FORSCHUNGSZENTRUM ROSSENDORF
Mitglied der Wissenschaftsgemeinschaft Gottfried Wilhelm Leibniz

WISSENSCHAFTLICH-TECHNISCHE BERICHTE
FZR-352
August 2002
ISSN 1437-322X



*Heiko Pietruske, Andreas Schaffrath*

# Entwicklung und Erprobung einer Schnittstelle zwischen der Messdatenerfassungs- und Automatisierungssoftware DIA*dem* von National Instruments und der Wasser/Dampf Stoffwertedaten-Unterbibliothek Libl97

Herausgeber:
Forschungszentrum Rossendorf e.V.
Postfach 51 01 19
D-01314 Dresden
Telefon +49 351 26 00
Telefax +49 351 2 69 04 61
http://www.fz-rossendorf.de/

Als Manuskript gedruckt

FORSCHUNGSZENTRUM ROSSENDORF
WISSENSCHAFTLICH-TECHNISCHE BERICHTE
**FZR-352**
August 2002

*Heiko Pietruske, Andreas Schaffrath*

# Entwicklung und Erprobung einer Schnittstelle zwischen der Messdatenerfassungs- und Automatisierungssoftware DIA*dem* von National Instruments und der Wasser/Dampf Stoffwertedaten-Unterbibliothek Libl97

# Abstract

The Institute of Safety Research (IfS) of the Forschungszentrum Rossendorf (FZR) e.V. is constructing a new large-scale multipurpose test facility TOPFLOW (Transient Two Phase Flow Test Facility). This facility will be probably put into operation in the next two months. For an effective evaluation of the start up experiments and the acceptance trials against the vendors FZR starts with the preparation of automated software tools for the measurement data logging and automation software DIAdem, which is distributed by National Instruments (NI). In a first step an interface for the coupling of a water/steam material property library LibIF97 of the university of applied science Zittau/Görlitz was developed.

This report describes the programming of the General Control Interface (GPI) and its coupling with DIAdem. Additionally the capability of this coupling in connection with autosequences for data evaluation was investigated. Further on effective methods for the TOPFLOW data evaluation were demonstrated and tested against a concrete example. Currently no TOPFLOW data are available. Therefore one selected NOKO experiment was evaluated and first practical experiences were collected. Even this example is easy understandable and clearly seen, it contains every step, which is necessary for the TOPFLOW data evaluation. This contains the opening of files, determination of water/steam material properties with the Dynamic-Link-Library LibIF97.dll, the linkage of different data channels and the generation of layouts for graphics and reports.

The tools presented in this report are an important steps for the evaluation of the experimental data of TOPFLOW. These tools will be adapted now for the assessment of the acceptance trails. Further on now the generation of the automated software sequences for the first scientific tests are developed.

# Kurzfassung

Am Institut für Sicherheitsforschung (IfS) des Forschungszentrums Rossendorf (FZR) e.V. wird derzeit die Mehrzweck-Thermohydraulikversuchsanlage **TOPFLOW** (Transient Two Phase Flow Test Facility) aufgebaut und in Betrieb genommen. Zur effizienten Auswertung dieser Tests sowie den anschließenden Abnahmen gegenüber den Lieferanten wird mit der Erstellung automatisierter Auswerteroutinen für die in TOPFLOW eingesetzte Messdatenerfassungs- und Automatisierungssoftware DIA*dem* von National Instruments (NI) begonnen. Der erste große Schritt hierzu ist die Entwicklung und Erprobung einer Schnittstelle zwischen DIA*dem* und der Wasser/Dampf-Stoffdatenbibliothek LibIF97 der Hochschule Zittau / Görlitz (FH).

In der vorliegenden Arbeit wurde die hierzu notwendige General Programming Interface (GPI) Schnittstelle programmiert und erfolgreich mit DIA*dem* gekoppelt. Anschließend wurden darüber hinaus die Leistungsfähigkeit dieser Kopplung zur automatisierten Datenauswertung mittels Autosequenzen untersucht und effiziente Methoden für die zukünftige Auswertung der Messdaten von TOPFLOW aufgezeigt.

Diese Ideen wurden anschließend an einem Beispiel umgesetzt. Da zur Zeit jedoch noch keine TOPFLOW-Versuchsdaten vorliegen, wurde der Nachweis einer erfolgreichen Kopplung anhand der Auswertung eines ausgewählten NOKO-Experiments erbracht. Hierdurch konnte ferner das Zusammenspiel der Stoffwertebibliothek mit Autosequenzen getestet und erste praktische Anwendererfahrungen gesammelt werden.

Das Beispiel der Auswertung eines NOKO-Experiments ist bewusst einfach und überschaubar gewählt. Es beinhaltet mit dem Öffnen eines Files, der Bestimmung von Stoffwerten mit Hilfe der Dynamic-Link-Library LibIF97.dll, der Verknüpfung von verschiedenen Kanälen und der Erstellung und Ausgabe von Präsentationsgrafiken bereits jetzt alle für die Auswertung von TOPFLOW Experimenten benötigten Elemente.

Die in dieser Arbeit entwickelten Tools sollen nun in einem weiteren Schritt für die Auswertung der TOPFLOW Experimente adaptiert und die zur Begutachtung und Bewertung der Abnahmeversuche notwendige Autosequenzen erstellt werden. Ferner soll jetzt schon mit der Erstellung der Auswerteroutinen und -autosequenzen für die ersten Versuchsreihen begonnen werden.

# Inhaltsverzeichnis

# Bildverzeichnis

# 1 Einleitung

Am Institut für Sicherheitsforschung (IfS) des Forschungszentrum Rossendorf (FZR) e.V. wird derzeit die Mehrzweck-Thermohydraulikversuchsanlage **TOPFLOW** (Transient Two Phase Flow Test Facility) aufgebaut und voraussichtlich Ende September 2002 in Betrieb gehen. Die Anlage dient zur Untersuchung von transienten Zweiphasenströmungen in vertikalen Rohrleitungen, von Siedephänomenen in großen Behältern sowie offnen Wasserpools, der Erprobung passiver Komponenten sowie des Betriebsverhaltens liegender (WWER-) Dampferzeuger ohne und in Anwesenheit nichtkondensierbarer Gase und der Untersuchung transienter Zweiphasenströmungen in reaktortypischen Geometrien (Ringraum, Heißer Strang) zur Schaffung einer experimentellen Datenbasis für die Entwicklung und Validierung von Modellen für Computational Fluid Dynamic (CFD-) Codes (vgl. [SCA-02]).

Zur effizienten Auswertung der bei den Experimenten anfallenden Datenmengen wird die komplette Handhabung der Experimentdaten mit Hilfe der Messdatenerfassungs- und Auswertungssoftware DIA*dem* 8.0 von National Instruments (NI) realisiert. Zur Handhabung der Daten gehört u.a. die:

- Erfassung der Experimentdaten sowie deren Speicherung in einer geeigneten Struktur,
- Durchführung von Plausibilitätskontrollen
- Auswertung der Experimente
- Validierung der Messdaten (u.a. anhand einer Kontrolle der Knotenbilanzen für Massen und Energien sowie eine Überprüfung der dp-Ketten in den einzelnen Kreisläufen),
- Betrachtung der Genauigkeiten

sowie

- Durchführung von Vorarbeiten für eine detaillierte phänomenologische Auswertung.

Die Messdaten gliedern sich in verschiedene Gruppen. Dies sind zum einen die Daten der Betriebmesstechnik. Hierzu zählen im wesentlichen Temperaturen, Drücke, Differenzdrücke und Relativgrößen (z.B. Öffnungsgrad einer Armatur). Diese werden im Experiment mit einer Frequenz von größer 1 Hz abgetastet und in einem einzigen File gespeichert. Darüber hinaus gibt es weitere Files für die räumlich und zeitlich hochauflösende Sondermesstechnik (u.a. Nadelsonden, Gittersensoren, Tomographen), zu deren Auswertung spezielle Softwaretools benötigt werden. Diese sollen aber in der vorliegenden Arbeit nicht weiter betrachtet werden.

Aus den gemessenen Betriebsdaten werden an zahlreichen Stellen indirekte Größen abgeleitet. Dies sind z.B. Massenströme, die anhand des an einem Drosselkörper gemessenen Druckabfalls sowie den Zuständen des Fluides, die sich anhand von Druck und Temperatur ergeben, berechnet werden. Weitere abgeleitete Größen sind z.B. der mit Hilfe einer Differenzdruckmessung gemessene Füllstand in einem Behälter, oder die Energieinhalte einzelner Kreisläufe. Anhand der o.g. Beispiele wird deutlich, dass bei der Handhabung der Messdaten an zahlreichen Stellen Stoffwerte wie Dichte, Enthalpie, Viskosität, Wärmeleitfähigkeit, spezifische Wärmkapazität, etc. benötigt werden. Diese Stoffwerte können sogenannten Wasser/Dampf-Tafeln entnommen werden.

Die derzeit gültige Industrie Formulierung der Internationalen Organisation für die Eigenschaften von Wasser und Wasserdampf IAPWS ist die sogenannte IF97. Diese wird nach Ablösen der IFC-67 im Jahr 1997 in Abnahme- und Garantierechnungen von Kraftwerken weltweit verwendet. Der Gültigkeitsbereich der IF97 erstreckt sich von 0 - 800 °C und 0,00061 - 100 MPa (vgl. [WAW-98]). Zur einfachen Handhabung wurde für die IF97 eine Stoffdaten-Unterprogrammbibliothek erstellt. Diese ermöglicht es komfortabel Stoffwerte z.B. in Microsoft Excel zu berechnen und die berechneten Stoffdaten in thermodynamischen Zustandsdiagrammen darzustellen (vgl. [KRH-01]).

Ziel des vorliegenden Belegs ist die Stoffwerte-Unterprogrammbibliothek LibIF97 der IF97 - diese wurde vom Fachbereich Technische Thermodynamik der Hochschule Zittau/Görlitz erstellt - an DIA*dem* anzukoppeln und die hierfür nötigen Schnittstellen zu erstellen. Da derzeit in TOPFLOW noch keine Experimentdaten vorliegen soll zum Nachweis der erfolgreichen Kopplung für ein ausgewähltes NOKO-Experiment die

erste Energiebilanz des Kondensatorbündels ausgewertet werden. Ferner soll das Zusammenspiel der Stoffwertebibliothek mit Autosequenzen getestet werden. Die vorliegende Arbeit beinhaltet somit eine wesentliche Vorarbeit für die automatisierte Handhabung der Versuchsdaten.

Die vorliegende Arbeit ist wie folgt gegliedert: in Kapitel 2 dieser Arbeit wird zunächst die Messdatenerfassungs- und Auswertungssoftware DIA*dem* 8.0 von National Instruments beschrieben. Hierbei liegt der Schwerpunkt auf der in DIA*dem* enthaltenen Schnittstelle zur Einbindung von Anwenderroutinen sowie dem Datentransfer. Diese Schnittstelle wird genutzt um die Wasser/Dampf-Stoffwerte Unterprogrammbibliothek LibIF97 einzubinden. In Kapitel 3 sind dann die Grundlagen der Programmierung von Autosequenzen in Visual Basic Script beschrieben. Mit Hilfe dieser Kenntnisse wird nun in Kapitel 4 für einen ausgewählten NOKO-Datensatz eine Autosequenz programmiert und speziell das Zusammenspiel von Autosequenzen und der Wasser/Dampf-Stoffwerte Unterprogrammbibliothek getestet. Eine wichtige Zielsetzung dieser Arbeit ist die Sammlung von Nutzererfahrungen. Diese sind in Kap. 4 beschrieben. Abschließend werden in Kapitel 5 die wichtigsten Ergebnisse der vorliegenden Arbeit zusammengefasst und ein Ausblick auf die nun anstehenden weiteren Arbeiten zur Programmierung der Messdatenerfassung sowie der Auswerteroutinen im Rahmen der Inbetriebnahme der Versuchsanlage gegeben.

# 2 Messdatenerfassungs- und Automatisierungssoftware DIA*dem*

Zu Beginn des EMSR (Elektro-, Mess-, Steuer- und Regeltechnik) Projektes "TOP-FLOW" war zunächst eine Grundsatzentscheidung bzgl. der für die Messdatenerfassung einzusetzenden Software zu treffen. Hierbei war neben der Flexibilität, die zu realisierenden Summenabtastraten, die Kompatibilität mit gängigen Bussystemen, das Vorhandensein einer breiten Palette von einsetzbaren Messgeräten (mit getesteten Treibern), einer übersichtlichen 2D und 3D Visualisierung der Messdaten speziell die Möglichkeit der Durchführung umfangreicher mathematischer Analysen mit der Option der Ankopplung eigener Auswertealgorithmen, ein wichtiges Auswahlkriterium. Ferner ist DIA*dem* in der Lage mit den gängigen WINDOWS Programmen zu kommunizieren und bietet in Kombination mit einer Datenbankanbindung (z.B. MS ACCESS, dBase, Oracle, FoxPro, Microsoft SQL-Server, MySQL) eine komfortable Möglichkeit zur Datenarchivierung und -dokumentation.

Nachfolgend wird im ersten Unterkapitel zunächst ein grober Überblick über die einzelnen Module von DIA*dem* (DATA, VIEW, GRAPH, CALC, DAC, VISUAL und AUTO) gegeben. Anschließend wird dann in Kap. 2.2 im Detail das General Programming Interface (GPI) vorgestellt, mit dem der Nutzer externe Funktionen und Unterprogramme an DIA*dem* ankoppeln kann. Dieses Interface wird auch zur Ankopplung der auf der IF97 der Internationalen Organisation für die Eigenschaften von Wasser und Wasserdampf IAPWS basierenden Stoffwerte-Unterprogrammbibliothek LibIF97 genutzt. Teil 3 dieses Kapitels erklärt, wie die programmierte Schnittstelle in DIA*dem* eingebunden wird und der Aufruf von Stoffwertefunktionen der LibF97 aus DIA*dem* im Detail erfolgt.

## *2.1 Genereller Überblick*

Die Stärke der von der Gesellschaft für Strukturanalyse (GfS) entwickelten und nach dem Zusammenschluss mit National Instruments (NI) nun von NI vertriebenen Messdatenerfassungs- und Automatisierungssoftware DIA*dem* ist - wie bereits in der Einleitung von Kap. 2 beschrieben - das breite Spektrum an Aufgaben, die komfor-

tabel mit DIA*dem* bearbeitet werden können, sowie die Flexibilität. Der Nutzer kann sich aus einer Vielzahl von Modulen eine speziell auf seine Anforderungen zugeschnittene DIA*dem* Variante zusammenstellen. Die einzelnen Module (diese werden in der DIA*dem* Terminologie Geräte genannt) beinhalten ein Paket von Applikationen, die für spezielle Aufgaben (z.B. der Messdatenerfassung und -verarbeitung, der Datenverwaltung, der Erstellung von Graphiken und deren Präsentation sowie der mathematischen und grafischen Analyse, etc.) konzipiert sind. Die wichtigsten Geräte, die am FZR zum Einsatz kommen, sind die Module DATA, VIEW, GRAPH, CALC, DAC, VISUAL und AUTO. Diese werden nachfolgend kurz vorgestellt.

Sämtliche Geräte sind unter einer einheitlichen Benutzeroberfläche zusammengefasst und stehen dem Nutzer jederzeit zur Verfügung (vgl. Bild 2.1). Im linken Bereich der Oberfläche wählt man die verschiedenen Geräte aus, welche dann im hier grau dargestellten rechten Bereich eingeblendet werden. Es ist jederzeit möglich verschiedene Geräte zur gleiche Zeit zu öffnen.

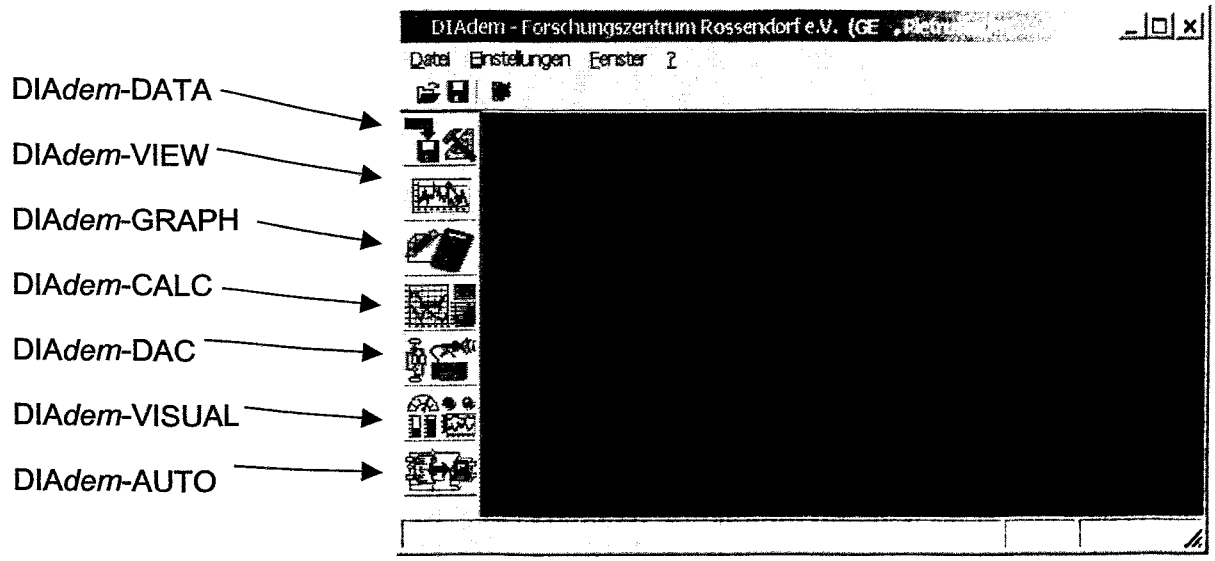
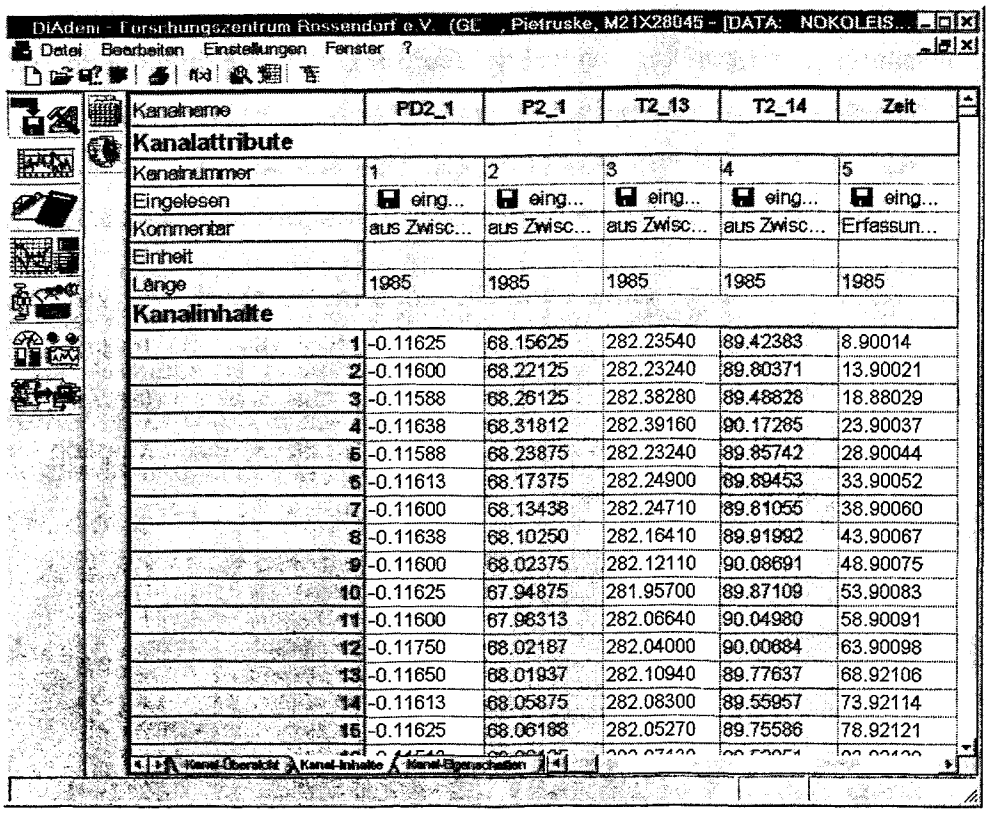



**Bild 2.1:** DIA*dem*-Oberfläche.

Das Gerät DIA*dem*-DATA dient zum Laden, Speichern, Sichten und Bearbeiten von Daten. Die einzelnen Datenfiles sind in Form von Kanälen aufgebaut, von denen jeder Kanal eine unabhängige Datenreihe darstellt (vgl. [NID-00]). Das Ansprechen der Kanäle kann alternativ über den Kanalnamen bzw. der Kanalnummer erfolgen. Bild 2.2 zeigt einen geladenen Datensatz, bestehend aus fünf Kanälen (PD2_1, P2_1,

T2_13, T2_14 und Zeit), wobei jedem Kanal Attribute, wie z.B. Kanalnummer, Kommentar, Einheit und Kanallänge, zugeordnet sind. Ferner lassen sich über einen integrierten Taschenrechner diese Kanäle miteinander verknüpfen und somit einfache mathematische Funktionen realisieren.




| Kanalname | PD2_1 | P2_1 | T2_13 | T2_14 | Zeit |
|---|---|---|---|---|---|
| **Kanalattribute** | | | | | |
| Kanalnummer | 1 | 2 | 3 | 4 | 5 |
| Eingelesen | eing... | eing... | eing... | eing... | eing... |
| Kommentar | aus Zwisc... | aus Zwisc... | aus Zwisc... | aus Zwisc... | Erfassun... |
| Einheit | | | | | |
| Länge | 1985 | 1985 | 1985 | 1985 | 1985 |
| **Kanalinhalte** | | | | | |
| 1 | -0.11625 | 68.15625 | 282.23540 | 89.42383 | 8.90014 |
| 2 | -0.11600 | 68.22125 | 282.23240 | 89.80371 | 13.90021 |
| 3 | -0.11588 | 68.26125 | 282.38280 | 89.48828 | 18.88029 |
| 4 | -0.11638 | 68.31812 | 282.39160 | 90.17285 | 23.90037 |
| 5 | -0.11588 | 68.23875 | 282.23240 | 89.85742 | 28.90044 |
| 6 | -0.11613 | 68.17375 | 282.24900 | 89.89453 | 33.90052 |
| 7 | -0.11600 | 68.13438 | 282.24710 | 89.81055 | 38.90060 |
| 8 | -0.11638 | 68.10250 | 282.16410 | 89.91992 | 43.90067 |
| 9 | -0.11600 | 68.02375 | 282.12110 | 90.08691 | 48.90075 |
| 10 | -0.11625 | 67.94875 | 281.95700 | 89.87109 | 53.90083 |
| 11 | -0.11600 | 67.96313 | 282.06640 | 90.04980 | 58.90091 |
| 12 | -0.11750 | 68.02187 | 282.04000 | 90.00684 | 63.90098 |
| 13 | -0.11650 | 68.01937 | 282.10940 | 89.77637 | 68.92106 |
| 14 | -0.11613 | 68.05875 | 282.08300 | 89.55957 | 73.92114 |
| 15 | -0.11625 | 68.06188 | 282.05270 | 89.75586 | 78.92121 |



**Bild 2.2:** Darstellung eines Datensatzes im Gerät DATA.

Das zweite Tool nennt sich DIA*dem*-VIEW. Hiermit können Daten gesichtet, quantifiziert und analysiert werden. Ferner können die Daten mittels DIA*dem*-DATA als Kurven in verschiedenen Koordinatensystemen dargestellt werden. Weitere Optionen ermöglichen die Identifizierung spezieller Datenpunkte (z.B. Minimum, Maximum, etc.). Die Erstellung von Präsentationsgrafiken, die sowohl Grafiken als auch Tabellen beinhalten können, erfolgt hingegen mittels DIA*dem*-GRAPH. Hier lassen sich Layout- und Formatvorlagen für Berichte, Veröffentlichungen, etc. definieren. Die mathematischen Analysefunktionen (z.B. die Glättung, das Differenzieren bzw. Integrieren) von Funktionskurven sind in dem Modul DIA*dem*-CALC enthalten (vgl. [NID-00]).

Bei der Auswertung von Versuchsreihen gibt es zahlreiche wiederkehrende Abläufe, die mittels einer individuell auf die jeweilige Mess- und Auswerteaufgabe zusammengestellte Kombination von verschiedenen Funktionalitäten der einzelnen DIA*dem* Module abgearbeitet werden. Zur Automatisierung dieser Arbeitsabläufe bietet das Modul DIA*dem*-AUTO sämtliche benötigten Hilfsmittel. Alle Funktionen der vorher genannten Geräte lassen sich in Autosequenzen aufrufen. Somit kann eine vollständig automatisierte Aufnahme von Messwerten, deren Validierung und Auswertung, eine Genauigkeitsbetrachtung sowie vorbereitende Arbeiten für eine phänomenologische Auswertung besonders effizient gestaltet werden (vgl. [NID-00], [NID-01]).

In der zuvor angegebenen Kurzbeschreibung fehlten bislang die Gerät DIA*dem*-DAC, mit dem sich Mess-, Steuerungs- und Regelungsaufgaben realisieren lassen, sowie DIA*dem*-VISUAL zur Online-Visualisierung von Messwerten. Beide Module werden in der vorliegenden Arbeit nicht verwendet und sind daher nur aus Vollständigkeitsgründen erwähnt.

## 2.2 *Schnittstelle zur Einbindung von Nutzerroutinen*

Jedes noch so umfangreiche und komfortable Programmsystem - und dies gilt auch für die Messdatenerfassungs- und Automatisierungssoftware DIA*dem* - wird nicht alle Nutzerwünsche befriedigen können. Bei vorgesehenen Erweiterungen wird der Entwickler also beurteilen, in wieweit sich spezielle Kundenwünsche auf einer breiten kommerziellen Basis vermarkten lassen. Ist dies der Fall, so wird es ein neues Modul geben, dass analog der in Kap. 2.1 beschriebenen DIA*dem*-Geräte fakultativ vom Nutzer gekauft und auch angewendet werden kann. Ist dies hingegen nicht der Fall, so sind diese Erweiterungen vom Nutzer selbst durchzuführen und mittels sogenannter Nutzerschnittstellen an die Software anzubinden.

Da DIA*dem* vornehmlich in der Automobilbranche z.B. zur Auswertung von Crash-Tests bzw. in der Produktkontrolle eingesetzt wird, zeichnete sich bereits nach dem ersten Gespräch mit dem Entwickler ab, dass dieser kein ausreichendes Vermarktungspotential für ein Wasser/Dampf-Stoffwerte Modul sieht. Somit war klar, dass diese Arbeiten selbst von FZR durchgeführt werden müssen (vgl. [NID-02]).

Nachfolgend wird nun zuerst die in DIA*dem* bereitgestellte Schnittstelle General Programming Interface (GPI) zur Einbindung von Nutzerroutinen vorgestellt und der zur Berechnung von Wasser/Dampf-Stoffwerten in Abhängigkeit von den Eingangsgrößen notwendige Datentransfer beschrieben. Diese Schnittstelle ist in Bild 2.3 (vgl. [NID-01]) dargestellt.

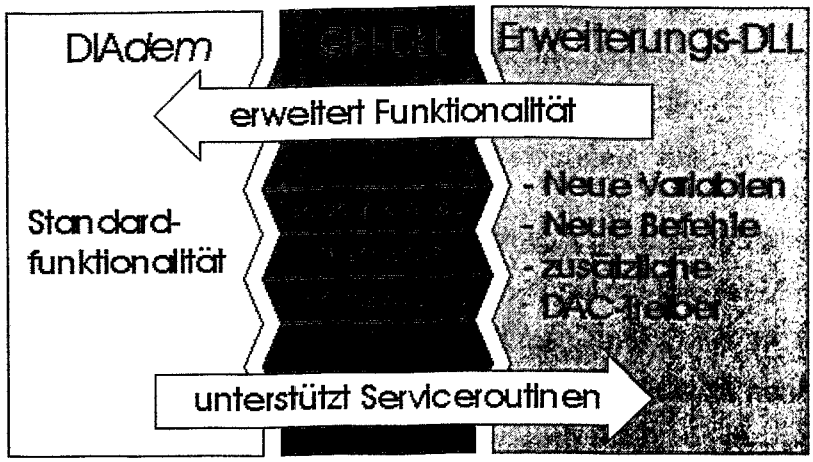



**Bild 2.3:** GPI-DLL – Schnittstelle zwischen DIA*dem* und externer Funktionen

Der linke Block „DIAdem" in Bild 2.3 stellt das Programmpaket in der vom Nutzer ausgesuchten Konfiguration dar. Hieran sind seitlich über dem in der Mitte von Bild 2.3 dargestellten General Programming Interface (GPI) die Nutzerroutinen in Form von sog. Dynamic-Link-Libraries (DLL) eingebunden. DLL´s sind ein wesentlicher Bestandteil der Architektur des Betriebssystems Windows. Die grundlegende Idee einer DLL ist es, bestimmte Funktionen allen Anwendungen zur Verfügung zu stellen und gleichzeitig den benötigten Code nur einmal zu laden und im RAM des Computers vorhalten zu müssen. Dynamic-Link-Libraries kann man sich im Prinzip wie eine Prozedurdatei vorstellen, wobei die einzelnen in der DLL enthaltenen Funktionen aus der jeweiligen Anwendersoftware heraus aufgerufen werden können. Die Voraussetzung hierfür ist jedoch, dass die benötigten Funktionsnamen sowie die in beiden Richtungen zu übergebenden Variablen bekannt sind.

Über das General Programming Interface (GPI) erfolgt der Aufruf der externen Funktionen aus der an DIA*dem* angebundenen Dynamic-Link-Library. D.h., über das GPI wird die externe Funktion aufgerufen, die notwendigen Eingangsinformationen aus dem Datenfile von DIA*dem* ausgelesen, an die jeweilige Funktion der DLL übergeben

und anschließend die Ausgangsinformationen wieder an DIA*dem* zurückgegeben. Diese Werte können anschließend mit den in Kap. 3.1 beschriebenen DIA*dem* Modulen weiterverarbeitet werden.

Diese abstrakt beschriebene Methode wird nachfolgend am Beispiel der Einbindung der zur Berechnung der Wasser/Dampfstoffwerte an der Hochschule Zittau/Görlitz entwickelten LibIF97 (vgl. [KRH-01]) nochmals im Detail beschrieben. Zum Aufruf der einzelnen Wasser/Dampf-Stoffwertefunktionen werden zunächst die zur Berechnung der Stoffwerte benötigten Eingangsgrößen im GPI definiert. Die Eingangsgrößen können die Zustandgrößen Druck p, Temperatur T, Umgebungstemperatur $T_U$, Dampfgehalt x, Enthalpie h oder Entropie s sein. Eine genaue Aufstellung der für den Aufruf der jeweiligen Stoffwertefunktion benötigten Eingangsgrößen ist Tab. 2.1 zu entnehmen. In dieser Tabelle sind ferner die Namen der Funktionsaufrufe sowie die Einheiten, in denen die jeweiligen Parameter übergeben werden, mit aufgeführt. Anschließend wird seitens der LibIF97 nach der Berechnung der jeweilige Ergebniswert an DIA*dem* zurückgegeben.

**Tab. 2.1:** Zusammenstellung der Übergabeparameter, Funktionsnamen und Einheiten für den Datentransfer zwischen DIA*dem* und den Stoffwertefunktionen der LibIF97.

| Funktion | Eingabevariablen | Aufruf in DIA*dem* | Maßeinheit des Ergebnisses |
|---|---|---|---|
| Temperaturleitfähigkeit | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_a_ptx_97 | $\frac{m^2}{s}$ |
| Spezifische isobare Wärmekapazität | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_cp_ptx_97 | $\frac{kJ}{kg \cdot K}$ |
| Spezifische isochore Wärmekapazität | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_cv_ptx_97 | $\frac{kJ}{kg \cdot K}$ |
| Differentialquotient $\left(\frac{\partial v}{\partial p}\right)_T (p,t,x)$ | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_dvdp_t_ptx_97 | $\frac{m^3}{kg \cdot kPa}$ |
| Differentialquotient $\left(\frac{\partial v}{\partial T}\right)_p (p,t,x)$ | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_dvdt_p_ptx_97 | $\frac{m^3}{kg \cdot K}$ |

**Tab. 2.1:** Zusammenstellung der Übergabeparameter, Funktionsnamen und Einheiten für den Datentransfer zwischen DIA*dem* und den Stoffwertefunktionen der LibIF97 (Fortsetzung 1).

| Funktion | Eingabevariablen | Aufruf in DIA*dem* | Maßeinheit des Ergebnisses |
|---|---|---|---|
| Spezifische Energie | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg], Umgebungstemperatur [°C] | WD_e_ptx_tu_97 | $\frac{kJ}{kg}$ |
| Dynamische Zähigkeit | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_eta_ptx_97 | $Pa \cdot s$ |
| Umkehrfunktion: Enthalpie | Druck [bar], Entropie [kJ/(kg*K)] | WD_h_ps_97 | $\frac{kJ}{kg}$ |
| Spezifische Enthalpie | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_h_ptx_97 | $\frac{kJ}{kg}$ |
| Isentropenexponent | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_kappa_ptx_97 | *[-]* |
| Wärmeleitfähigkeit | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_lambda_ptx_97 | $\frac{W}{m \cdot K}$ |
| Kinematische Viskosität | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_nue_ptx_97 | $\frac{m^2}{s}$ |
| Umkehrfunktion: Druck | Enthalpie [kJ/kg], Entropie [kJ/(kg*K)] | WD_p_hs_97 | *bar* |
| Prandtl-Zahl | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_pr_ptx_97 | *[-]* |
| Dampfdruck | Temperatur [°C] | WD_ps_t_97 | *bar* |
| Dichte | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_rho_ptx_97 | $\frac{kg}{m^3}$ |
| Umkehrfunktion: Entropie | Druck [bar], Enthalpie [kJ/kg] | WD_s_ph_97 | $\frac{kJ}{kg \cdot K}$ |
| Spezifische Entropie | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_s_ptx_97 | $\frac{kJ}{kg \cdot K}$ |
| Oberflächenspannung | Druck [bar] | WD_sigma_p_97 | *mPa m* |
| Oberflächenspannung | Temperatur [°C] | WD_sigma_t_97 | *mPa m* |
| Umkehrfunktion: Temperatur | Enthalpie [kJ/kg], Entropie [kJ/(kg*K)] | WD_t_hs_97 | *°C* |

**Tab. 2.1:** Zusammenstellung der Übergabeparameter, Funktionsnamen und Einheiten für den Datentransfer zwischen DIA*dem* und den Stoffwertefunktionen der LibIF97 (Fortsetzung 2).

| Funktion | Eingabevariablen | Aufruf in DIA*dem* | Maßeinheit des Ergebnisses |
|---|---|---|---|
| Umkehrfunktion: Temperatur | Druck [bar], Enthalpie [kJ/kg] | WD_t_ph_97 | $°C$ |
| Umkehrfunktion: Temperatur | Druck [bar], Entropie [kJ/(kg*K)] | WD_t_ps_97 | $°C$ |
| Siedetemperatur | Druck [bar] | WD_ts_p_97 | $°C$ |
| Spezifische innere Energie | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_u_ptx_97 | $\frac{kJ}{kg}$ |
| Umkehrfunktion: spezifisches Volumen | Druck [bar], Enthalpie [kJ/kg] | WD_v_ph_97 | $\frac{m^3}{kg}$ |
| Umkehrfunktion: spezifisches Volumen | Druck [bar], Entropie [kJ/(kg*K)] | WD_v_ps_97 | $\frac{m^3}{kg}$ |
| Spezifisches Volumen | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_v_ptx_97 | $\frac{m^3}{kg}$ |
| Isentrope Schallgeschwindigkeit | Druck [bar], Temperatur [°C], Dampfgehalt [kg/kg] | WD_w_ptx_97 | $\frac{m}{s}$ |
| Umkehrfunktion: Dampfanteil | Enthalpie [kJ/kg], Entropie [kJ/(kg*K) | WD_x_hs_97 | $\frac{kg}{kg}$ |
| Umkehrfunktion: Dampfanteil | Druck [bar], Enthalpie [kJ/kg] | WD_x_ph_97 | $\frac{kg}{kg}$ |
| Umkehrfunktion: Dampfanteil | Druck [bar], Entropie [kJ/(kg*K) | WD_x_ps_97 | $\frac{kg}{kg}$ |

Die Programmierung der Schnittstelle zur Einbindung der Wasser/Dampf-Stoffwertebibliothek erfolgt in Borland Delphi 6. In dieser Schnittstelle sind sämtliche zu übergebenden Variablen sowie die Funktionsaufrufe der einzelnen Stoffwerteroutinen - für jede Funktion der Wasser/Dampf-Stoffwertebibliothek, die in DIA*dem* aufgerufen werden soll, muss dies separat erfolgen - zu definieren. Nachfolgend wird diese Defi-

nition am Beispiel des Funktionsaufrufs zur Berechnung der spezifischen Enthalpie beschrieben.

Zur Berechnung der spezifischen Enthalpie für Wasser, Dampf und Zweiphasengemische in Abhängigkeit vom Druck p, der Temperatur T sowie dem Dampfgehalt x wird zuerst der Funktionsname "WD_h_ptx_97" definiert (s. Tab. 2.1). Mit diesem wird aus DIA*dem* die entsprechende Funktion über die GPI-Schnittstellendatei commands.dll aus der Stoffwertebibliothek Lib97.dll aufgerufen. Die Datei commands.dll ist das compilierte und nun ausführbare File der Quelldatei commands.pas, die eine Sammlung der notwendigen Definitionen und allgemeinen Funktionen zur rechentechnischen Umsetzung enthält.

Die Definition von "WD_h_ptx_97" verweist innerhalb von commands.pas auf die Funktion "Fct_h_ptx_97", die die einzelnen Rechenschritte vom Einlesen der Parameter bis zur Ausgabe der Ergebnisse verwaltet. Dazu werden wiederum die in der Datei berechnung.pas enthaltenen allgemeinen Prozeduren

hptx_lesen(p, t, x) und

hptx_schreiben(h_ptx_97(p, t, x))

aufgerufen. In der Prozedur hptx_lesen werden die aus DIA*dem* an commands.dll übergebenen Werte der Parameter Druck, Temperatur und Dampfgehalt in für die Wasser/Dampfbibliothek Lib97.dll benötigten Eingabevariablen umgeleitet. Daran anschließend wird die Prozedur „hptx_schreiben" aufgerufen, die den Aufruf der Wasserdampfbibliothek Lib97.dll enthält. Aus den nun bekannten Eingabevariablen wird der gewünschte Wert Enthalpie berechnet und als Ergebnisvariable der Prozedur hptx_schreiben für die Rückgabe an DIA*dem* gespeichert. In Anhang A ist der vollständige Quellcode aufgelistet.

Bevor die Wasser/Dampf-Stoffwertebibiothek in DIA*dem* genutzt werden kann, muss die neue Datei commands.dll im Programmsystem angemeldet werden. Dies geschieht über den Aufruf des Unterpunktes GPI-DLL Registrierung im Menü Einstellungen des Ausgangsbildschirms von DIA*dem*. In der dortigen Tabelle wird die Dyna-

mic-Link-Library commands.dll eingetragen. Nach einem Neustart von DIA*dem* steht dann der Befehlssatz dem Anwender zur Verfügung

## 2.3 *Aufruf der Wasser/Dampf-Stoffwertefunktionen in Autosequenzen*

Nachdem nun die Wasser/Dampf-Stoffwertebibliothek in DIA*dem* verfügbar ist, soll diese mittels Autosequenzen angesprochen werden können. Erst mit diesem Schritt wird eine effiziente und automatisierte Auswertung der Messdaten ermöglicht.

Der prinzipielle Aufruf einer Wasser/Dampf-Stoffwertefunktion in einer Autosequenz wird anhand des Beispiels „Berechnung der Enthalpie für Wasser, Dampf oder Zweiphasengemisch" beschrieben. In dem konkreten Beispiel soll dies anhand der Berechnung der Enthalpie für eine unterkühlte Flüssigkeit bei einem Druck p von 2 bar und einer Temperatur T von 100 °C erfolgen. Hierzu sind zuerst die Eingangsvariablen ChnAttribVal(1), ChnAttribVal(2) und ChnAttribVal(3) gemäß Tab. 2.1 mit den Werten für den Druck p, Temperatur T und Dampfgehalt x zu belegen, d.h.

**ChnAttribVal(1) = 2**

**ChnAttribVal(2) = 100**

und

**ChnAttribVal(3) = -1.**

Die Belegung der Variabalen ChnAttribVal(3) mit dem Wert -1 hat folgende Bewandnis: die Stoffwertebibiliothek Lib97.dll unterscheidet die Systemzustände unterkühlte bzw. gesättigte Flüssigkeit, Zweiphasengebiet und Sattdampf bzw. überhitzter Dampf. Für alle Parameterkombinationen von Druck und Temperatur außerhalb des Nassdampfgebiets wird der Dampfgehalt x mit dem Wert -1 belegt.

Als nächstes erfolgt der Aufruf der Wasser/Dampf-Stoffwertefunktion mit dem in der Schnittstellendatei commands.dll definierten Funktionsaufruf für die Enthalpie:

**WD_h_ptx_97.**

Das Ergebnis, d.h. die Enthalpie bei dem aktuellen Zustand (hier 419,1 KJ/kg), wird anschließend unter der Variablen

**ChnAttribVal(4)**

zurückgegeben.

Für die automatisierte Auswertung der Messdatenfiles muss diese Vorgehensweise nicht nur auf einen einzelnen Zeitpunkt des Messdatenfiles, sondern auf das gesamte Messintervall angewendet werden. Die hierzu notwendige Steuerung des Datenflusses erfolgt mit Hilfe der Befehle des DIA*dem* Gerät DATA. Hierauf wird in Kapitel 3 der vorliegenden Arbeit im Detail eingegangen.

# 3 Programmierung von Autosequenzen in MS Visual Basic Script

Nachdem in Kap. 2 die Handhabung und Anbindung der Dynamic-Link-Library Lib97.dll sowie der Aufbau und der zur Anbindung der Lib97 an DIA*dem* benötigten GPI-Schnittstelle beschrieben wurde, widmet sich das dritte Kapitel der effizienten Handhabung der Daten. Dies soll vornehmlich automatisiert mit Hilfe sog. Autosequenzen erfolgen.

Das Kapitel beginnt zunächst mit einer Beschreibung des Geräts DIA*dem*-AUTO, einer Schnittstelle, die dem Nutzer die Möglichkeit bietet, mittels selbst erstellter Autosequenzen alle in DIA*dem* vorhandenen Geräte zu steuern. Im zweiten Teil werden die wichtigsten Befehle der Autosequenzen beschrieben. Die Anwendung dieser Grundlagen, d.h. die Anwendung auf ein konkretes Beispiel, wird in Kap. 4 beschrieben.

Die Programmierung der Autosequenzen erfolgt in der Programmiersprache Microsoft Visual Basic Script (MS VB Script), die derzeit wohl meistgenutzte Skriptsprache für Windows. Die Verwendung von Microsoft VB Script bietet bei der Erstellung von Autosequenzen dem Anwender eine Vielzahl hilfreicher Funktionen und eine weitaus höhere Leistung als die konventionelle Autosequenzsprache von DIA*dem*. Hierbei ist speziell der komfortable Debugger zu nennen, der das Auffinden von Fehlern in Skripten sehr vereinfacht. VB Script-Autosequenzen können wahlweise aus dem Editor heraus gestartet werden oder von der jeweiligen Autosequenzdatei.

## *3.1 Grundlagen von DIAdem-AUTO*

DIA*dem*-AUTO stellt einen Editor innerhalb von DIA*dem* dar, welcher MS Visual Basic Script interpretieren kann und die Möglichkeit bietet, alle in DIA*dem* vorhandenen Geräte (DATA, VIEW, GRAPH, CALC, DAC, VISUAL - vgl. Kap. 2.1) zu steuern. Somit können diverse Arbeitsabläufe, wie zum Beispiel die Aufnahme, Validierung und Auswertung von Messwerten, automatisiert werden.

In Bild 3.1 ist die Standardoberfläche von DIA*dem*-AUTO dargestellt. Sie teilt sich in drei Blöcke auf: auf der linken Seite befindet sich der sogenannte Datei-Browser, in dem alle geöffneten Skripte namentlich aufgelistet sind. Rechts befindet sich der Editor, der den Quellcode der jeweils ausgewählten Autosequenz enthält. Im unteren Teil schließt sich ein Systemfenster an, in dem DIA*dem* Dateioperationen oder auch Fehlermeldungen anzeigt.

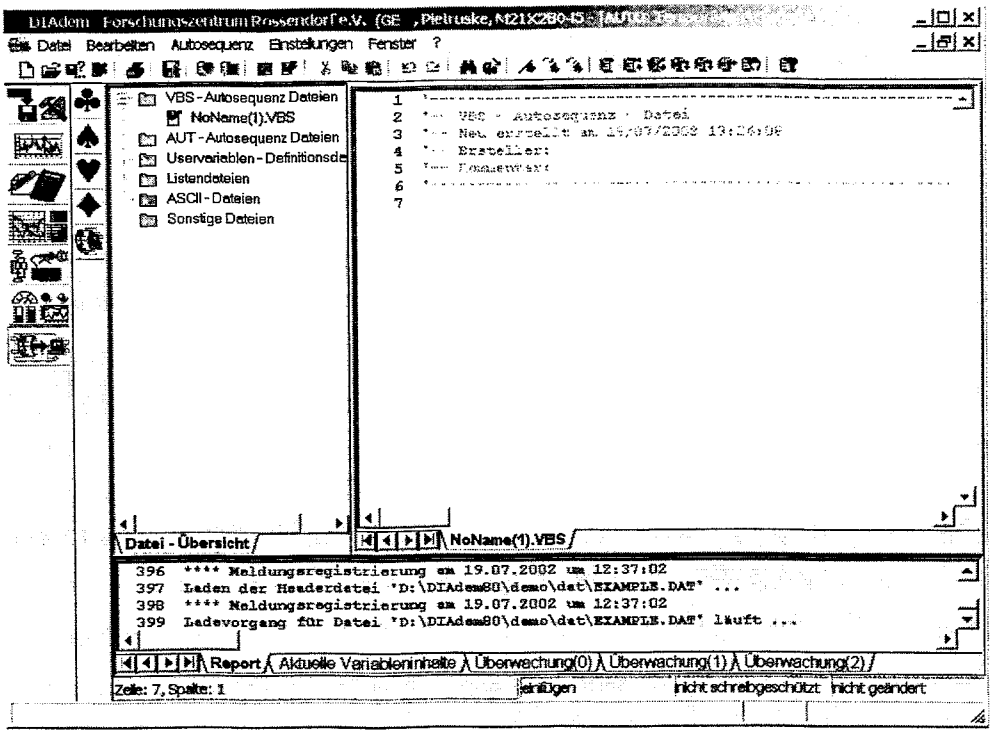



**Bild 3.1:** Oberfläche von DIA*dem*-AUTO.

Die Erstellung einer Autosequenz kann auf zwei verschiedenen Wegen geschehen. Dies sind der sog. Teach-In-Modus sowie die Erstellung von VB Scripten. Nach Aktivierung des Teach-In-Modus werden alle relevanten Aktionen des Anwenders von DIA*dem* aufgezeichnet und in einer von DIA*dem* generierten Autosequenz abgespeichert. Diese Sequenz ermöglicht es, die vom Anwender vorgezeigten Schritte beliebig oft zu wiederholen. Dieser Teach-In-Modus besitzt darüber hinaus den Vorteil, dass der Anwender keine Programmierkenntnisse von VB Scripten besitzen muss - allerdings lassen sich im Teach-In-Modus nur einfache Vorgänge (z.B. die Verknüpfung verschiedener Kanäle in DIA*dem*-DATA) automatisieren.

Im Teach-In-Modus ist es nicht möglich, nutzerdefinierte Funktionen aus Dynamic-Link-Libraries aufzuzeichnen. Daher müssen die für die Auswertung der TOPFLOW Datenfiles notwendigen Autosequenzen selbst entworfen und in VB Scripten realisiert werden. Die hierzu benötigten Grundlagen sind im nachfolgenden Unterkapitel beschrieben.

Nach Ihrer Erstellung werden die Autosequenzen aus der in Bild 3.1 dargestellten Oberfläche von DIA*dem*-Auto gestartet. Dabei können im Hintergrund sämtliche Funktionen der anderen DIA*dem*-Geräte in der Autosequenz genutzt werden. Die Ergebnisse bei der Ausführung der Autosequenzen werden auf dem Monitor angezeigt, können alternativ aber auch in Dateien abgelegt oder auf einem Drucker ausgegeben werden.

## 3.2 *Ausgewählte Befehle für Autosequenzen*

Nachfolgend werden ausgewählte Befehle für Autosequenzen vorgestellt. Diese müssen in ein VB Script integriert werden. Die Auswahl der nachfolgend vorgestellten Befehle orientiert sich an den Anforderungen der vorliegenden Arbeit und umfasst den Befehlsumfang der in Kap. 4 vorgestellten Autosequenzen zur Auswertung eines ausgewählten NOKO-Versuchs. Generell wird ein Befehl in einem VB Script mit dem Vorsatz „Call" aufgerufen.

Mit dem Befehl

**Call DataLoad("Example1.dat")**

wird ein bestehender Datensatz durch die Autosequenz geöffnet und die Daten der geöffneten Datei (hier die Datei „Example1.dat") zur weiteren Bearbeitung eingelesen. Der Befehl zum Speichern des (bearbeiteten) Datensatzes heißt

**Call DataSave(„Example2.dat") .**

Die verschiedenen Kanäle (vgl. Kap. 2.1) lassen sich mit Hilfe des Befehls FormulaCalc beliebig miteinander verknüpfen, wie z.B. in dem nachfolgenden Beispiel, in dem der Kanal „Ergebnis" aus den Werten der Kanäle „Eingabe1", „Eingabe2" und „Eingabe3" gebildet wird:

**Call FormulaCalc("Ch('Ergebnis') := 'Eingabe1' * ( 'Eingabe2' - 'Eingabe3' ) ") .**

Ein weiteres Beispiel ist die Multiplikation eines Kanals mit einer Konstanten (hier die Zahl Pi, die als Konstante in DIA*dem* zur Verfügung steht). Dies geschieht mit Hilfe der Befehlszeile

**Call FormulaCalc("Ch('Ergebnis') := 'Eingabe1' * PI() .**

In VB Scripten können auch Schleifen verwendet werden. Hierzu zählt die sog. „Do-While"-Schleife, für die nachfolgend ein Beispiel angeführt ist:

```
L1=1

    Do While L1<=3

        MSGLINEDISP( "Der Wert L1 betraegt: "&str(L1)

    L1=L1+1

Loop
```

L1 ist die Zählvariable, welche am Anfang auf eins zurückgesetzt wird. Die nächste Zeile setzt den Beginn der Schleife. Sie wird so lange durchgeführt, bis der Zähler L1 kleiner oder gleich 3 ist. Der Befehl MSGLINEDISP zeigt eine Information (z.B. beim zweiten Durchlaufen der Schleife: *Der Wert L1 beträgt: 2* an.) in der Statusleiste der DIA*dem*-Oberfläche an.

Um einen Wert einer einzelnen Zelle des Datensatzes auslesen zu können, wird folgende Formel verwendet:

**R1 = chd(20,cno("Eingabe1"))**

Es wird der Wert der Zelle der Zeile 20 des Kanals „Eingabe1" unter der Variablen R1 abgelegt. Alternativ kann statt der konstanten Zeile 20 auch eine Zählvariable eingebunden werden. Dies ermöglicht in Verbindung mit einer Schleifenfunktion eine Serienauswertung.

In Kapitel 4 wird zusätzlich anhand des Beispiels „Auswertung eines NOKO-Experiments“ die Anwendung der o.g. Befehlen nochmals im Detail an einem konkreten Beispiel erklärt.

# 4 Auswertung eines ausgewählten NOKO-Datensatzes

Zur Validierung der in Kap. 2 beschriebenen Schnittstelle zur Einbindung der Wasser/Dampf-Stoffwertebibliothek LibIF97 sowie speziell des Zusammenspiels dieser Dynamic-Link-Library mit den in Kap. 3 beschriebenen Autosequenzen, wird nun ein erster Funktionstest (Auswertung eines ausgewählten NOKO-Experiments) durchgeführt. Ziel dieser Validierung ist, für die Inbetriebnahmeversuche von TOPFLOW ein getestetes Tool für die Datenauswertung bereitzustellen, erste Erfahrungen hiermit zu sammeln und einen Einarbeitungsstand zu erreichen, der eine schnelle Adaption unter den bei der Inbetriebnahme herrschenden Bedingungen (Zeitdruck, Hektik, Anwesenheit diverser Firmen, Abnahmen, Lärm, etc.) erlaubt.

Dieses Kapitel beinhaltet in Kap. 4.1 eine kurze Beschreibung des NOKO-Versuchsstandes. Des weiteren wird in Kap. 4.2 ein Messdatenfile eines NOKO-Experiments mittels Autosequenz in DIA*dem* ausgewertet und so die korrekte Funktionsweise der implementierten Schnittstelle und die Kopplung der Wasser/Dampf-Stoffwertebibliothek LibIF97 an DIA*dem* nachgewiesen. Anschließend sind exemplarisch in Kap. 4.3 Ergebnisse der Auswertungen vorgestellt.

## *4.1 Kurzbeschreibung NOKO*

Der Name NOKO leitet sich aus dem Begriff Notkondensator ab. Dieser ist eine passive Komponente des von Framatome ANP entwickelten innovativen Reaktorkonzepts SWR1000 (vgl. [SCA-961]). Diesem Kraftwerkstyp werden derzeit gute Chancen eingeräumt, in Finnland als dann fünftes Kernkraftwerk realisiert zu werden.

Eine wichtige Komponente dieses Reaktorkonzeptes ist der Notkondensator, dessen Aufbau und Wirkungsweise in Bild 4.1 dargestellt ist. Der Notkondensator besteht aus einem direkt an den Reaktordruckbehälter (RDB) angeschlossenen Wärmetauscherbündel, das in einem seitlich vom RDB im Containment gelegenen Wasserpool angeordnet ist. Die Temperatur dieses Wasser beträgt ca. 30 °C. Bei einer Absenkung des Wasserspiegels im RDB als Folge eines Lecks oder Abblasen von Dampf werden die Rohre des Notkondensatorsystems freigelegt. In den Rohren kondensiert der Dampf, wobei die dabei freiwerdende Wärme an das geodätische Flutbecken

abgegeben wird. Durch Gravitation fließt das kondensierte Wasser zurück in den RDB und es entsteht ein zweiphasiger Naturumlauf. Deshalb ist dieses System eine passive Komponente, d.h. sie funktioniert selbständig ohne externe Energiezufuhr und ist unabhängig von Operatorhandlungen.

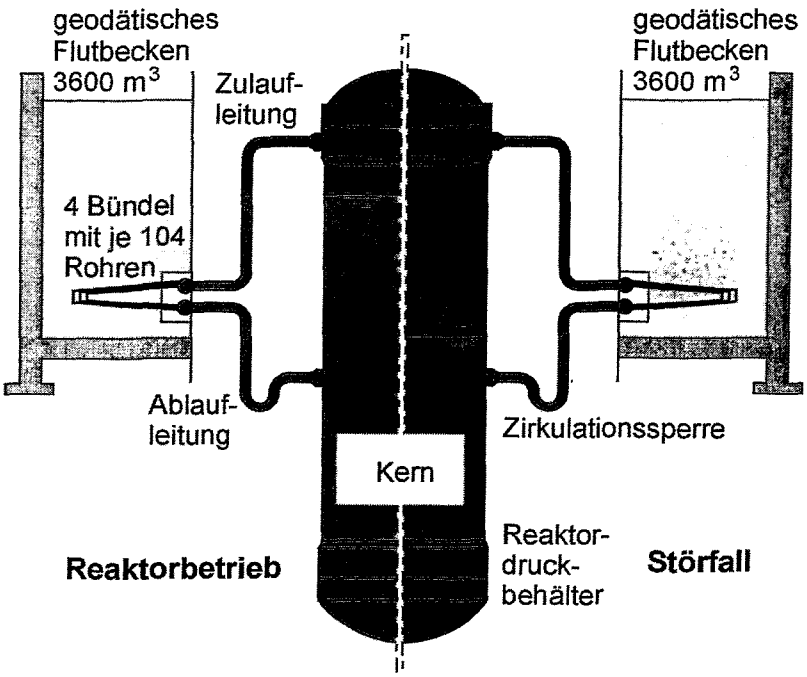



**Bild 4.1:** Funktionsweise des Notkondensators des SWR600/1000.

Zur Untersuchung des Betriebsverhaltens dieses Kondensators wurde 1993 am Institut für Sicherheitsforschung und Reaktortechnik (ISR) der Forschungszentrum Jülich GmbH mit Unterstützung des Bundesministeriums für Forschung und Technologie (BMFT), dem Verband der Elektrizitätswirtschaft (VDEW) und Siemens der NOKO-Versuchsstand errichtet.

Das Anlagenschema des Notkondensatorversuchstandes ist in Bild 4.2 dargestellt. Der RDB wird durch das Druckgefäß simuliert, welches zur Einstellung der Freilegung der Kondensatorrohre während der Experimente dient. Der für die Experimente benötigte Wasserdampf wird durch einen Elektrokessel mit einer maximalen Leistung von 4 MW und in Form von Nassdampf mit einem Dampfgehalt von bis zu 25% erzeugt. Die Phasen dieses Nassdampfes werden anschließend in dem Separator ge-

trennt und zur Testsektion (Dampf) oder zurück in den Elektrokessel (Wasser) geleitet.

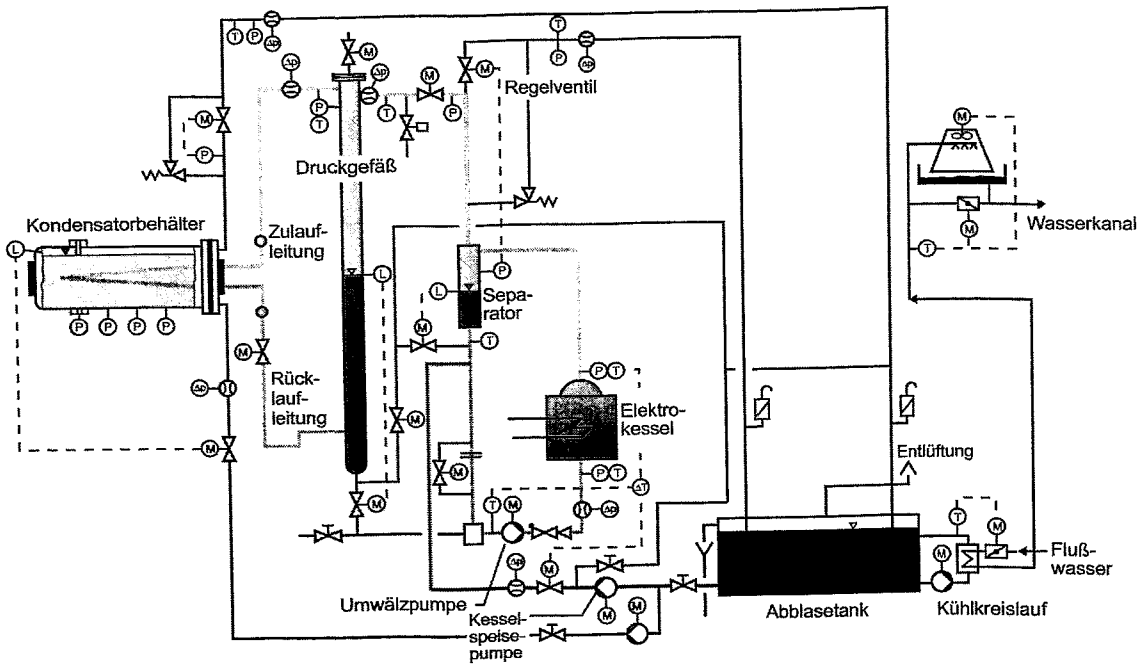



**Bild 4.2:** Anlagenschema des NOKO-Versuchsstands.

Die eigentliche Testsektion besteht aus dem Druckgefäß, einem Modell des RDB, an welches die Kondensatorvor- und -rücklaufleitungen seitlich angeschlossen sind. Das Testbündel besteht aus 8 im Mittel 9,8 m langen, gegenüber der Horizontalen leicht geneigten Rohren und ist im Kondensatorbehälter, der das geodätische Flutbecken simuliert, angeordnet. Das Testbündel ist aus dem Originalmaterial gefertigt und besitzt die Originalgeometrie. Die Skalierung wurde somit nur über die Reduktion der Anzahl der Rohre (4 Bündel á 104 Rohre im SWR600/1000, 8 Rohre im NOKO Versuchsstand) im Verhältnis 1:52 durchgeführt. Aufgrund der großen Effektivität der Wärmeübertragung wurde das NOKO-Bündel am FZJ aber nur mit 4 Rohren betrieben. In diesem Fall besitzt der Skalierungsfaktor den Wert 1:104.

Neben den Notkondensatorversuchen wurden in NOKO u.a. auch Experimente zu weiteren passiven Komponenten (5 verschiedenen Varianten von passiven Impulsgebern, Gebäude- und Plattenkondensatoren, Notkondensatorbündel mit einem optimierten Design) sowie Grundlagenexperimente zur Kondensation von Gemischen

aus Wasserdampf und nichtkondensierbaren Gasen in waagerechten Rohren durchgeführt (vgl. [SCA-962]).

Für die Validierung der Einbindung der Wasser/Dampf-Stoffwertebibliothek LibIF97 sowie speziell des Zusammenspiels dieser Dynamic-Link-Library mit den Autosequenzen wurde der Versuchsdatensatz des Experimentes NOKO A3 ausgewählt. Während dieses Versuchs wurde bei einem Druck im Druckgefäß von 7 MPa der Füllstand im Druckbehälter auf 3,7 m abgesenkt. Der Druck im Kondensatorbehälter betrug 0,1 MPa.

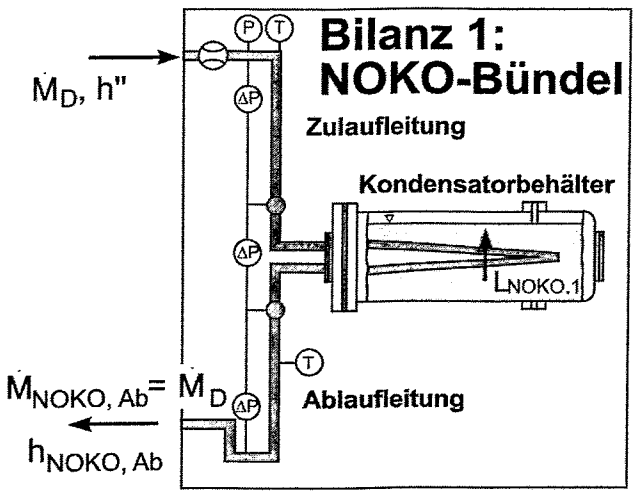



**Bild 4.3:** Energiebilanz zur Berechnung der Kondensatorleitung.

Die während des Experiments im Kondensator übertragene Leistung wird anhand einer Energiebilanz für das Testbündel berechnet. Es gilt:

$$L_{NOKO,1} = \dot{M}_{NOKO} \cdot (h'' - h_{NOKO,Ab}) \qquad \text{(Gl. 1)}$$

In obiger Gleichung kennzeichnet $L_{NOKO}$ die Leistung des Notkondensators, $\dot{M}_{NOKO}$ den Massenstrom, h" die Enthalpie des Dampfes sowie $h_{NOKO,Ab}$ die Enthalpie des Kondensats. Der Massenstrom $\dot{M}_{NOKO}$ berechnet sich aus dem an der Messstelle in der Dampfleitung gemessenen Druckverlust PD2_1 wie folgt:

$$\dot{M}_{NOKO} = 0{,}1041 \cdot \sqrt{\rho_{NOKO} \cdot p_{PD2\_1}} \qquad \text{(Gl. 2)}$$

$$\rho_{NOKO} = \rho(p \text{ bei } P2\_1A,\ T_{S,NOKO} \text{ bei } P2\_1A) \qquad \text{(Gl. 3)}$$

$$p_{P2\_1A} = p_{P2\_1} + 1 \qquad \text{(Gl. 4)}$$

Bild 4.3 ist die Lage der Messstellen in der Dampf- sowie in der Kondensatleitung zu entnehmen. P2_1A gibt den absoluten Druck innerhalb des Kreislaufes wieder, dieser wird als Differenz zur Umgebung mittels der P2_1 gemessen. T2_13 stellt die Temperatur des Dampfes in der Zu-, T2_14 die Temperatur des Kondensats in der Ablaufleitung des Bündels dar. Die Enthalpie der Zulaufleitung bestimmt sich aus der Temperatur T2_13 und dem Dampfgehalt von x=1. Die Enthalpie des Rücklaufes wird aus dem Druck P2_1 und der Temperatur T2_14 bestimmt. Das folgende Kapitel gibt den Quellcode der Autosequenz zur Berechnung der Kondensatorleistung wieder.

## *4.2 Quellcode der Autosequenz*

Der vorliegende Quellcode dient zur Berechnung der Leistung des Kondensators und zur Erstellung und Ausgabe eines Diagramms, in dem die Kondensatorleistung über der Versuchszeit aufgetragen ist. Dieses Unterkapitel beschreibt blockweise die einzelnen Anweisungen innerhalb des Quellcodes. Zur besseren Übersichtlichkeit werden hierzu die am linken Rand der Listings angegebenen Zeilennummern verwendet.

Die Zeilen 1 bis 7 sind Kommentarzeilen und beinhalten eine Beschreibung der Autosequenz (Autor, Erstellungsdatum). Folglich werden diese nicht ausgeführt.

```
001   '--------------------------------------------------------------------
002   '-- VBS - Autosequenz - Datei
003   '-- Neu erstellt am 30/07/2002 16:13:30
004   '-- Ersteller: Heiko Pietruske
005   '-- Kommentar: Auswertung NOKO-Datensatz -> Bestimmung LKOND1
006   '--            als Teil des Semesterbelegs
007   '--------------------------------------------------------------------
```

In Zeile 8 wird der Messdatensatz mit den Kanälen „Zeit", „PD2_1", „P2_1", „T2_13" und „T2_14" geladen. Als nächstes wird in Zeile 9 aus dem gemessenen

Differenzdruck zur Umgebung der Absolutdruck berechnet und in dem Kanal „P2_1A" gespeichert.

```
008   Call DataLoad("NOKOLEISTUNG.dat")
009   Call FormulaCalc("Ch('P2_1A') := 'P2_1' + 1")
010
```

In den Zeilen 11 bis 30 erfolgt die Berechnung der Dichte mittels der implementierten Wasserdampftafel. In Zeile 12 wird die Kanalnummer des nächsten freien Kanals gesucht und unter der Variablen L1 abgelegt. Zeile 13 definiert den freien Kanal als „RHO_NOKO". Damit DIA*dem* in einen Kanal Daten schreiben kann, muss die Kanallänge definiert werden. In Zeile 14 wird dann die Länge des Kanals „Zeit" abgefragt und dieser Wert auch der Spalte „RHO_NOKO" zugewiesen. Die nächsten 15 Zeilen bilden eine Do-While-Schleife. Die Wasserdampf-Tafel kann immer nur einen Zustand berechnen und somit muss jede Zeile einzeln aufgerufen werden. Die Schleife geht jede einzelne Zeile durch und liest die gegebenen Daten ein.

```
011   '-- Berechnung RHO_NOKO
012   L1 = CNo("free")
013   ChnName(L1) = "RHO_NOKO"
014   cl(L1)= cl(cno("Zeit"))
015
016   lCountT = 1
017   Do While lCountT <=  cl(L1)
018
```

Zur Berechnung der Dichte sind 3 Eingabewerte notwendig. Dies sind Druck, Temperatur und Dampfgehalt. In Zeile 19 wird der Wert aus Kanal „P2_1A" unter der Variablen ChnAttribVal1(1) abgelegt. Die Zeilennummer entspricht dabei dem aktuellen Wert der Schleifenzählvariable LCountT. Da der Dampf im Zulauf gesättigt ist, wird die Übergabevariable für die Temperatur ChnAttribVal2(1) mit dem Wert -1 belegt. Der Wert 1 für den Dampfgehalt x wird als Variable ChnAttribVal3(1) abgelegt.

```
019     ChnAttrVal1(1) = chd(lCountT,cno("P2_1A"))
020     ChnAttrVal2(1) = -1
021     ChnAttrVal3(1) = 1
022
```

Der Befehl in Zeile 23 gibt in der Statusleiste aus, welche Zeile aktuell von der Autosequenz abgearbeitet wird. In Zeile 25 wird dann die externe Funktion zur Berechnung der Dichte aufgerufen und in Zeile 27 der durch die Wasserdampftafel berechnete Wert in der jeweilige Zeile des Kanals „RHO_NOKO" gespeichert.

```
023     msglinedisp( "Kopiere Wert No "&str(lcountT))
024
025     WD_rho_ptx_97
026
027     chd(lCountT,cno("RHO_NOKO"))=ChnAttrVal4(1)
028
```

In Zeile 29 wird der Zähler der Schleife um 1 erhöht. Diese Schleife wird solange durchlaufen, bis der Zähler den Wert der Kanallänge überschreitet. In Zeile 30 wird die Schleife beendet.

```
029      lCountT = lCountT + 1
030   Loop
031
```

Die Zeilen 32 bis 44 führen die Berechnung des Massenstroms $\dot{M}_{NOKO}$ mittels der in Kapitel 4.1 beschriebenen Gleichung 2 durch. In Zeile 33 wird der nächste freie Kanal gesucht und in Zeile 34 mit dem Namen „M_NOKO" bezeichnet. In Zeile 35 wird die Länge dieses Kanals definiert. In der Schleife von Zeile 37 bis 44 werden die Daten zeilenweise ausgewertet. Hierzu werden die entsprechenden Daten aus den

Kanälen „RHO_NOKO" und „PD2_1" ausgelesen, in Zeile 40 hieraus der Massenstrom berechnet und dieser Wert in Zeile 41 in den Kanal „M_NOKO" geschrieben.

```
032   '-- Berechnung M_NOKO
033   L1 = CNo("free")
034   ChnName(L1) = "M_NOKO"
035   cl(L1)= cl(cno("Zeit"))
036   lCountT = 1
037   Do While lCountT <=  cl(L1)
038     R1 = chd(lCountT,cno("RHO_NOKO"))
039     R2 = chd(lCountT,cno("PD2_1"))
040     Call FormulaCalc("R3 := 0.1041 * sqrt( R1 * -R2 )")
041     chd(lCountT,cno("M_NOKO")) = R3
042     msglinedisp( "Bearbeite Wert No "&str(lcountT))
043     lCountT = lCountT + 1
044   Loop
045
```

In den folgenden Zeilen 46 bis 65 und 67 bis 86 werden die Enthalpien analog zur Berechnung der Dichte in den Zeilen 11 bis 30 durchgeführt.

```
046   '-- Berechnung HD_T2_13
047   L1 = CNo("free")
048   ChnName(L1) = "HD_T2_13"
049   cl(L1)= cl(cno("Zeit"))
050
051   lCountT = 1
052   Do While lCountT <=  cl(L1)
053
054     ChnAttrVal1(1) = -1
055     ChnAttrVal2(1) = chd(lCountT,cno("T2_13"))
056     ChnAttrVal3(1) = 1
057
058     msglinedisp( "Kopiere Wert No "&str(lcountT))
059
060 WD_h_ptx_97
061
```

```
062  chd(lCountT,cno("HD_T2_13"))=ChnAttrVal4(1)
063
064  lCountT = lCountT + 1
065   Loop
066
067   '-- Berechnung H_NOKO_AB
068   L1 = CNo("free")
069   ChnName(L1) = "H_NOKO_AB"
070   cl(L1)= cl(cno("Zeit"))
071
072   lCountT = 1
073   Do While lCountT <=  cl(L1)
074
075     ChnAttrVal1(1) = chd(lCountT,cno("P2_1"))
076     ChnAttrVal2(1) = chd(lCountT,cno("T2_14"))
077     ChnAttrVal3(1) = -1
078
079     msglinedisp( "Kopiere Wert No "&str(lcountT))
080
081     WD_h_ptx 97
082
083     chd(lCountT,cno("H_NOKO_AB"))=ChnAttrVal4(1)
084
085     lCountT = lCountT + 1
086   Loop
087
```

In Zeile 88 wird nun abschließend die Leistung des Kondensator gemäß Gl. 1 aus Kap. 4.1 berechnet.

```
088   FormulaCalc("Ch('L_NOKO'):= M_NOKO' * ( 'HD_T2_13' - 'H_NOKO_AB' )")
089
090
091   '-----------------------------------------------------------------
092   'Ende Berechnungsteil
093   '-----------------------------------------------------------------
094
095
```

```
096
```

In den Zeilen 97 bis 152 wird das Diagramm „Leistung über Versuchszeit" erstellt. In Zeile 100 werden eventuelle Bilddaten vorheriger Berechnungen im Systemspeicher gelöscht und in Zeile 101 eine Vorlage für das Layout geladen. Die Vorlage enthält Objekte, wie z.B. Textblöcke, Diagrammflächen und Grafikobjekte. Die Zeilen 103 bis 109 löschen den Inhalt dieser Objekte. Zeile 111 aktualisiert das Layout und in Zeile 112 wird eine Pause eingestellt, um das leere Layout 3 Sekunden lang anzuzeigen.

```
097   '-------------------------------------------------------------------
098   'Begin Grafikausgabe
099   '-------------------------------------------------------------------
100   Call PicDelete
101   Call Picload("c:\vorlage.LPD")
102
103   L1 = 1
104   Do While L1 <= 20
105     If PicObj(L1) <> "" Then
106       Call GraphObjChnClear(PicObj(L1))
107     End If
108     L1 = L1 + 1
109   Loop
110
111   Call PicUpdate
112   Call Pause(3)
113
```

In Zeile 114 wird das in der Vorlage gespeicherte Objekt „Diagramm" geöffnet und in den folgenden Zeilen verschiedene Eigenschaften wie z.B. X-, Y-Achsenbezeichnungen, Farbe der Kurve und die Quelldaten für X- und Y-Achse mit den Kanälen „Zeit" und „L_NOKO" festgelegt. In Zeile 129 wird dieses Objekt „Diagramm" wieder geschlossen.

```
114   Call GraphObjOpen(PicObj(1))
```

```
115
116     Call GraphObjOpen(D2AxisXObj(1))
117       D2AxisXTxt = "Versuchszeit t [s]"
119     Call GraphObjClose(D2AxisXObj(1))
120     Call GraphObjOpen(D2AxisYObj(1))
121        D2AxisYTxt = "Kondensatorleistung L [kW]"
122     Call GraphObjClose(D2AxisYObj(1))
123     D2ChnX(1) = "Zeit"
124     D2ChnY(1) = "L_NOKO"
125     Call GraphObjOpen(D2CurveObj(1))
126     D2CurveColor = "blue"
127     Call GraphObjClose(D2CurveObj(1))
128
129   Call GraphObjClose(PicObj(1))
130
```

In den Zeilen 131 bis 138 und 140 bis 147 werden zwei Textobjekte definiert (prozentuale Lage auf dem Layout, Textinhalt, Farbe und Ausrichtung des Textes), anschließend wird in Zeile 149 das Layout aktualisiert. Mittels des Befehls in Zeile 150 wird das Diagramm 10 Sekunden lang anzeigt.

```
131   Call GraphObjOpen(TxtObj(1))
132     TxtTxt = "A3"
133     TxtSize = 4
134     TxtPosX = 95
135     TxtPosY = 12
136     TxtRelPos = "left"
137     TxtColor = "black"
138   Call GraphObjClose(TxtObj(1))
139
140   Call GraphObjOpen(TxtObj(2))
141        TxtTxt = "Kondensatorleistung über der Versuchszeit"
142        TxtSize = 4
143        TxtPosX = 50
144        TxtPosY = 93
145        TxtRelPos = "cent."
146        TxtColor = "black"
147   Call GraphObjClose(TxtObj(2))
```

```
148
149   Call PicUpdate
150   Call Pause(10)
151
```

Mit dem Befehl in Zeile 155 wird das Layout in den Zwischenspeicher von Windows kopiert und steht somit anderen Anwendungen zur Verfügung. Optional kann statt „WinClip" auch „WinPrint" eingesetzt werden und die Ausgabe erfolgt dann auf den im Windows gewählten Standarddrucker. In Zeile 156 wird das Fenster von DIA*dem*-AUTO aufgerufen. Die berechneten Daten werden durch den Befehl in Zeile 157 automatisch in der Datei „Ausgabe_LKOND.DAT" abgelegt.

```
152   '-------------------------------------------------------------------
153   'Beginn Druckerausgabe
154   '-------------------------------------------------------------------
155   Call PicPrint("WinClip")
156   Call WndShow("AUTO", "OPEN")
157   Call DataSave("Ausgabe_LKOND.dat")
158   '-------------------------------------------------------------------
159   'Ende der Autosequenz
160   '-------------------------------------------------------------------
```

Mit Zeile 160 endet die Autosequenz. Das Ergebnisdiagramm der Autosequenz wird im folgenden Kapitel vorgestellt.

## 4.3 *Auswertung des NOKO Experiments A3 mit einer Autosequenz und Validierung der Ergebnisse*

Nachdem die Autosequenz aus DIA*dem*-AUTO gestartet wurde, berechnete diese die gesuchten Größen (Massenstrom und Kondensatorleitung) innerhalb einer Sekunde. Als Ergebnis erhält der Anwender untenstehendes Diagramm (vgl. Bild 4.4). Es zeigt die Kondensatorleistung in Abhängigkeit von der Versuchszeit. Die Autosequenz schreibt dafür in die Vorlagedatei des Diagramms die dazugehörigen Achsenbezeichnungen, den Diagrammtitel und die Versuchsnummer.

Die Kurve in Bild 4.4 zeigt über eine Messdauer von 500 s eine nahezu konstante Kondensatorleistung mit Werten zwischen ca. 629,55 kW und 633,3 kW. Der Unterschied zwischen dem Minimum und Maximum liegt bei Werten von 0,5% v.M. und belegt die Stabilität der Versuchsanlage während des Experiments.

Um eine qualitative Aussage über die berechneten Größen mittels der für DIA*dem* erstellten Autosequenzen in Verbindung mit der angekoppelten Wasser/Dampf-Stoffwertebibliothek machen zu können, wurde die gesamte Berechnung der Leistung für das Kondensatorbündel in Excel mit dem dafür vorliegenden Add-In FluidEXL der Hochschule Zittau/Görlitz durchgeführt. Der Vergleich des Excel-Diagramms in Bild 4.5 mit dem DIA*dem*-Diagramm zeigt eine perfekte Übereinstimmung. Diese wird bestätigt durch den stichprobenartigen Vergleich der Werte für die Kondensatorleistung zu ausgewählten Zeitpunkten.

Abschließen kann festgestellt werden, dass die Autosequenz ohne Fehlermeldungen zügig abgearbeitet wird. Zukünftig sollten bei der Erstellung von Auswerteautosequenzen Sicherheitsvorkehrungen gegen Programmabbrüche getroffen werden. Hierunter ist zu verstehen, dass z.B. bei der Auswertung der Gleichung für den Massenstrom gemäß Gl. 1 in Kap. 4.1 bei negativen Differenzdrücken die Abarbeitung der Autosequenz nicht unterbrochen wird. Eine ähnliche Problematik ist bei Brüchen mit gegen Null gehenden Nennern oder bei den Wasser/Dampf-Stoffwertefunktionen beim Wechsel von Zustandsgebieten zu beachten.

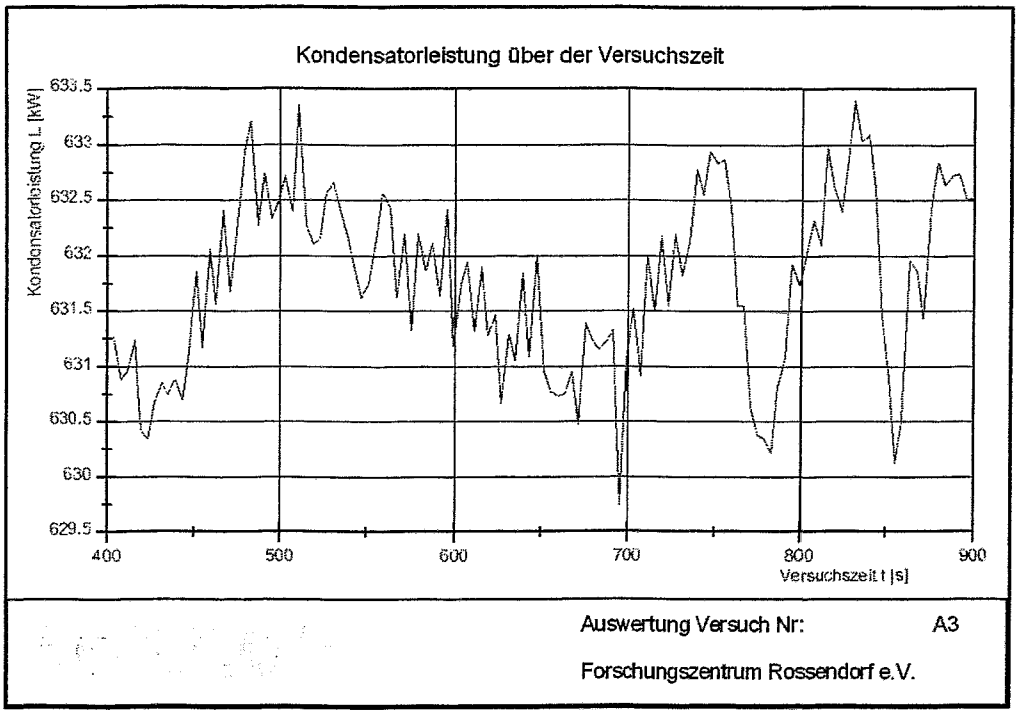



**Bild 4.4:** NOKO-Leistung in Abhängigkeit von der Versuchszeit berechnet und dargestellt mittels eine DIA*dem* Autosequenz.

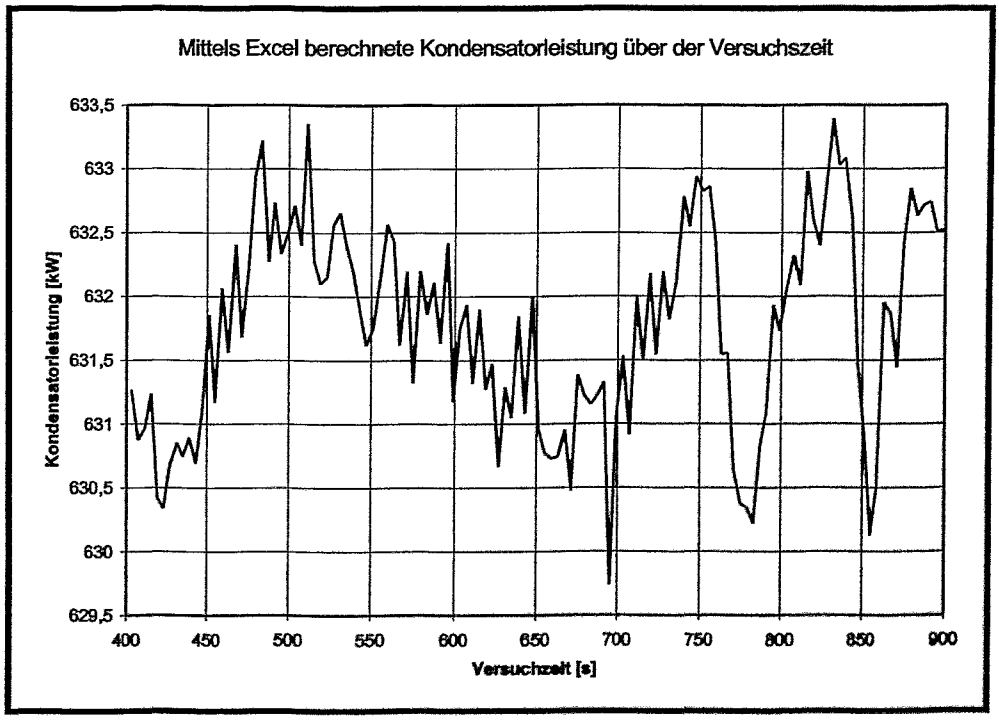



**Bild 4.5:** NOKO-Leistung in Abhängigkeit von der Versuchszeit berechnet und dargestellt mittels EXCEL mit Add-In FluidEXL.

# 5 Zusammenfassung und Ausblick

Am Institut für Sicherheitsforschung (IfS) des Forschungszentrums Rossendorf (FZR) e.V. wird derzeit die Mehrzweck-Thermohydraulikversuchsanlage **TOPFLOW** (Transient Two Phase Flow Test Facility) aufgebaut und in Betrieb genommen. Die kalte Inbetriebnahme einschließlich der Sicherheitsüberprüfungen durch den zuständigen technischen Überwachungsverein (TÜV Süddeutschland) ist abgeschlossen und es wird im August 2002 noch mit der heißen Erprobung der Versuchsanlage begonnen. Diese Erprobung umfasst - gemäß der Definition nach den Technischen Regel für Dampfkessel TRD 601 / Blatt 3 - Einstellarbeiten an der im Betrieb befindlichen Anlage sowie deren Teile, der Feststellung und Überprüfung von sicherheitstechnisch relevanten Betriebsdaten sowie Funktionsprüfungen.

Zur effizienten Auswertung dieser Tests sowie den anschließenden Abnahmen gegenüber den Lieferanten wird mit der Erstellung automatisierter Auswerteroutinen für die in TOPFLOW eingesetzte Messdatenerfassungs- und Automatisierungssoftware DIA*dem* von National Instruments (NI) begonnen. Der erste große Schritt hierzu ist die Entwicklung und Erprobung einer Schnittstelle zwischen DIA*dem* und der Wasser/Dampf-Stoffdatenbibliothek LibIF97 der Hochschule Zittau / Görlitz (FH).

In der vorliegenden Arbeit wurde die hierzu notwendige General Programming Interface (GPI) Schnittstelle programmiert und erfolgreich mit DIA*dem* gekoppelt. Anschließend wurden darüber hinaus die Leistungsfähigkeit dieser Kopplung zur automatisierten Datenauswertung mittels Autosequenzen untersucht und effiziente Methoden für die zukünftige Auswertung der Messdaten von TOPFLOW aufgezeigt.

Diese Ideen wurden anschließend an einem Beispiel umgesetzt. Da zur Zeit jedoch noch keine TOPFLOW-Versuchsdaten vorliegen, wurde der Nachweis einer erfolgreichen Kopplung anhand der Auswertung eines ausgewählten NOKO-Experiments erbracht. Hierdurch konnte ferner das Zusammenspiel der Stoffwertebibliothek mit Autosequenzen getestet und erste praktische Anwendererfahrungen gesammelt werden.

Das Beispiel der Auswertung eines NOKO-Experiments ist bewusst einfach und überschaubar gewählt. Es beinhaltet mit dem Öffnen eines Files, der Bestimmung von Stoffwerten mit Hilfe der Dynamic-Link-Library LibL97.dll, der Verknüpfung von verschiedenen Kanälen und der Erstellung und Ausgabe von Präsentationsgrafiken bereits jetzt alle für die Auswertung von TOPFLOW Experimenten benötigten Elemente.

Die in dieser Arbeit entwickelten Tools sollen nun in einem weiteren Schritt für die Auswertung der TOPFLOW Experimente adaptiert und die zur Begutachtung und Bewertung der Abnahmeversuche notwendige Autosequenzen erstellt werden. Ferner soll jetzt schon mit der Erstellung der Auswerteroutinen und -autosequenzen für die ersten Versuchsreihen begonnen werden.

# 6 Literatur

[HIE-02] E. F. Hicken, M. Fethke, H. Jaegers, A. Schaffrath. *„Der NOKO-Versuchsstand der Forschungszentrum Jülich GmbH – Rückblick auf sieben Jahre experimentelle Untersuchungen zur Erhöhung der Sicherheit von Leichtwasserreaktoren“*, atw – Internationale Zeitschrift für Kernenergie, Heft 5 - Mai, S. 343-348 (2002).

[KRH-01] H.-J. Kretzschmar, I. Stöcker. *„FluidEXL - Handbuch Stoffdaten-Unterbibliotheken für Arbeitsfluide der Energietechnik“*, Hochschule Zittau / Görlitz, Fachgebiet Technische Thermodynamik (2001).

[NID-00] *„DIAdem – Die PC-Werkstatt“*, Handbuch zu DIA*dem* 7.0, National Instruments Engineering GmbH & Co. KG., Aachen (2000).

[NID-01] im Lieferumfang des Softwarepakets DIA*dem* 8.0 enthaltene Onlinehilfe (2002).

[NID-02] National Instruments, Produktinformationen und Wissensdatenbank zu DIA*dem*, www.gfs-ac.de (2002).

[SCA-961] A. Schaffrath, H. Jaegers. *„Allgemeine Beschreibung des NOKO-Versuchsstandes“*, Berichte des Forschungszentrum Jülich: Jül-3167, Institut für Sicherheitsforschung und Reaktortechnik (1996).

[SCA-962] A. Schaffrath. *„Experimentelle und analytische Untersuchungen zur Wirksamkeit des Notkondensators des SWR600/1000“*, Dissertation, Fakultät für Maschinenbau der Ruhr-Universität Bochum (1996).

[SCA963] A.Schaffrath, H. Jaegers, P. David. *„NOKO Quick look Reports zu den Versuchen A1-A21, B1-B9, C1-C7, D1-D3 und S1-S3“*, Forschungszentrum Jülich, 43 Bände, Mai-Dezember 1996.

[SCA-02] A. Schaffrath, A.-K. Krüssenberg, F.-P. Weiß, H.-M. Prasser. *"TOPFLOW – Die Mehrzweck-Thermohydraulik-Versuchsanlage des Forschungszentrums Rossendorf e.V. – Aufbau, Ziele, Perspektiven"*, atw - Internationale Zeitschrift für Kernenergie, Heft 6 - Juni, S. 383-388 (2002).

[WAW-98] W. Wagner, A. Kruse. *„Zustandsgrößen von Wasser und Wasserdampf"*, Springer-Verlag, Berlin (1998).

# Anhang A: Quelltext der Datei commands.pas

```
Library Commands;

uses
  DataType,
  DataProc,
  WinTypes,
  WinProcs,
  WEMIProc,
  strings,
  GPIProc,
  GPIType,
  Berechnung in 'Berechnung.pas';

const
  CDLLNAME = 'COMMANDS.DLL';


type
  TVarInfo = record
    Name         : TGPIName;          (* interface name
                  *)
    cVarType     : TGPIVarType;       (* variable type
                  *)
    aDim         : TGPIVarDim;        (* (0,0) = scalar; (n,0) = vector;
(n,m) = matrix     *)
    szVarDLLName: TGPIDLLName;        (* Name of the DLL that contains the
access function; *)
                                      (* ZERO = Identical with the
registered DLL           *)
    szVarFctName: TGPIProcName;       (* Export name of the access function
                  *)
  end;

const
  aVarInfo : array[0..4] of TVarInfo = ((Name         : 'lVar';
                                         cVarType     : eGPIVARTYPELONG;
                                         aDim         : ( 0, 0);
                                         szVarDLLName : CDLLNAME;
                                         szVarFctName : 'fVariableDo'),
                                        (Name         : 'dVar';
                                         cVarType     : eGPIVARTYPEDOUBLE;
                                         aDim         : ( 0, 0);
                                         szVarDLLName : CDLLNAME;
                                         szVarFctName : 'fVariableDo'),
                                        (Name         : 'szVar';
                                         cVarType     : eGPIVARTYPESTRING;
                                         aDim         : ( 0, 0);
                                         szVarDLLName : CDLLNAME;
                                         szVarFctName : 'fVariableDo'),
                                        (Name         : 'cVar';
                                         cVarType     : eGPIVARTYPEENUM;
                                         aDim         : ( 0, 0);
                                         szVarDLLName : CDLLNAME;
                                         szVarFctName : 'fVariableDo'),
                                        (Name         : 'cDynVar';
                                         cVarType     : eGPIVarTypeDynEnum;
                                         aDim         : ( 0, 0);
                                         szVarDLLName : CDLLNAME;
                                         szVarFctName : 'fVariableDo'));

type
  TVarUnion = record
    case i: byte of
      1 : (b : Byte);
      2 : (l : LongInt);
      3 : (d : Double);
      4 : (sz: array[0..79] of char);
    end;
```

```
TVarData = record
  Value: TVarUnion;
  Min  : TVarUnion;
  Max  : TVarUnion;
end;

var
  aVarData: array[0..pred((sizeof(aVarInfo) div sizeof(TVarInfo)))] of
TVarData;


type
  TCommandInfo = record
    Name          : TGPIName;          (* interface name
             *)
    szProcDLLName: TGPIDLLName;        (* Name of the DLL that contains the
access function; *)
                                       (* ZERO = Identical with the assigned
DLL             *)
    szProcFctName: TGPIProcName;       (* Export name of the physical function
             *)
    cAutoseq      : TGPIAuto;          (* 0=Do not register; 1=Register
             *)
    nParam        : WORD;              (* Number of procedure parameters
             *)
    aParam        : TaGPIParamRec;     (* Box with procedure parameters
             *)
  end;

const
  aCommandInfo: array[0..35] of TCommandInfo =((Name          : 'FCT1';
                                                szProcDLLName: CDLLNAME;
                                                szProcFctName: 'Function1';
                                                cAutoseq     : 0;
                                                nParam       : 0),
                                               (Name          : 'FCT2';
                                                szProcDLLName: CDLLNAME;
                                                szProcFctName: 'Function2';
                                                cAutoseq     : 0;
                                                nParam       : 3),
                                               (Name          :
 'DYNREGISTER';                                 szProcDLLName: CDLLNAME;
                                                szProcFctName:
 'DynRegister';                                 cAutoseq     : 0;
                                                nParam       : 0),
                                               (Name          :
 'DYNUNREGISTER';                               szProcDLLName: CDLLNAME;
                                                szProcFctName:
 'DynUnRegister';                               cAutoseq     : 0;
                                                nParam       : 0),
                                               (Name          :
 'WD_h_ptx_97';                                 szProcDLLName: CDLLNAME;
                                                szProcFctName:
 'Fct_h_ptx_97';                                cAutoseq     : 0;
                                                nParam       : 0),
                                               (Name          :
 'WD_a_ptx_97';                                 szProcDLLName: CDLLNAME;
                                                szProcFctName:
 'Fct_a_ptx_97';
```

```
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_cp_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_cp_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_cv_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_cv_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_dvdp_T_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_dvdp_T_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_dvdT_p_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_dvdT_p_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_e_ptx_tu_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_e_ptx_tu_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_eta_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_eta_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_h_ps_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_h_ps_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_kappa_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_kappa_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_lambda_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_lambda_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_nue_ptx_97';
```

```
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_nue_ptx_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_p_hs_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_p_hs_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_Pr_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_Pr_ptx_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_ps_t_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_ps_t_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_rho_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_rho_ptx_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_s_ph_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_s_ph_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_s_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_s_ptx_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_sigma_p_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_sigma_p_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_sigma_t_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_sigma_t_97';
                                                    cAutoseq     : 0;
                                                    nParam       : 0),
                                                   (Name         :
'WD_t_hs_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_t_hs_97';
                                                    cAutoseq     : 0;
```

```
                                                    nParam        : 0),
                                                   (Name          :
'WD_t_ph_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_t_ph_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_t_ps_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_t_ps_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_ts_p_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_ts_p_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_u_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_u_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_v_ph_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_v_ph_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_v_ps_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_v_ps_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_v_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_v_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_w_ptx_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_w_ptx_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_x_hs_97';
                                                    szProcDLLName: CDLLNAME;
                                                    szProcFctName:
'Fct_x_hs_97';
                                                    cAutoseq      : 0;
                                                    nParam        : 0),
                                                   (Name          :
'WD_x_ph_97';
                                                    szProcDLLName: CDLLNAME;
```

```
                                                szProcFctName:
'Fct_x_ph_97';
                                                cAutoseq      : 0;
                                                nParam        : 0),
                                               (Name          :
'WD_x_ps_97';
                                                szProcDLLName: CDLLNAME;
                                                szProcFctName:
'Fct_x_ps_97';
                                                cAutoseq      : 0;
                                                nParam        : 0)
                                                );


function  GPICallBack(hGPIInstV: TGPIHandle;
                      uCommandV: TGPICommand;
                      InParaV  : TGPIInPara;
                      OutParaV : TGPIOutPara): TGPIReturn; stdcall; export;
var
  pDLLT           : TpGPIDLLInfo;
  pINFT           : TpGPIInfo;
  pGPIVarPropsT   : TpGPIVarProps;
  pGPIProcPropsT  : TpGPIProcProps;
  nVariablesIdxT  : TGPIInPara;
  nCommandsIdxT   : TGPIInPara;
  nT              : Integer;
  uParameterCountT: Word;

begin

  GPICallBack := 1;

  case (uCommandV) of
    eGPIINIT:
    begin
      for nVariablesIdxT:= 0  to pred(sizeof(aVarInfo) div
sizeof(TVarInfo)) do
      begin
        case aVarInfo[nVariablesIdxT].cVarType of
          eGPIVARTYPELONG:
          begin
            aVarData[nVariablesIdxT].Value.l := 7;
            aVarData[nVariablesIdxT].Min.l   := -1;
            aVarData[nVariablesIdxT].Max.l   := 10;
          end;
          eGPIVARTYPEDOUBLE:
          begin
            aVarData[nVariablesIdxT].Value.d := 13.13;
            aVarData[nVariablesIdxT].Min.d   := -1313.13;
            aVarData[nVariablesIdxT].Max.d   := 1313.13;
          end;
          eGPIVARTYPESTRING:
          begin
            strcopy(aVarData[nVariablesIdxT].Value.sz, 'This text comes
from the DLL COMMANDS!');
            aVarData[nVariablesIdxT].Min.l   := sizeof(TVarUnion);
            aVarData[nVariablesIdxT].Max.l   := sizeof(TVarUnion);
          end;
          eGPIVARTYPEENUM:
          begin
            aVarData[nVariablesIdxT].Value.b := 0;
            strcopy(aVarData[nVariablesIdxT].Min.sz, 'aaa~bbb~ccc');

            nT := 0;
            while (aVarData[nVariablesIdxT].Min.sz[nT] <> char(0)) do
            begin
```

```
                if ('~' = aVarData[nVariablesIdxT].Min.sz[nT]) then
                begin
                  aVarData[nVariablesIdxT].Min.sz[nT] := char(0);
                end;
                inc(nT);
              end;

            end;
            eGPIVARTYPEDYNENUM:
            begin
              strcopy(aVarData[nVariablesIdxT].Value.sz, 'monday');
            end;
          end;
        end;

        strcopy(@aCommandInfo[1].aParam[1].szParamName[0], 'lVar');
        aCommandInfo[1].aParam[1].cParamType := eGPIVARTYPELONG;

        strcopy(@aCommandInfo[1].aParam[2].szParamName[0], 'dVar');
        aCommandInfo[1].aParam[2].cParamType := eGPIVARTYPEDOUBLE;

        strcopy(@aCommandInfo[1].aParam[3].szParamName[0], 'szVar');
        aCommandInfo[1].aParam[3].cParamType := eGPIVARTYPESTRING;

      end;
      eGPIEXIT:
      begin
      end;
      eGPIGETDLLINFO:
      begin
        pDLLT := TpGPIDLLInfo(OutParaV);
        strcopy(@pDLLT^.szName,      CDLLNAME);
        strcopy(@pDLLT^.szComment,   '');
        strcopy(@pDLLT^.szCopyright, '');

        pDLLT^.nVersion    := GPIVERSION;
        pDLLT^.nExtensions := sizeof(aVarInfo) div sizeof(TVarInfo) +
                              sizeof(aCommandInfo) div sizeof(TCommandInfo);

        strcopy(@pDLLT^.CompanyInfo.szCompany, '');
        strcopy(@pDLLT^.CompanyInfo.szAddress, '');
        strcopy(@pDLLT^.CompanyInfo.szPhone,   '');
        strcopy(@pDLLT^.CompanyInfo.szVersion, '');
      end;
      eGPIGETINFO:
      begin
        pINFT := TpGPIInfo(OutParaV);

        if (InParaV <= sizeof(aVarInfo) div sizeof(TVarInfo)) then
        begin
          nVariablesIdxT := InParaV - 1;
          strcopy(@pINFT^.szName[0], @aVarInfo[nVariablesIdxT].Name[0]);
          pINFT^.cType   := eGPIVARIABLE;
          pINFT^.hHandle := MAKELONG(nVariablesIdxT, eGPIVARIABLE);
        end else
        begin
          if (InParaV <= (  sizeof(aVarInfo) div sizeof(TVarInfo)
                          + sizeof(aCommandInfo) div sizeof(TCommandInfo)))
then
          begin
            nCommandsIdxT :=  InParaV
                            - sizeof(aVarInfo) div sizeof(TVarInfo)
                            -1;
            strcopy(@pINFT^.szName[0], @aCommandInfo[nCommandsIdxT].Name[0]);
            pINFT^.cType   := eGPICOMMAND;
            pINFT^.hHandle := MAKELONG(nCommandsIdxT, eGPICOMMAND);
          end;
```

```
      end;
    end;
    eGPIGETPROPS:
    begin
      if (HIWORD(InParaV) = eGPIVARIABLE) then
      begin
        pGPIVarPropsT := TpGPIVarProps(OutParaV);
        nVariablesIdxT:= LOWORD(InParaV);

        pGPIVarPropsT^.cVarType := aVarInfo[nVariablesIdxT].cVarType;

        pGPIVarPropsT^.aDim[1]  := aVarInfo[nVariablesIdxT].aDim[1];
        pGPIVarPropsT^.aDim[2]  := aVarInfo[nVariablesIdxT].aDim[2];
        strcopy(@pGPIVarPropsT^.szVarDLLName[0],
@aVarInfo[nVariablesIdxT].szVarDLLName[0]);
        strcopy(@pGPIVarPropsT^.szVarFctName[0],
@aVarInfo[nVariablesIdxT].szVarFctName[0]);
      end else
      begin
        if (HIWORD(InParaV) = eGPICOMMAND) then
        begin
          pGPIProcPropsT := TpGPIProcProps(OutParaV);
          nCommandsIdxT  := LOWORD(InParaV);

          strcopy(@pGPIProcPropsT^.szProcDLLName[0],
@aCommandInfo[nCommandsIdxT].szProcDLLName[0]);
          strcopy(@pGPIProcPropsT^.szProcFctName[0],
@aCommandInfo[nCommandsIdxT].szProcFctName[0]);
          pGPIProcPropsT^.cAutoseq := aCommandInfo[nCommandsIdxT].cAutoseq;
          pGPIProcPropsT^.nParam   := aCommandInfo[nCommandsIdxT].nParam;

          uParameterCountT := 1;
          while (uParameterCountT <= aCommandInfo[nCommandsIdxT].nParam) do
          begin

strcopy(@pGPIProcPropsT^.aParam[uParameterCountT].szParamName[0],

@aCommandInfo[nCommandsIdxT].aParam[uParameterCountT].szParamName[0]);

            pGPIProcPropsT^.aParam[uParameterCountT].cParamType :=

aCommandInfo[nCommandsIdxT].aParam[uParameterCountT].cParamType;

            inc(uParameterCountT);
          end;
        end;
      end;
    end;

  end;

end;


function fVariableDo(      hGPIInstP  : TGPIHandle;
                       const hVarNumberV: TGPIHandle;
                       const uActionV   : TGPIAction;
                       const lRowV      : TGPIIndex;
                       const lColumnV   : TGPIIndex;
                       const ValueP     : Pointer) : LongInt; stdcall;
export;
type
  Tpb     = ^Byte;
  Tpl     = ^LongInt;
  Tpd     = ^Double;
  TppChar = ^pChar;
var
```

```
  nVariablesIdxT : Word;
  lReturnValueT  : LongInt;
  pszT           : pChar;
  pszScanT       : pChar;
begin
  nVariablesIdxT  := LOWORD(hVarNumberV);
  lReturnValueT   := 1;

  case (uActionV) of
    eGPIVARGET:
    begin
      case (aVarInfo[nVariablesIdxT].cVarType) of
        eGPIVARTYPELONG:
        begin
          Tpl(ValueP)^ := aVarData[nVariablesIdxT].Value.l;
        end;
        eGPIVARTYPEDOUBLE:
        begin
          Tpd(ValueP)^ := aVarData[nVariablesIdxT].Value.d;
        end;
        eGPIVARTYPESTRING,
        eGPIVARTYPEDYNENUM:
        begin
          strcopy(pChar(ValueP), aVarData[nVariablesIdxT].Value.sz);
        end;
        eGPIVARTYPEENUM:
        begin
          Tpb(ValueP)^ := aVarData[nVariablesIdxT].Value.b;
        end;
      end;
    end;
    eGPIVARSET:
    begin
      case (aVarInfo[nVariablesIdxT].cVarType) of
        eGPIVARTYPELONG:
        begin
          aVarData[nVariablesIdxT].Value.l := Tpl(ValueP)^;
        end;
        eGPIVARTYPEDOUBLE:
        begin
          aVarData[nVariablesIdxT].Value.d := Tpd(ValueP)^;
        end;
        eGPIVARTYPESTRING,
        eGPIVARTYPEDYNENUM:
        begin
          strcopy(aVarData[nVariablesIdxT].Value.sz, pChar(ValueP));
        end;
        eGPIVARTYPEENUM:
        begin
          aVarData[nVariablesIdxT].Value.b := Tpb(ValueP)^;
        end;
      end;
    end;
    eGPIVARGETMIN:
    begin
      case (aVarInfo[nVariablesIdxT].cVarType) of
        eGPIVARTYPELONG:
        begin
          Tpl(ValueP)^ := aVarData[nVariablesIdxT].Min.l;
        end;
        eGPIVARTYPEDOUBLE:
        begin
          Tpd(ValueP)^ := aVarData[nVariablesIdxT].Min.d;
        end;
      end;
    end;
    eGPIVARGETMAX:
```

```
begin
  case (aVarInfo[nVariablesIdxT].cVarType) of
    eGPIVARTYPELONG:
    begin
      Tpl(ValueP)^ := aVarData[nVariablesIdxT].Max.l;
    end;
    eGPIVARTYPEDOUBLE:
    begin
      Tpd(ValueP)^ := aVarData[nVariablesIdxT].Max.d;
    end;
  end;
end;
eGPIVARSTRLEN:
begin
  case (aVarInfo[nVariablesIdxT].cVarType) of
    eGPIVARTYPESTRING,
    eGPIVARTYPEDYNENUM:
    begin
      Tpb(ValueP)^ := sizeof(TVarUnion);
    end;
  end;
end;
eGPIVARCREATELIST:
begin
   case (aVarInfo[nVariablesIdxT].cVarType) of
    eGPIVARTYPEENUM:
    begin
      TppChar(ValueP)^ := aVarData[nVariablesIdxT].Min.sz;
    end;
    eGPIVARTYPEDYNENUM:
    begin

      if (0 = lRowV) then
      begin
        pszT := strnew('monday~tuesday~wednesday#');

        pszScanT  := pszT;
        while (#0 <> pszScanT^) do
        begin
          if ('~' = pszScanT^) then
          begin
            pszScanT^ := #0;
          end else
          begin
            if ('#' = pszScanT^) then
            begin
              pszScanT^ := #0;
            end;
          end;
          inc(pszScanT);
        end;

      end else
      begin
        pszT := nil;
      end;

      TppChar(ValueP)^ := pszT;

    end;
  end;
end;
eGPIVARRELEASELIST:
begin
  case (aVarInfo[nVariablesIdxT].cVarType) of
    eGPIVARTYPEDYNENUM:
    begin
```

```
          if (nil <> TppChar(ValueP)^) then
          begin
            strdispose(TppChar(ValueP)^);
          end;
        end;
      end;
    end else
    begin
      lReturnValueT := 0;
    end;
  end;
  fVariableDo := lReturnValueT;
end;

procedure Function1; stdcall; export;
begin
  MessageBox(hDIAdemWndGet,
             'This function was just called! It has no parameters.',
             'Function1',
             MB_ICONINFORMATION or
             MB_OK);
end;

procedure Function2(const lParamV   : LongInt;
                    const dParamV   : Double;
                    const lpszParamV: pChar); stdcall; export;
var
  szT1: array[0..255] of Char;
  szT2: array[0..159] of Char;
  sgT : string[30];
  szT : array[0..29] of Char;

  Buffer  : record
    l         : LongInt;
    pszDouble: pChar;
    psz       : pChar;
  end;

begin
  str(dParamV:4:4, sgT);
  strpcopy(szT, sgT);

  with Buffer do
  begin
    l          := lParamV;
    pszDouble  := szT;
    psz        := lpszParamV;
  end;

  strcopy(szT1, 'This function was just called! It has three parameters.');

  wvsprintf(szT2,
            'lParamV = %ld, dParamV = %s, lpszParamV = %s',
            @Buffer);

  strcat(szT1, szT2);

  MessageBox(hDIAdemWndGet,
             szT2,
             'Function2',
             MB_ICONINFORMATION or
             MB_OK);
end;


var
  lDynVar : LongInt;
```

```
const
  CLDYNVAR            : pChar = 'lDynVar';
  CLDYNVARINDEX                = 0;
  CDYNFUNCTIONCOMMAND: pChar = 'DYNFUNCTION';
  CDYNFUNCTION        : pChar = 'DynFunction';
  CACCESSFUNCTION     : pChar = 'fDynVariableDo';


function fDynVariableDo(      hGPIInstP  : TGPIHandle;
                        const hVarNumberV: TGPIHandle;
                        const uActionV   : TGPIAction;
                        const lRowV      : TGPIIndex;
                        const lColumnV   : TGPIIndex;
                        const ValueP     : Pointer) : LongInt; stdcall;
export;
type
  Tpl = ^LongInt;
var
  nVariablesIdxT : Word;
  lReturnValueT  : LongInt;
begin
  nVariablesIdxT  := LOWORD(hVarNumberV);
  lReturnValueT   := 1;

  if (CLDYNVARINDEX = nVariablesIdxT) then
  begin
    case (uActionV) of
      eGPIVARGET:
      begin
        Tpl(ValueP)^ :=lDynVar;
      end;
      eGPIVARSET:
      begin
        lDynVar      := Tpl(ValueP)^;
      end;
      eGPIVARGETMIN:
      begin
        Tpl(ValueP)^ := -1313;
      end;
      eGPIVARGETMAX:
      begin
        Tpl(ValueP)^ := +1313;
      end;
      else
      begin
        lReturnValueT:= 0;
      end;
    end;
  end;

  fDynVariableDo := lReturnValueT;
end;

procedure DynFunction; stdcall; export;
begin
  MessageBox(hDIAdemWndGet,
             'This function was just called! It has no parameters.',
             'DynFunction',
             MB_ICONINFORMATION or
             MB_OK);
end;

procedure DynRegister; stdcall; export;
var
  GPIVarPropsT : TGPIVarProps;
  GPIHandleT   : TGPIHandle;
```

```
  AccessInfoT  : TGPIHandle;
  GPIProcPropsT: TGPIProcProps;
begin

  (*
     The registry of the dynamic variable lDynVar.
  *)
  GPIVarPropsT.cVarType := eGPIVARTYPELONG;
  GPIVarPropsT.aDim[1]  := 0;
  GPIVarPropsT.aDim[2]  := 0;
  strcopy(GPIVarPropsT.szVarDLLName,CDLLNAME);
  strcopy(GPIVarPropsT.szVarFctName, CACCESSFUNCTION);

  GPIHandleT  := hGPIGet(hInstance);

  AccessInfoT := MAKELONG(CLDYNVARINDEX, eGPIVARIABLE);

  fGPIVarRegister(GPIHandleT,
                  CLDYNVAR,
                  AccessInfoT,
                  @GPIVarPropsT);

  FillChar(GPIProcPropsT, sizeof(GPIProcPropsT), 0);

  strcopy(GPIProcPropsT.szProcDLLName, CDLLNAME);
  strcopy(GPIProcPropsT.szProcFctName, CDYNFUNCTION);

  MAKELONG(0, eGPICOMMAND);

  fGPICommandRegister(GPIHandleT,
                      CDYNFUNCTIONCOMMAND,
                      0,
                      @GPIProcPropsT);

end;

procedure DynUnRegister; stdcall; export;
begin
  fGPIVarUnregister(CLDYNVAR);

  fGPICommandUnregister(CDYNFUNCTIONCOMMAND);
end;

procedure Fct_h_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  hptx_Lesen(p,t,x);
  hptx_Schreiben(h_ptx_97(p,t,x));
end;

procedure Fct_a_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  aptx_Lesen(p,t,x);
  aptx_Schreiben(a_ptx_97(p,t,x));
end;

procedure Fct_cp_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  cpptx_Lesen(p,t,x);
  cpptx_Schreiben(cp_ptx_97(p,t,x));
end;

procedure Fct_cv_ptx_97; stdcall; export;
var p,t,x : Double;
begin
```

```
  cvptx_Lesen(p,t,x);
  cvptx_Schreiben(cv_ptx_97(p,t,x));
end;

procedure Fct_dvdp_T_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  dvdptptx_Lesen(p,t,x);
  dvdptptx_Schreiben(dv_dp_T_ptx_97(p,t,x));
end;

procedure Fct_dvdT_p_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  dvdtpptx_Lesen(p,t,x);
  dvdtpptx_Schreiben(dv_dT_p_ptx_97(p,t,x));
end;

procedure Fct_e_ptx_tu_97; stdcall; export;
var p,t,x,tu : Double;
begin
  eptxtu_Lesen(p,t,x,tu);
  eptxtu_Schreiben(e_ptx_tu_97(p,t,x,tu));
end;

procedure Fct_eta_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  etaptx_Lesen(p,t,x);
  etaptx_Schreiben(eta_ptx_97(p,t,x));
end;

procedure Fct_h_ps_97; stdcall; export;
var p,s : Double;
begin
  hps_Lesen(p,s);
  hps_Schreiben(h_ps_97(p,s));
end;

procedure Fct_kappa_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  kappaptx_Lesen(p,t,x);
  kappaptx_Schreiben(kappa_ptx_97(p,t,x));
end;

procedure Fct_lambda_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  lambdaptx_Lesen(p,t,x);
  lambdaptx_Schreiben(lambda_ptx_97(p,t,x));
end;

procedure Fct_nue_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  nueptx_Lesen(p,t,x);
  nueptx_Schreiben(nue_ptx_97(p,t,x));
end;

procedure Fct_p_hs_97; stdcall; export;
var a,b : Double;
begin
  phs_Lesen(a,b);
  phs_Schreiben(p_hs_97(a,b));
end;
```

```
procedure Fct_Pr_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  prptx_Lesen(p,t,x);
  prptx_Schreiben(Pr_ptx_97(p,t,x));
end;

procedure Fct_ps_t_97; stdcall; export;
var t : Double;
begin
  pst_Lesen(t);
  pst_Schreiben(ps_t_97(t));
end;

procedure Fct_rho_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  rhoptx_Lesen(p,t,x);
  rhoptx_Schreiben(rho_ptx_97(p,t,x));
end;

procedure Fct_s_ph_97; stdcall; export;
var p,h : Double;
begin
  sph_Lesen(p,h);
  sph_Schreiben(s_ph_97(p,h));
end;

procedure Fct_s_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  sptx_Lesen(p,t,x);
  sptx_Schreiben(s_ptx_97(p,t,x));
end;

procedure Fct_sigma_p_97; stdcall; export;
var p : Double;
begin
  sigmap_Lesen(p);
  sigmap_Schreiben(sigma_p_97(p));
end;

procedure Fct_sigma_t_97; stdcall; export;
var t : Double;
begin
  sigmat_Lesen(t);
  sigmat_Schreiben(sigma_t_97(t));
end;

procedure Fct_t_hs_97; stdcall; export;
var h,s : Double;
begin
  ths_Lesen(h,s);
  ths_Schreiben(t_hs_97(h,s));
end;

procedure Fct_t_ph_97; stdcall; export;
var p,h : Double;
begin
  tph_Lesen(p,h);
  tph_Schreiben(t_ph_97(p,h));
end;

procedure Fct_t_ps_97; stdcall; export;
var p,s : Double;
begin
  tps_Lesen(p,s);
```

```
  tps_Schreiben(t_ps_97(p,s));
end;

procedure Fct_ts_p_97; stdcall; export;
var p : Double;
begin
  tsp_Lesen(p);
  tsp_Schreiben(ts_p_97(p));
end;

procedure Fct_u_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  uptx_Lesen(p,t,x);
  uptx_Schreiben(u_ptx_97(p,t,x));
end;

procedure Fct_v_ph_97; stdcall; export;
var p,h : Double;
begin
  vph_Lesen(p,h);
  vph_Schreiben(v_ph_97(p,h));
end;

procedure Fct_v_ps_97; stdcall; export;
var p,s : Double;
begin
  vps_Lesen(p,s);
  vps_Schreiben(v_ps_97(p,s));
end;

procedure Fct_v_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  vptx_Lesen(p,t,x);
  vptx_Schreiben(v_ptx_97(p,t,x));
end;

procedure Fct_w_ptx_97; stdcall; export;
var p,t,x : Double;
begin
  wptx_Lesen(p,t,x);
  wptx_Schreiben(w_ptx_97(p,t,x));
end;

procedure Fct_x_hs_97; stdcall; export;
var h,s : Double;
begin
  xhs_Lesen(h,s);
  xhs_Schreiben(x_hs_97(h,s));
end;

procedure Fct_x_ph_97; stdcall; export;
var p,h : Double;
begin
  xph_Lesen(p,h);
  xph_Schreiben(x_ph_97(p,h));
end;

procedure Fct_x_ps_97; stdcall; export;
var p,s : Double;
begin
  xps_Lesen(p,s);
  xps_Schreiben(x_ps_97(p,s));
end;

exports
```

```
  Commands.fVariableDo                Index 1,
  Commands.GPICallBack                Index 4,
  Commands.fDynVariableDo             Index 5,

  Commands.Function1                  Index 13,
  Commands.Function2                  Index 14,
  Commands.DynFunction                Index 15,
  Commands.Fct_h_ptx_97               Index 16,
  Commands.Fct_a_ptx_97               Index 17,
  Commands.Fct_cp_ptx_97              Index 18,
  Commands.Fct_cv_ptx_97              Index 19,
  Commands.Fct_dvdp_t_ptx_97          Index 20,
  Commands.Fct_dvdt_p_ptx_97          Index 21,
  Commands.Fct_e_ptx_tu_97            Index 22,
  Commands.Fct_eta_ptx_97             Index 23,
  Commands.Fct_h_ps_97                Index 24,
  Commands.Fct_kappa_ptx_97           Index 25,
  Commands.Fct_lambda_ptx_97          Index 26,
  Commands.Fct_nue_ptx_97             Index 27,
  Commands.Fct_p_hs_97                Index 28,
  Commands.Fct_Pr_ptx_97              Index 29,
  Commands.Fct_ps_t_97                Index 30,
  Commands.Fct_rho_ptx_97             Index 31,
  Commands.Fct_s_ph_97                Index 32,
  Commands.Fct_s_ptx_97               Index 33,
  Commands.Fct_sigma_p_97             Index 34,
  Commands.Fct_sigma_t_97             Index 35,
  Commands.Fct_t_hs_97                Index 36,
  Commands.Fct_t_ph_97                Index 37,
  Commands.Fct_t_ps_97                Index 38,
  Commands.Fct_ts_p_97                Index 39,
  Commands.Fct_u_ptx_97               Index 40,
  Commands.Fct_v_ph_97                Index 41,
  Commands.Fct_v_ps_97                Index 42,
  Commands.Fct_v_ptx_97               Index 43,
  Commands.Fct_w_ptx_97               Index 44,
  Commands.Fct_x_hs_97                Index 45,
  Commands.Fct_x_ph_97                Index 46,
  Commands.Fct_x_ps_97                Index 47,

  Commands.DynRegister                Index 50,
  Commands.DynUnRegister              Index 51;

begin
end.
```

# Anhang B: Quelltext der Datei berechnung.pas

```
unit Berechnung;

interface
uses
  DataType, DataProc, GPIType, GPIProc,
  strings, Dialogs, StdCtrls, Controls;

    procedure hptx_Lesen ( var p,t,x : Double );
    procedure hptx_Schreiben ( h : Double );

    procedure aptx_Lesen ( var p,t,x : Double );
    procedure aptx_Schreiben ( a : Double );

    procedure cpptx_Lesen ( var p,t,x : Double );
    procedure cpptx_Schreiben ( cp : Double );

    procedure cvptx_Lesen ( var p,t,x : Double );
    procedure cvptx_Schreiben ( cv : Double );

    procedure dvdptptx_Lesen ( var p,t,x : Double );
    procedure dvdptptx_Schreiben ( dvdpt : Double );

    procedure dvdtpptx_Lesen ( var p,t,x : Double );
    procedure dvdtpptx_Schreiben ( dvdtp : Double );

    procedure eptxtu_Lesen ( var p,t,x,tu : Double );
    procedure eptxtu_Schreiben ( e : Double );

    procedure etaptx_Lesen ( var p,t,x : Double );
    procedure etaptx_Schreiben ( eta : Double );

    procedure hps_Lesen ( var p,s : Double );
    procedure hps_Schreiben ( h : Double );

    procedure kappaptx_Lesen ( var p,t,x : Double );
    procedure kappaptx_Schreiben ( kappa : Double );

    procedure lambdaptx_Lesen ( var p,t,x : Double );
    procedure lambdaptx_Schreiben ( lambda : Double );

    procedure nueptx_Lesen ( var p,t,x : Double );
    procedure nueptx_Schreiben ( nue : Double );

    procedure phs_Lesen ( var h,s : Double );
    procedure phs_Schreiben ( p : Double );

    procedure prptx_Lesen ( var p,t,x : Double );
    procedure prptx_Schreiben ( pr : Double );

    procedure pst_Lesen ( var t : Double );
    procedure pst_Schreiben ( ps : Double );

    procedure rhoptx_Lesen ( var p,t,x : Double );
    procedure rhoptx_Schreiben ( rho : Double );

    procedure sph_Lesen ( var p,h : Double );
    procedure sph_Schreiben ( s : Double );

    procedure sptx_Lesen ( var p,t,x : Double );
    procedure sptx_Schreiben ( s : Double );

    procedure sigmap_Lesen ( var p : Double );
    procedure sigmap_Schreiben ( sigma : Double );

    procedure sigmat_Lesen ( var t : Double );
    procedure sigmat_Schreiben ( sigma : Double );

    procedure ths_Lesen ( var h,s : Double );
    procedure ths_Schreiben ( t : Double );
```

```
    procedure tph_Lesen ( var p,h : Double );
    procedure tph_Schreiben ( t : Double );

    procedure tps_Lesen ( var p,s : Double );
    procedure tps_Schreiben ( t : Double );

    procedure tsp_Lesen ( var p : Double );
    procedure tsp_Schreiben ( ts : Double );

    procedure uptx_Lesen ( var p,t,x : Double );
    procedure uptx_Schreiben ( u : Double );

    procedure vph_Lesen ( var p,h : Double );
    procedure vph_Schreiben ( v : Double );

    procedure vps_Lesen ( var p,s : Double );
    procedure vps_Schreiben ( v : Double );

    procedure vptx_Lesen ( var p,t,x : Double );
    procedure vptx_Schreiben ( v : Double );

    procedure wptx_Lesen ( var p,t,x : Double );
    procedure wptx_Schreiben ( w : Double );

    procedure xhs_Lesen ( var h,s : Double );
    procedure xhs_Schreiben ( x : Double );

    procedure xph_Lesen ( var p,h : Double );
    procedure xph_Schreiben ( x : Double );

    procedure xps_Lesen ( var p,s : Double );
    procedure xps_Schreiben ( x : Double );


    function h_ptx_97(var p,t,x : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name '_HPTX97@12';

    function a_ptx_97(var p,t,x : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name '_APTX97@12';

    function cp_ptx_97(var p,t,x : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name '_CPPTX97@12';

    function cv_ptx_97(var p,t,x : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name '_CVPTX97@12';

    function dv_dp_T_ptx_97(var p,t,x : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name '_DVDPT97@12';

    function dv_dT_p_ptx_97(var p,t,x : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name '_DVDTP97@12';

    function e_ptx_tu_97(var p,t,x,tu : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name
'_EPTXTU97@16';

    function eta_ptx_97(var p,t,x: double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name
'_ETAPTX97@12';

    function h_ps_97(var p,s : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name '_HPS97@8';

    function kappa_ptx_97(var p,t,x : double) : Double;stdcall;
                      external 'LibIF97_bar_C.dll' name
'_KAPPTX97@12';

    function lambda_ptx_97(var p,t,x : double) : Double;stdcall;
```

```
                    external 'LibIF97_bar_C.dll' name
'_LAMPTX97@12';

    function nue_ptx_97(var p,t,x : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name
'_NUEPTX97@12';

    function p_hs_97(var h,s : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_PHS97@8';

    function Pr_ptx_97(var p,t,x : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_PRPTX97@12';

    function ps_t_97(var t : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_PST97@4';

    function rho_ptx_97(var p,t,x : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name
'_RHOPTX97@12';

    function s_ph_97(var p,h : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_SPH97@8';

    function s_ptx_97(var p,t,x : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_SPTX97@12';

    function sigma_p_97(var p : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_SIGMAP97@4';

    function sigma_t_97(var t : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_SIGMAT97@4';

    function t_hs_97(var h,s : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_THS97@8';

    function t_ph_97(var p,h : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_TPH97@8';

    function t_ps_97(var p,s : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_TPS97@8';

    function ts_p_97(var p : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_TSP97@4';

    function u_ptx_97(var p,t,x : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_UPTX97@12';

    function v_ph_97(var p,h : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_VPH97@8';

    function v_ps_97(var p,s : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_VPS97@8';

    function v_ptx_97(var p,t,x : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_VPTX97@12';

    function w_ptx_97(var p,t,x : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_WPTX97@12';

    function x_hs_97(var h,s : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_XHS97@8';

    function x_ph_97(var p,h : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_XPH97@8';

    function x_ps_97(var p,s : double) : Double;stdcall;
                    external 'LibIF97_bar_C.dll' name '_XPS97@8';
```

```
implementation
{------------------------------------------------------------------------
---------}
procedure hptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                  { Kanalheader }
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);                { Init khead }
   nDataObjHeaderGet(kanal ,khead);        { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure hptx_Schreiben ( h : Double );
var
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;             { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := h;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{------------------------------------------------------------------------
---------}
{------------------------------------------------------------------------
---------}
procedure aptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                  { Kanalheader }
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);                { Init khead }
   nDataObjHeaderGet(kanal ,khead);        { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure aptx_Schreiben ( a : Double );
var
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;             { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := a;
```

```
   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure cpptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure cpptx_Schreiben ( cp : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := cp;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure cvptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure cvptx_Schreiben ( cv : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
```

```
   kanal       := 1;              { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := cv;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure dvdptptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                 { Kanalheader }
    UDaten     : TDataChannelUserData;           { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure dvdptptx_Schreiben ( dvdpt : Double );
var
    UDaten     : TDataChannelUserData;           { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;              { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := dvdpt;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure dvdtpptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                 { Kanalheader }
    UDaten     : TDataChannelUserData;           { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure dvdtpptx_Schreiben ( dvdtp : Double );
```

```
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := dvdtp;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure eptxtu_Lesen ( var p,t,x,tu : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
   tu      := UDaten.ad[3];
end;

procedure eptxtu_Schreiben ( e : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[4] := e;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure etaptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
```

```
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure etaptx_Schreiben ( eta : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := eta;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure hps_Lesen ( var p,s : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   s       := UDaten.ad[1];
end;

procedure hps_Schreiben ( h : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := h;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure kappaptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
```

```
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p      := UDaten.ad[0];
   t      := UDaten.ad[1];
   x      := UDaten.ad[2];
end;

procedure kappaptx_Schreiben ( kappa : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := kappa;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure lambdaptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);             { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p      := UDaten.ad[0];
   t      := UDaten.ad[1];
   x      := UDaten.ad[2];
end;

procedure lambdaptx_Schreiben ( lambda : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := lambda;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure nueptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
```

```
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure nueptx_Schreiben ( nue : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;             { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := nue;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure phs_Lesen ( var h,s : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   h       := UDaten.ad[0];
   s       := UDaten.ad[1];
end;

procedure phs_Schreiben ( p : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;             { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := p;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
```

```
procedure prptx_Lesen ( var p,t,x : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p      := UDaten.ad[0];
   t      := UDaten.ad[1];
   x      := UDaten.ad[2];
end;

procedure prptx_Schreiben ( pr : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := pr;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure pst_Lesen ( var t : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   t      := UDaten.ad[0];
end;

procedure pst_Schreiben ( ps : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[1] := ps;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
```

```
---------}
{-------------------------------------------------------------------
---------}
procedure rhoptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure rhoptx_Schreiben ( rho : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := rho;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure sph_Lesen ( var p,h : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   h       := UDaten.ad[1];
end;

procedure sph_Schreiben ( s : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := s;
```

```
    lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                        SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure sptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure sptx_Schreiben ( s : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := s;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                       SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure sigmap_Lesen ( var p : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
end;

procedure sigmap_Schreiben ( sigma : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
```

```
                        SizeOf(UDaten), @UDaten);
   UDaten.ad[1] := sigma;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure sigmat_Lesen ( var t : Double );
var
    khead       : TDataObjHeader;                  { Kanalheader }
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);                { Init khead }
   nDataObjHeaderGet(kanal ,khead);        { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                        SizeOf(UDaten), @UDaten);
   t       := UDaten.ad[0];
end;

procedure sigmat_Schreiben ( sigma : Double );
var
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;             { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                        SizeOf(UDaten), @UDaten);
   UDaten.ad[1] := sigma;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                          SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure ths_Lesen ( var h,s : Double );
var
    khead       : TDataObjHeader;                  { Kanalheader }
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);                { Init khead }
   nDataObjHeaderGet(kanal ,khead);        { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                        SizeOf(UDaten), @UDaten);
   h       := UDaten.ad[0];
   s       := UDaten.ad[1];
end;

procedure ths_Schreiben ( t : Double );
var
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;             { zeitkanal }
```

```
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := t;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure tph_Lesen ( var p,h : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p      := UDaten.ad[0];
   h      := UDaten.ad[1];
end;

procedure tph_Schreiben ( t : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := t;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure tps_Lesen ( var p,s : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p      := UDaten.ad[0];
   s      := UDaten.ad[1];
end;

procedure tps_Schreiben ( t : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
```

```
    kanal       : Integer;
begin
   kanal        := 1;              { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := t;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure tsp_Lesen ( var p : Double );
var
    khead       : TDataObjHeader;                  { Kanalheader }
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);                { Init khead }
   nDataObjHeaderGet(kanal ,khead);        { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
end;

procedure tsp_Schreiben ( ts : Double );
var
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;              { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[1] := ts;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure uptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                  { Kanalheader }
    UDaten      : TDataChannelUserData;            { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);                { Init khead }
   nDataObjHeaderGet(kanal ,khead);        { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure uptx_Schreiben ( u : Double );
```

```
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := u;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure vph_Lesen ( var p,h : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   h       := UDaten.ad[1];
end;

procedure vph_Schreiben ( v : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := v;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure vps_Lesen ( var p,s : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   s       := UDaten.ad[1];
```

```
end;

procedure vps_Schreiben ( v : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := v;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure vptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   t       := UDaten.ad[1];
   x       := UDaten.ad[2];
end;

procedure vptx_Schreiben ( v : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := v;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure wptx_Lesen ( var p,t,x : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal        := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
```

```
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p      := UDaten.ad[0];
   t      := UDaten.ad[1];
   x      := UDaten.ad[2];
end;

procedure wptx_Schreiben ( w : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[3] := w;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure xhs_Lesen ( var h,s : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   h      := UDaten.ad[0];
   s      := UDaten.ad[1];
end;

procedure xhs_Schreiben ( x : Double );
var
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := x;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure xph_Lesen ( var p,h : Double );
var
    khead      : TDataObjHeader;                { Kanalheader }
    UDaten     : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal      : Integer;
begin
```

```
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   h       := UDaten.ad[1];
end;

procedure xph_Schreiben ( x : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := x;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}
{-------------------------------------------------------------------
---------}
procedure xps_Lesen ( var p,s : Double );
var
    khead       : TDataObjHeader;                { Kanalheader }
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal       := 1; { Festlegung Zeitkanal }
   khead.lsize := SizeOf(TDataObjHeader);              { Init khead }
   nDataObjHeaderGet(kanal ,khead);       { Header Kanal kanal lesen
}
   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   p       := UDaten.ad[0];
   s       := UDaten.ad[1];
end;

procedure xps_Schreiben ( x : Double );
var
    UDaten      : TDataChannelUserData;          { Userdaten pro Kanal
}
    kanal       : Integer;
begin
   kanal       := 1;            { zeitkanal }

   lDataObjProperties (kanal, ORD(eChannelUserDataGet),
                       SizeOf(UDaten), @UDaten);
   UDaten.ad[2] := x;

   lDataObjProperties (kanal, ORD(eChannelUserDataSet),
                         SizeOf(UDaten), @UDaten);
end;
{-------------------------------------------------------------------
---------}

end.
```