedule.
NmiSchedule, DWORD	Bitweise Auswertung.	<p>Bit0: Wenn dieses Bit gesetzt ist, wird der Timerintervall nicht verändert und auf den Interrupt-Vektor des Timerchips keine eigene Routine gelegt. Der Scheduler wird von der Nmi-Handler-Routine aufgerufen.</p> <p>Bit1: Wenn dieses Bit gesetzt ist, kann der IO-Treiber, der den NMI behandelt, bestimmen, wann NT gescheduled werden muss.</p> <p>Bit2: Wenn dieses Bit gesetzt ist, wird der RTE-Kernel keine Interrupt-Routine für den zyklischen Interrupt selbst installieren. Das muss ein passender IO-Treiber machen, der einen zyklischen Interrupt auf Anforderung erzeugt und die Routine, die normalerweise als NMI-Handler dient, aufruft (oder mit seinem zyklischen Interrupt verbindet).</p> <p>Bit3: Die höchstprioritäre Task wird nur dann aufgerufen, wenn der IO-Treiber, der den Scheduler-Interrupt behandelt und erzeugt, wenn die NMI-Enter-Routine IODRV_NMI_FLAG_SYNCHRONIZE_HIGHPRIOTASK zurückgibt.</p>
MaxNonNtInt, DWORD	50 – 950	Wenn der Scheduler auf einem Nmi arbeitet, der asymmetrisch erzeugt wird (mit dem Ziel, NT weniger CPU-Zeit als der PLC zu geben), wird mit diesem Wert bestimmt, nach welchem Interrupt NT drankommen soll, und nach welchem IEC-tasks, wobei NT immer den kürzeren von beiden zieht.
MaxPLCTime, DWORD	10 – 90	Wird konventionell, also mit dem Timertick gescheduled, bestimmt diese Zahl, in welchem Verhältnis die Zeitscheiben vom NT und vom Laufzeitsystem zueinander stehen. Steht hier also 90, bekommt die PLC 90% CPU-Zeit, NT bekommt 10%.
Path, STRING	\\?\\<gültiger Verzeichnisname, der mit Laufwerksbuchst. beginnt.>	Der Pfad, unter dem alle, das Laufzeitsystem betreffende, Files gesucht und gespeichert werden. Maximal 128 Zeichen.
RetainFlags, DWORD	alle 32 Bits einzeln verwendet	<p>Default: Retain-Speicher wird dynamisch im RAM alloziert, Speicherung in Datei beim Beenden.</p> <p>Bit0 = 1: Retain-Speicher nicht dynamisch allozieren.</p> <p>Bit1 = 1: Retain-Speicher auf physikalische Adresse legen.</p>

Name	Mögliche Werte	Bedeutung
		Sind nur Bits 0 und 1 gesetzt (3), wird angenommen, dass der Retain-Bereich direkt in einem SRAM liegt. Keine explizite Speicherung der Retain-Daten erfolgt. Bit2 = 1: Retains in Datei beim Beenden speichern, wie Default. Bit3 = 1: Retains über IO-Treiber speichern, zyklisch.
RetainAddr	gültige physikalische Speicheradresse	Muss vorhanden sein, wenn Bit1 in RetainFlags gesetzt. Dieser Wert wird als physikalische Speicheradresse für die Speicherung der Retain-Daten verwendet. Die Länge steht in RetainSize.
RetainSize	Größe der gesamten Retain-Daten (unter der physikalischen Adresse, bzw. wieviel alloziert werden soll.)	Muss vorhanden sein, wenn Bit1 in RetainFlags gesetzt ist.
HandleBlueScreen	1 oder 0	Ist dieser Wert vorhanden und 0, so wird der NT-Bluescreen erzeugt, im Falle eines Systemabsturzes. Ist dieser Wert nicht vorhanden oder 0, so „friert“ der Rechner im Falle eines Absturzes ein. Die PLC (wenn sie nicht die Ursache des Absturzes war) läuft weiter.
StopIODrivers	0 oder 1	Der RTService startet automatisch alle aufgelisteten IO-Treiber. Ist dieser Wert 1, so stoppt er sie auch automatisch beim Beenden des Systems.
OutputFlags	alle 32 Bits einzeln verwendet	Bit0: 0: Kein Forcen der Ausgänge beim stoppen der PLC. 1: Wenn die PLC in den Zustand STOP geht, werden die Ausgänge alle auf den Wert des Bit1 (dieses Werts) gesetzt.
StackpoolSize	Sinnvolle Größe des Pools in Byte	Die gesamte Größe des Pools, aus dem sich die Tasks, die von externen Treibern angelegt werden, ihren Stack holen (beim Anlegen der Task). Der Pool wird beim Systemstart alloziert. Wird hier ein Wert kleiner als 512 eingetragen, wird das externe Taskhandling deaktiviert, d.h. IO-Treiber können keine Tasks anlegen. Wird hier nichts angegeben, so werden 64kbytes standardmäßig alloziert.
CodeSize	Größe des Codebereichs in Byte	Die Steuerung legt 2 mal diesen Bereich für den von CoDeSys erzeugten Maschinencode an (IEC-Bausteine). Fehlt der Eintrag, werden 4MB angenommen.
DataSize	Größe des Datenbereichs in Byte	Die Steuerung legt einen Datenbereich für die IEC-Variablen, Merker und PA an. Fehlt der Eintrag, werden 2MB angenommen.
WdgMultiplier	0 – 0xffffffff, 0xffffffff : disable	Die Anzahl der Taskzykluszeiten, um die die Ausführung einer Task länger sein darf, bevor die Watchdog-Überwachung anschlägt. Mit 0 (Default) wird der Watchdog sofort, wenn die projektierte Taskzykluszeit überschritten wird, ausgelöst. Ein Wert 0xffffffff schaltet die Watchdog-Überwachung aus.
Resolution	50,100,200	Gibt (nur für die PLCs mit der Target-Id 44 und 45) die zeitliche Auflösung in µs an.
InterruptFlags	0 oder 1 (momentan)	Wenn 1 werden die Interrupt-Funktionen in einer Umgebung ausgeführt, in der Floatingpoint-Rechnungen erlaubt sind.
AutoStartPLC	Bitweise codiert	Bit0-2 schließen einander aus. Bit0: Der Service startet den Kernel immer automatisch. Bit1: Der Service startet den Kernel nie. Bit2: Der Service startet den Kernel nur, wenn er das letzte Mal korrekt beendet wurde.

Name	Mögliche Werte	Bedeutung
		<p>Bit3-5 schließen einander aus. Bit3: Die PLC lädt immer das Bootprojekt. Bit4: Die PLC lädt das Bootprojekt nie.</p> <p>Bit5: : Die PLC lädt das Bootprojekt, wenn sie das letzte Mal korrekt beendet wurde.</p> <p>Bit6-8 schließen einander aus. Bit6: Das Bootprojekt wird immer gestartet, Bit7: Bootprojekt wird nie gestartet, Bit8: Das Bootprojekt wird nur gestartet, wenn die PLC das letzte Mal korrekt beendet wurde.</p>
DebugFlags	Bitweise codiert	<p>Bit0: Wenn 1, dann werden keine Exception-Handler von der RTE ins System eingehängt -> Fehlersuche mit Kernel-Debugger wird möglich.</p> <p>Bit1: Wenn 1, dann werden beim Programmdownload alle externen Referenzen, die nicht aufgelöst werden können, auf eine Dummyfunktion geleitet -> Download ohne spezielle Systembibliotheken möglich.</p>

Unter dem Unterschlüssel von RegKeyRuntime „IO Drivers“ befindet sich eine Liste mit Namen von IOTreibern. Die Namen der Treiber können die Form <Treibername> oder <Objektname><space><Treibername> haben. Die zweite Form der Eintrags wird verwendet, falls der Treiber ein Treiberobjekt anlegt, dessen Name nicht gleich dem Namen des Treibers ist. Die Werte müssen Driver<Ild.Nummer> heißen, wobei Ild.Nummer von 0 bis 4 gehen kann. Die Werte sind Zeichenfolgen. Mit dieser Zeichenfolge werden die Treiber über die normalen Mechanismen (z.B. CreateFile) angesprochen.

Mehr Informationen zu IOTreibern gibt es im RTIODrv-Toolkit und dem Dokument, das die Schnittstelle zum Laufzeitsystem beschreibt.

Unter dem Unterschlüssel von RegKeyRuntime „External DLLs“ kann sich eine Liste mit DLLs, die beim Systemstart vom Service geladen werden, befinden. Die Liste hat die Form:

Dllx <optional Pfad><Dateiname>

x ist eine laufende Nummer, mit 0 beginnend. Die Funktionen in den DLLs können über die Bibliotheksfunktion „DllCall“ aus dem IEC-Programm heraus aufgerufen werden. Mehr Informationen zum einhängen von DLLs gibt es im AsynchDll-Toolkit.

Oben genannt sind nur die Registry-Einträge, die das Laufzeitsystem direkt benutzt. Es werden aber vom Gesamtsystem evtl. noch mehr Einträge benutzt, da die IO-Treiber natürlich ihre eigenen, für sie notwendigen Einträge anlegen und verwalten.

Änderungshistorie

Version	Beschreibung	Bearbeiter	Datum
	Erstellung	AF	11.10.2000
	Fortlaufende Aktualisierungen	AF	20.11.2000 – 16.12.2004
	Änderungen zu V2.3.4.0	AF	05.08.2005
2.0	Beginn der Dok.versionierung; aktuelle Formatvorlage; Freigabe	MN	17.08.2005
2.1	Kap.2.1.3: „NT“ statt „OS“-settings; Beschreibung von „The PLC has max ___ %CPU time“	MN	22.09.2005
2.1	Kap.3.1: Beschreibung der WaitFlags des Hilschertreibers (#5399)	AF	13.10.2005
2.1	Formal Review, Release	MN	13.10.2005
2.2	Passend zur Version 2.3.5.1 der RTE die Beschreibung der SysLibPciCards und der SysLibShm hinzugefügt. (#5271)	AF	01.06.2006
2.2	Die neue Beschreibung des Config Dialogs eingefügt. (#6078), Kap. 2.1.3	AF	21.06.2006
2.2	<p>Review: Seite 4, Zeile 6, Spalte 71: „CoDeSys-Menu“ ersetzen durch „CoDeSys-Menü“</p> <p>Review: Seite 5, Zeile 8, Spalte 30: „wobei die Aufteilung konfigurierbar ist.“ Evt. Verweis aus Kapitel 6 Taskkonfiguration einfügen</p> <p>Review: Seite 11, Zeile 2, Spalte 2: „auch wenn T#1ms in der Taskkonfiguration steht“ Unklar wofür „T#“ steht</p> <p>Review: Seite 12, Zeile 19: „Retain“ die Optionen „Physical address“ und „Bytes retain memory“ sind nicht beschrieben</p> <p>Review: Seite 14, Zeile 2, Spalte 99: „In Password muss jeweils ein Passwort und das identische in Confirm password eingetragen.“ Fehlt „werden“ am Satzende</p> <p>Review: Seite 14, Zeile 10, Spalte 9: „Die Treiber wie die Dlls müssen jeweils ...“ Evt. missverständlich formuliert, besser „Die Treiber als auch die Dlls müssen jeweils ...“</p> <p>Review: Seite 27, Zeile 2, Spalte 27: „erweiterte Taskkonfiguration“ es ist nicht offensichtlich, wie die erweiterte Taskkonfiguration geöffnet werden kann, evt. Verweis auf CoDeSys Handbuch einfügen</p> <p>Review: Seite 28, Zeile 33, Spalte 96: „dann kann von ms auf us umgeschaltet“ Schreibfehler „us“ in „µs“ umändern</p>	SR	21.06.2006
2.2	Überarbeitung hinsichtlich der Review-Punkte, entsprechende Aktualisierung der englischen Doku-Version, Freigabe V2.1	MN	21.06.2006