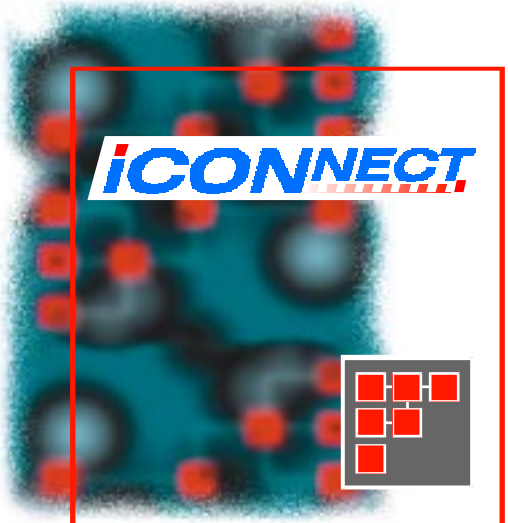
ICONNECT

Mit Abstand der bessere Weg

MICRO-EPSILON

SOFTWARE SYSTEMS



Graphisches Entwicklungssystem zum objektorientierten Design von Datenverarbeitungs-Algorithmen

Windows, Windows NT sind Warenzeichen der Microsoft Corporation.

MICRO-EPSILON MESSTECHNIK
GmbH & Co.KG
Bereich Software
Königbacher Str. 15
94496 Ortenburg

Tel. 08542/168-0
Fax 08542/168-90
e-mail: info@micro-epsilon.de
http://www.micro-epsilon.com

X9750025-C019109MSC

Einführung

ICONNECT ist ein Entwicklungs- und Laufzeitsystem für Signalverarbeitungsaufgaben der Automatisierungs-, Meß- und Prüftechnik, sowie für Prozeßüberwachung und -steuerung. In einer graphischen Oberfläche entwerfen Sie einen Signalgraphen, der den Datenfluß eines komplexen Ablaufs beschreibt. ICONNECT stellt in einer Bibliothek Algorithmen (Module) zur Verfügung, die Sie am Monitor mit der Maus „verdrahten“.

Das vorliegende Tutorial wurde geschrieben, um den Entwickler in die Lage zu versetzen, selbst Applikationen zu entwerfen. Die einzelnen Inhalte werden anhand von Beispielen erläutert, die sämtlich auf der CD enthalten sind. Auf der CD finden Sie auch eine Übersicht der in den Beispielen demo_01 bis demo_47 verwendeten Module (Pfad: \manual \pdf\deutsch \verzeichnis.pdf). Im Anschluß an diese Einführung wird die Definition einer Applikation, bestehend aus Signalgraph ohne eigene Benutzerschnittstelle, besprochen. Ferner werden Funktionen des Editors diskutiert und gezeigt, wie Sie die einzelnen Anwendungen auf komfortable Art und Weise dokumentieren.

Die grafische Benutzerschnittstelle ermöglicht es, die Applikation zu bedienen, ohne Detailkenntnis vom Signalgraphen zu besitzen. Es ist außerdem möglich eigene Algorithmenmodule (siehe Handbuch Modulprogrammierung) als DLL mit geeigneter Schnittstelle in die Entwicklungsumgebung einzubinden.

ICONNECT ist somit ein ständig wachsendes Softwarepaket, das gerade in der Entwicklung von Einzelsystemen ein ideales Werkzeug darstellt.

In der Online-Hilfe finden Sie weitere Informationen über Ein-/Ausgang, Funktionsweise, Parameter und dergleichen zu den einzelnen Modulen.

Viel Erfolg bei den ersten Schritten!

Ihr MICRO-EPSILON Team

Lizenzvertrag

Bitte lesen Sie diesen Software-Lizenzvertrag sorgfältig. Er enthält die Bedingungen, zu denen Ihnen eine ICONNECT-Lizenz gewährt wird. Wenn Sie mit den Bedingungen nicht einverstanden sind, schicken Sie die Ware mit der ungeöffneten Datenträgerpackung an Ihren autorisierten ICONNECT-Händler, der Ihnen den Kaufpreis zurückerstatten wird.

Achtung!
Die Anfertigung von Kopien von ICONNECT ohne die Zustimmung von MICRO-EPSILON ist rechtswidrig, verpflichtet Sie zu Schadenersatz und stellt eine strafbare Handlung gemäß §§106ff Urheberrechtsgesetz dar.

Achtung!
Der Besitz oder die gewerbliche Verwendung eines Programms, einer Vorrichtung oder sonstiger Gegenstände, die dazu bestimmt sind, die Entfernung oder Umgehung einer Kopierschutzvorrichtung oder der Knoten-Identifizierungsnummer, mit der dieses Programm versehen ist, zu ermöglichen, stellt eine Rechtsverletzung dar.

1. LIZENZ

MICRO-EPSILON gewährt Ihnen eine nicht übertragbare Lizenz für die Nutzug dieses Exemplars von ICONNECT und des Begleitmaterials auf in Ihrem Besitz oder unter Ihrer Kontrolle befindlichen Geräten gemäß den nachstehenden Bestimmungen:

A. EINZELPLATZ-INSTALLATION

Wenn Sie eine Lizenz für eine ICONNECT- Einzelplatzversion erworben haben, so darf Ihr Exemplar nur als Einzelplatzinstallation gemäß den nachstehenden Bestimmungen verwendet werden:

Die Nutzung dieses Exemplars von ICONNECT darf zu keinem Zeitpunkt von mehr als einer Person oder auf mehr als einem Einzelplatzrechner oder auf mehr als einer Arbeitsstation erfolgen.

B. ZUSÄTZLICHE RECHTE UND PFLICHTEN

Sie sind berechtigt, Kopien der Originaldatenträger als Archivkopien zu fertigen, die nur zu Sicherungszwecken benutzt werden dürfen. Im übrigen darf das ICONNECT-Programm und das Begleitmaterial nur vervielfältigt werden, soweit es im Rahmen dieser Lizenz ausdrücklich gestattet ist. Sie dürfen das ICONNECT-Programm oder Begleitmaterial zu keinem Zweck ändern, übersetzen, bearbeiten, umgestalten oder davon abgeleitete Werke herstellen. Die Übertragung des gewährten Nutzungsrechts auf Dritte ist zulässig, jedoch nur wenn (a) sämtliche Datenträger zusammen mit dem Begleitmaterial an den Dritten veräußert werden, (b) Sie keinerlei Kopien von ICONNECT (einschließlich Kopien auf der Festplatte) mehr behalten und (c) der Dritte sich ausdrücklich verpflichtet, sämtliche Bedingungen dieser Lizenz einzuhalten; mit einer solchen Übertragung erlischt Ihr Nutzungsrecht.

Für die Überlassung der Softwareprodukte an Kunden, die Wiederverkäufer sind gilt:
Endkunden darf nur ein Nutzungsrecht eingeräumt werden, wenn Sie sich verpflichten, die in diesem Lizenzvertrag genannten Bestimmungen zu beachten.

Hinweise, Schilder oder Warenzeichen von MICRO-EPSILON dürfen weder von Ihrem Exemplar von ICONNECT noch vom Begleitmaterial entfernt werden.

Das ICONNECT-Programm darf nicht rekonstruiert, dekompiliert oder disassembliert werden. Wenn Ihr Exemplar des ICONNECT-Programms mit einer Kopierschutzvorrichtung (Hardware Lock, Dongle) versehen ist, ist die Nutzung des Exemplars nur in Verbindung mit dieser Vorrichtung oder mit einer Ersatzvorrichtung gestattet, die von MICRO-EPSILON oder einem autorisierten ICONNECT-Händler geliefert wurde. Die Kopierschutzvorrichtung darf keinesfalls entfernt oder umgangen werden.

2. RECHTSINHABERSCHAFT

MICRO-EPSILON bleibt Inhaber der Urheber- und sonstigen Schutzrechte an ICONNECT und am Begleitmaterial, auch falls hiervon mit Zustimmung von MICRO-EPSILON Kopien angefertigt wurden. Die Software und die Dokumentation enthalten Geschäftsgeheimnisse von MICRO-EPSILON und/oder deren Lizenzgeber; sie sind urheberrechtlich geschützt. Der Kunde wird dies beachten, insbesondere Copyright-Vermerke und/oder Serialisierungsnummern nicht löschen.

Der Kunde wird die Software und die Dokumentation ohne schriftliche Zustimmung von MICRO-EPSILON Dritten, die nicht Wiederverkäufer oder Endkunde sind, nicht zugänglich machen.

3. UNERLAUBTES KOPIEREN

Wenn Sie das ICONNECT-Programm oder Begleitmaterial ohne Erlaubnis kopieren oder sonstige Bestimmungen dieser Lizenzvereinbarung verletzen, endet die Nutzungsberechtigung ohne weiteres. MICRO-EPSILON behält sich in diesem Fall alle Rechte vor.

4. GEWÄHRLEISTUNG UND HAFTUNGSAUSSCHLUSS

MICRO-EPSILON gewährleistet, daß ICONNECT auf den gelieferten Datenträgern ordnungsgemäß aufgezeichnet ist und daß das Programm im wesentlichen die in der Programmbeschreibung aufgeführten Funktionen hat. Unter Ausschluß jeder anderen Gewährleistung sind MICRO-EPSILON bzw. der autorisierte ICONNECT-Vertragshändler - nach unserer Wahl - bereit, innerhalb einer sechsmonatigen Frist, die mit der Übergabe des Programmpakets beginnt, entweder (a) die Brauchbarkeit des Programms mit angemessenem Aufwand und innerhalb einer angemessenen Zeit herzustellen, (b) Ihr Exemplar von ICONNECT durch funktionell gleichwertige Programme zu ersetzen oder (c) den Kaufpreis gegen Rückgabe des Produkts zu erstatten, womit die Lizenz endet.

Eine weitergehende Gewährleistungspflicht besteht nicht. Insbesondere besteht keine Gewährleistung dafür, daß ICONNECT Ihren speziellen Zweckerfordernissen genügt. Sie tragen die alleinige Verantwortung für Auswahl, Installation und Nutzung sowie für die mit dem Programm beabsichtigten Ergebnisse. MICRO-EPSILON übernimmt weder eine Gewährleistung für den ununterbrochenen oder fehlerlosen Betrieb des Programms noch für dessen Eignung für einen bestimmten Zweck.

5. HAFTUNGSBESCHRÄNKUNG

Soweit aus zwingenden Vorschriften eine Haftung von MICRO-EPSILON bestehen sollte, beschränkt sich diese auf schuldhaft verursachte und voraussehbare Schäden und auf einen Betrag nur bis zur Höhe des Verkaufspreises. Eine Haftung für entgangenen Gewinn, eingetretene Verluste, mittelbare Schäden und Folgeschäden ist ausgeschlossen, ebenso eine Haftung für Datenverluste. Der Nutzer erkennt an, daß diese Risikoverteilung in Anbetracht der Höhe der Lizenzgebühr angemessen ist. Diese Haftungsbeschränkungen gelten nicht für zugesicherte Eigenschaften und Schäden, die auf Vorsatz oder grober Fahrlässigkeit von MICRO-EPSILON beruhen oder die aus der Verletzung von solchen Vertragspflichten resultieren, die für die Errichtung des Vertragszwecks von grundlegender Bedeutung sind.

6. ALLGEMEINES

Auf diese Vereinbarungen ist nicht das Einheitliche U.N.-Kaufrecht, sondern ausschließlich das Recht der Bundesrepublik Deutschland anwendbar. Aus dieser Lizenzvereinbarung etwa entstehende Streitigkeiten unterliegen der deutschen Gerichtsbarkeit. Gerichtsstand ist Passau. Diese Lizenz erlischt ohne weitere Mitteilung oder Handlung seitens MICRO-EPSILON, wenn über Ihr Vermögen ein Konkurs- oder Vergleichsverfahren eröffnet wird oder Ihr Unternehmen aufgelöst wird. Diese Vereinbarung gibt alle etwaigen Abreden mit MICRO-EPSILON wieder und ersetzt alle etwaigen sonstigen Willenserklärungen und Werbeaussagen, die sich auf das Produkt ICONNECT und das Begleitmaterial beziehen. Mündliche Nebenabreden wurden nicht getroffen. Die Vertragsbestimmungen bleiben auch bei Unwirksamkeit einzelner Bestimmungen in ihren übrigen Teilen wirksam. Unwirksame Bestimmungen gelten als ersetzt durch solche Regelungen, die dem angestrebten wirtschaftlichen Erfolg in rechtlich zulässiger Weise möglichst nahe kommen. „MICRO-EPSILON“ und „ICONNECT“ sind eingetragene Warenzeichen der MICRO-EPSILON MESSTECHNIK GmbH & Co.KG.

1. Installation

Lesen Sie vor der Installation sorgfältig den Lizenzvertrag!

Für die Installation der **Vollversion** von ICONNECT führen Sie folgende Schritte aus:

1. Schließen Sie den mitgelieferten Dongle an die parallele Schnittstelle LPT1 Ihres Rechners an.

2. Legen Sie die CD in Ihr CD-ROM-Laufwerk ein.

Abb. 1.1: Start Bildschirm

Wählen Sie aus dem Hauptmenü den Eintrag **Installation** (siehe Abb. 1.1) und dann **ICONNECT Vollinstallation** aus.

3. Es erscheinen nun auf dem Bildschirm einige Informationen zur weiteren Vorgehensweise. Sie haben die Möglichkeit das Verzeichnis für ICONNECT zu bestimmen.

4. Im Anschluß an die Installation erfolgt die Lizensierung. Legen Sie nach Aufforderung die Lizenzdiskette in das Laufwerk ein und drücken Sie den Button **Start**.

Für die Installation der **Demoversion** von ICONNECT führen Sie folgende Schritte aus:

1. Legen Sie die CD in Ihr CD-ROM-Laufwerk ein. Wählen Sie aus dem Hauptmenü den Eintrag **Installation** (siehe Abb. 1.1) und dann **ICONNECT Vollinstallation** aus.

2. Es erscheinen nun auf dem Bildschirm einige Informationen zur weiteren Vorgehensweise. Sie haben die Möglichkeit das Verzeichnis für ICONNECT zu bestimmen.

3. Für die Demoversion ist keine Lizensierung nötig. Quittieren Sie den Dialog für die Lizensierung mit dem Button **Beenden**.

2. Editorfunktionen

ICONNECT meldet sich nach dem Start mit dem Anmeldedialog. Der Benutzername wird vorgegeben. Als Paßwort ist der Benutzername zu wiederholen und mit dem OK-Button zu bestätigen. Benützer der Demo-version klicken ohne Angabe des Paßworts auf den OK-Button. Ordnen Sie die Symbolleisten (siehe Abb. 2.1a) mit gedrückter linker Maustaste ähnlich Abb. 2.1b an.

Abb. 2.1a: Schieben der Symbolleiste

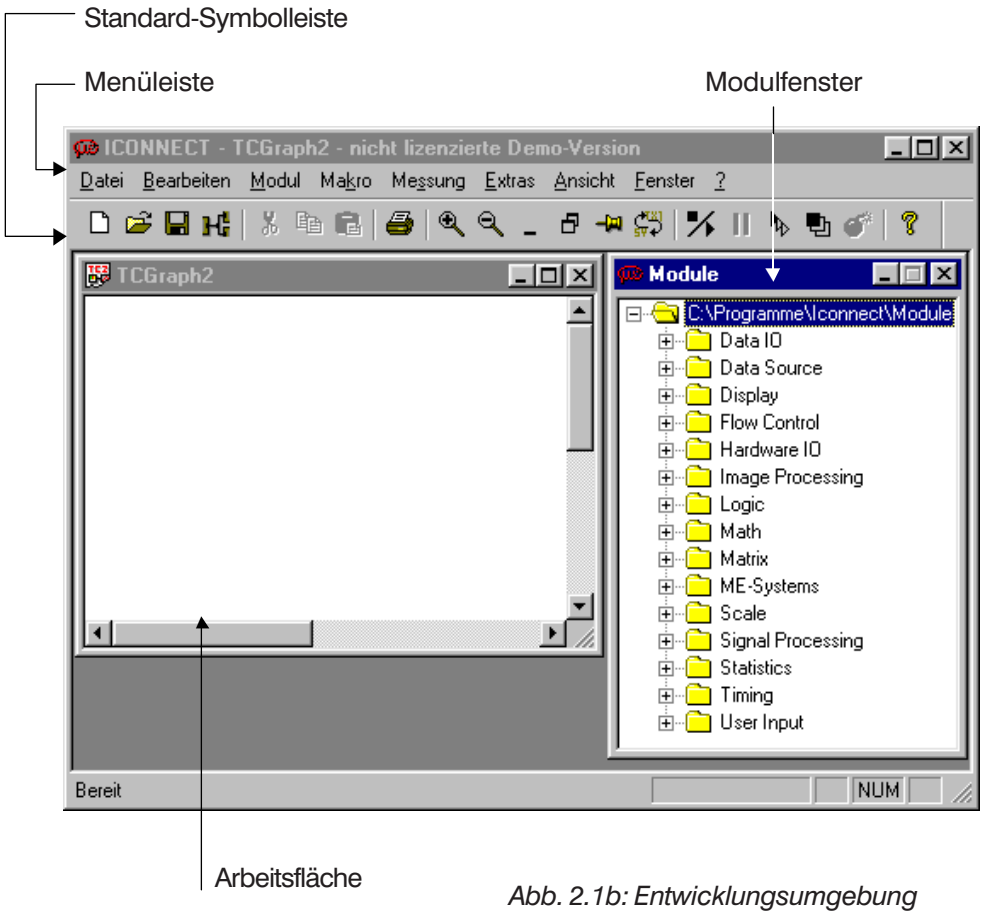


Abb. 2.1b: Entwicklungsumgebung von ICONNECT

Die **Menüleiste** erlaubt den Zugriff auf Editorfunktionen, die Steuerungsfunktionen für die Signalgraphen, sowie die Benutzerverwaltung (siehe Kap. 5).

Die **Standard-Symbolleiste** stellt den Zugriff auf die wichtigsten Funktionen von ICONNECT durch Anklicken von Tool-Buttons zur Verfügung.

Das **Modulfenster** ermöglicht die Auswahl von Modulen.

In der **Arbeitsfläche** wird der Signalgraph gezeichnet.

Mit dem in Abb. 2.2 vorgestellten Beispiel soll nun die Funktionsweise der Entwicklungsumgebung erarbeitet werden.

Hinweis:
Im folgenden wird als Links-Klick das Betätigen der linken Maustaste bezeichnet, als Rechts-Klick das der rechten. Unter einem Doppel-Links-Klick wird das zweifache Ausführen eines Links-Klick verstanden. Analoges gilt für den Doppel-Rechts-Klick.

2.1 Zeichnen von Signalgraphen

Die Anwendungen werden durch das Zeichnen von Signalgraphen erstellt. In diesem Kapitel lernen Sie die Grundlagen hierzu. Als Beispiel sehen Sie die Parametrisierung eines Funktionsgenerators und die Darstellung seines Ausgangssignals auf einem Display.

2.1.1 Der Signalgraph, das ICONNECT-Programm

Abb 2.2 zeigt ICONNECT mit zwei geöffneten Fenstern am Bildschirm. Im Fenster mit der Bezeichnung **demo_01 - Signalgraph aktiv** ist das Beispiel eines Signalgraphen dargestellt. Die einzelnen Module führen unterschiedliche Aufgaben aus, wie z.B. I/O-Operationen, logische und arithmetische Funktionen bis zur Visualisierung von bearbeiteten Meßdaten. Alle Module besitzen auf der linken Seite ihre Eingänge und auf der rechten Seite die Ausgänge. Über diese Modulports werden die Daten in den parametrisierten Algorithmus eingespeist und nach der Bearbeitung ausgegeben. Das Fenster mit der Bezeichnung **AnalogChart1** ist das zum gleichnamigen Modul gehörende Displayfenster. Es visualisiert die verarbeiteten Meßdaten. Der Signalgraph wird über die Icons in der Standard-Symbolleiste

 Start / Stop

 Pause

oder per Menü gesteuert. Im Menü **Messung** sind dies die Befehle **Start**/ **Stop** bzw. **Pause**.

Die Funktion eines Signalgraphen ist aus dem Blockschaltbild erkennbar. Im Signalgraph aus Abb. 2.2 wird im Modul **SinGen1** (FuncGen) ein sinusförmiges Signal erzeugt. Das Modul **Scale1** skaliert dieses Signal auf einen Bereich von -1 bis 1 Milli-Ampere. ICONNECT führt in allen Kommunikationskanälen Informationen über die charakteristischen Eigenschaften eines Signals mit. Diese enthalten auch die Einheit und den Bereich eines Signals. In jedem Modul werden diese Informationen weitergegeben oder modifiziert. In den Displays wird somit die Anzeige der richtigen Einheiten gewährleistet (siehe Kap. 4). Im **AnalogChart1** werden die aufgenommen Daten gemäß den Parametereinstellungen an das Displayfenster geschickt und dort angezeigt.

Im folgenden wird die Realisierung dieses Beispiels besprochen, dabei wird auf die einzelnen Funktionen des Editors und des Hilfesystems eingegangen.

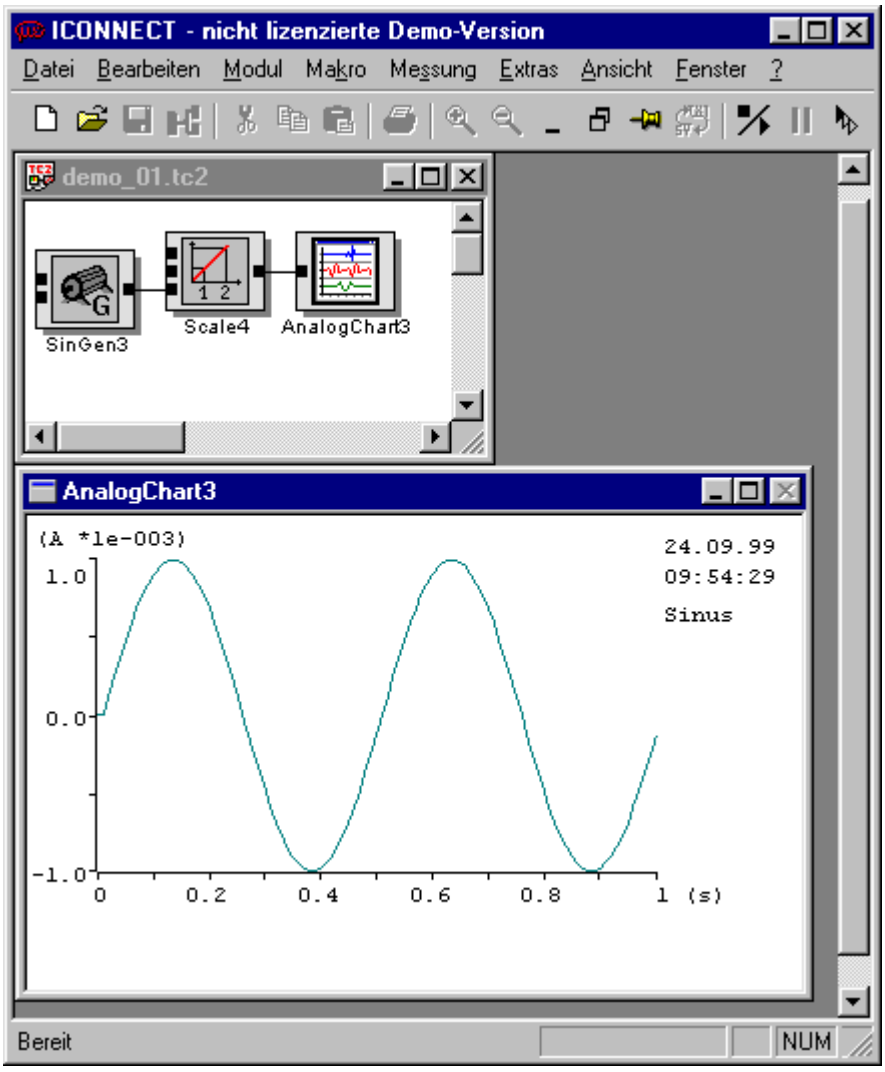


Abb. 2.2: ICONNECT Programm

Hinweis:
Alle in diesem Tutorial besprochenen bzw. vorgestellten Beispiele finden Sie im Verzeichnis c:\examples\demo, wobei c der Buchstabe Ihrer Festplatte ist.

2.1.2 Auswahl der Module

Um den Funktionsgenerator auszuwählen, führen Sie folgende Schritte aus:

1. Links-Klick auf den Menüeintrag **Module**

2. Auswahl der Einträge **User I/O > FuncGen**, erneuter Links-Klick. Das Icon, das den Algorithmus symbolisiert, erscheint auf der Arbeitsfläche.

3. Mit der Maus kann das Icon über die Arbeitsfläche bewegt werden. Plazierung des Icons durch Links-Klick.

Die beiden anderen Module **Scale** (**Module > Scale**) und **AnalogChart** (**Module > Display**) werden analog ausgewählt und plaziert.

2.1.3 Definition der Modulparameter

Ein Modul besitzt im allgemeinen einen modulspezifischen Dialog. Sie haben zwei Möglichkeiten den Dialog zu starten. Entweder durch einen Doppel-Links-Klick auf das Symbol eines Modulicons oder durch einen Rechts-Klick auf das Modulicon, wodurch ein Pop-Up-Menü geöffnet wird. Hier muß der Eintrag **Eigenschaften** durch einen Links-Klick ausgewählt werden.

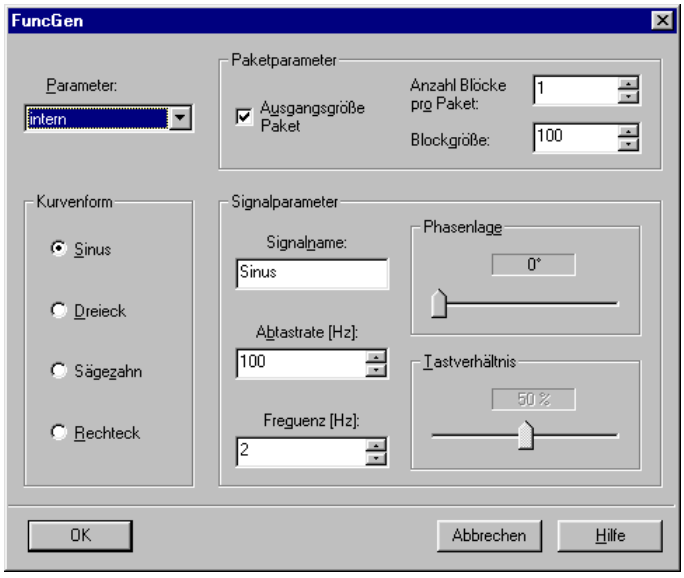


*Abb. 2.3: Dialog des Moduls **FuncGen***

Ein Funktionsgenerator generiert Testdaten. Es werden die Kurvenformen Sinus, Rechteck, Dreieck und Sägezahn angeboten. Weitere Informationen zu dem Modul sind über die Online-Hilfe abrufbar.

Die Paketparameter **Blockgröße** und **Anzahl Blöcke pro Paket** charakterisieren den Datenstrom zwischen den Modulen. ICONNECT unterstützt eine blockorientierte Kommunikation. Daten, die zu einer definierten Einheit gehören, werden zu einem Paket zusammengefaßt. Ein Paket stellt eine Menge semantisch zusammengehöriger Daten dar, z.B. ein Bild, eine Messung oder die Daten eines bestimmten Werkstücks. Das Paket wird mit Informationen, dem sog. **TypeInfo** versehen (siehe Kap. 4).

Ein Paket besteht aus mindestens einem Datenblock und dieser wiederum aus mindestens einem Datum (z.B. einem Meßpunkt). Die zweistufige Aufteilung in Pakete und Blöcke ist sinnvoll, da auf diese Weise die Berechnung von Zwischenergebnissen bei Paketen möglich ist, die äußerst umfangreich sind bzw. über einen großen Zeitraum gemessen werden. Bei der Entwicklung von Signalgraphen ist zu beachten, daß manche Module nur paketweise bzw. nur blockweise arbeiten. Mit der Checkbox **Ausgangsgröße Paket** kann eingestellt werden, ob ein Paket mit der entsprechenden Anzahl Blöcken als Ausgangssignal generiert wird, oder wenn dies nicht angewählt ist ein endlos laufendes Paket, also eine Dauermessung erzeugt wird.

2.1.4 Auswahl der Parameterquellen in ICONNECT.

Parameter können auf drei verschiedene Arten erzeugt werden:

1. Als **interne** Parameter werden solche bezeichnet, die im Dialog definiert werden.

2. Wird als Parameterquelle **DB** (Datenbank) gewählt, so können über den Modulport **DB** z.B. beim Funktionsgenerator die Frequenz und die Abtastrate während der Applikationslaufzeit aus einer Datenbank eingelesen werden.

3. Mit dem Eingang **EXT** können beliebige Daten aus dem Signalgraphen für die entsprechenden Parameter benutzt werden, wenn als Quelle **extern** angeklickt ist (siehe dazu Kap. 3.2.2).

Hinweis:
*Mehr zu **Blockgröße**, **AnzahlBlöcke pro Paket**, **Paket** finden Sie in Kap. 4.*

2.1.5 Verdrahten der Modulports

Wird mit dem Mauszeiger ein Modulport berührt, so erscheint der Name des Ein- bzw. Ausgangs. Durch einen Links-Klick auf den Modulport wird der Anfang eines Kommunikationskanals erzeugt. Ein Links-Klick auf den Port des korrespondierenden Moduls stellt die Verbindung fertig. Eine versehentlich begonnene Verbindung brechen Sie durch Links-Klick auf das Arbeitsblatt ab.
ICONNECT „säubert" Ihre Arbeitsfläche (die Modul-Icons werden neu angeordnet und mit geraden Linien verdrahtet), wenn Sie aus dem Menü **Bearbeiten > Rechtwinklige Verdrahtung** wählen.

2.1.6 Speichern und Laden von Signalgraphen

Durch einen Links-Klick auf den Menüeintrag **Datei** und die Auswahl von **Speichern unter** wird der Dateidialog von WINDOWS geöffnet. Das Suffix **.tc2** wird automatisch an den Dateinamen angehängt, der in der entsprechenden Eingabezeile eingetragen wird. Durch einen Links-Klick auf den Speichern-Button wird die Datei abgespeichert.

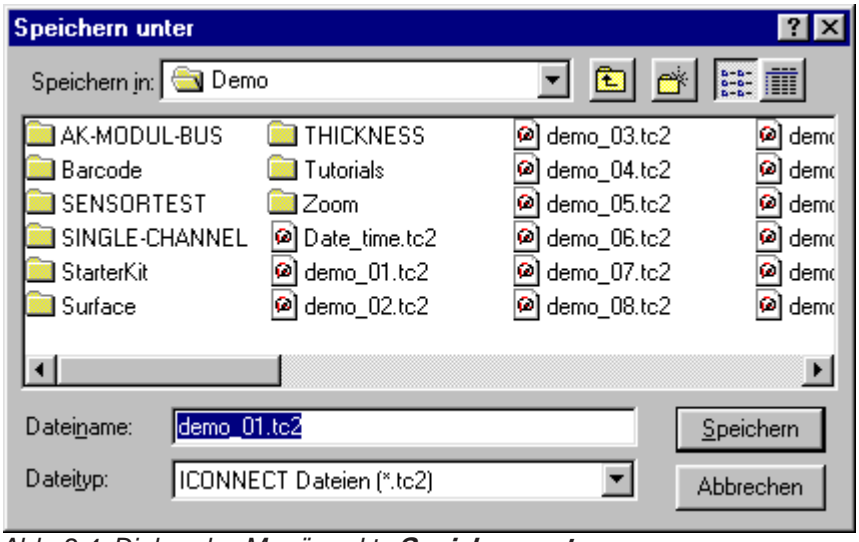


*Abb. 2.4: Dialog des Menüpunkts **Speichern unter***

Für das Laden von Signalgraphen ist im Menü **Datei** der Eintrag **Öffnen** oder das **Ordner-Symbol** zu wählen. Analog zum Speichern einer Datei sind im Dialog die fehlenden Angaben einzutragen und zu quittieren. Die bisher beschriebenen Aktionen können auch durch die im Menü angegebenen **Tastatur-Shortcuts** ausgeführt werden.

2.2 Online-Hilfe

Sie starten die Online-Hilfe durch Auswahl des **Fragezeichens** in der Menüleiste und die Auswahl des Unterpunktes **Hilfethemen.** Auch die Information für die Module kann neben der oben angesprochenen Möglichkeit durch einen Rechts-Klick auf das Modulicon und der Auswahl des Hilfe-Eintrags abgerufen werden.

2.3 Bearbeiten der Module im Signalgraph

Löschen

Markieren Sie mit einem Links-Klick das zu löschende Modul und betätigen Sie anschließend die **Entf-Taste** der Tastatur. Alternativ löschen Sie ein Objekt mit dem Menü-Punkt **Löschen** im Menü **Bearbeiten**.

Ausschneiden, Kopieren und Einfügen

Markieren Sie das Modul und wählen Sie eines von drei in Windows bekannten Verfahren.

- Menü **Bearbeiten**
- Betätigen der Buttons **Schere**, **Doppeltes-Dokument** und **Briefumschlag**
- Eingabe des entsprechenden, im Menü **Bearbeiten** angegebenen, **Tastatur-Shortcuts** (z.B. **Strg X** für Ausschneiden).

Modul verschieben

Markieren Sie das Modul und bewegen es mit festgehaltener linker Maustaste. Mit dem Loslassen der Taste ist das Modul positioniert.

Hinweis:
Die Modul-Beschriftung in einem Signalgraphen besteht aus **Modul**-Name *und* **Modul**-Nummer *(z.B. **Scale1**, Abb. 2.2). In einer ICONNECT-Sitzung werden die Modulnummerierungen hochgezählt. Abweichungen zwischen den abgebildeten Signalgraphen im Tutorial und auf Ihrem Bildschirm sind deshalb möglich.*

2.4 Bearbeiten einer Modulgruppe im Signalgraphen

Die Aktionen aus Kap. 2.3 gelten auch für eine Gruppe von Modulen. Mehrere Module markieren Sie durch

- Auswahl der Objekte mit Links-Klick und gedrückter Shift-Taste
- Zeichnen eines Markierungsrahmens. Halten Sie dazu auf dem Arbeitsblatt die linke Maustaste gedrückt und ziehen Sie mit der Maus über die gewünschten Objekte einen Rahmen.

Im Menü **Bearbeiten** können Sie mit dem Eintrag **Alles Markieren** den gesamten Signalgraphen selektieren. Um die Markierung einer Gruppe aufzuheben ist ein Links-Klick auf das Arbeitsblatt auszuführen.

2.5 Beispiel

Nachdem die wichtigsten Funktionen der Entwicklungsumgebung vorgestellt sind, sollen Sie die Handhabung an einem Beispiel vertiefen. Dazu generieren Sie einen Zufallszahlengenerator. Pro Sekunde wird ein Wert erzeugt und in einem digitalen Display visualisiert. Das fertige Programm ist im Pfad c:\examples\demo\demo_02 abgelegt. Wir empfehlen Ihnen das Programm selbständig zu erstellen. Führen Sie dazu folgende Schritte aus:

1. Auswahl und Plazierung des Moduls **Random** über das Menü **Module > User I/O > Random** oder aus der Modulleiste **User I/O** das entsprechende Symbol. Wird ein Symbol mit dem Mauszeiger angewählt, so wird sein Name in einem Tool-Tip angezeigt.

2. Doppel-Links-Klick auf das Modulicon, um den Parameterdialog zu öffnen (siehe Abb. 2.5). Einstellung der Parameter Blockgröße = 1, Abtastrate = 1 Hz und Gauss-Verteilung erzeugt jede Sekunde eine Zufallszahl. Wie bereits vorher angesprochen entscheidet der Parameter **Paket** über die Charakterisierung der Ausgangsdaten. In diesem Beispiel werden Pakete übertragen, die jeweils einen Datenblock mit einem Wert enthalten.

3. Auswahl und Plazierung des Moduls **DigitalDisp** (siehe Kap. 2.1.2)

4. Verbindung der beiden Ports (siehe Kap. 2.1.5)

5. Betätigen des Start-Symbols

*Abb. 2.5: Dialog für das Modul **Random***

In Abb. 2.6 sehen Sie das Ergebnis Ihrer bisherigen Arbeit.

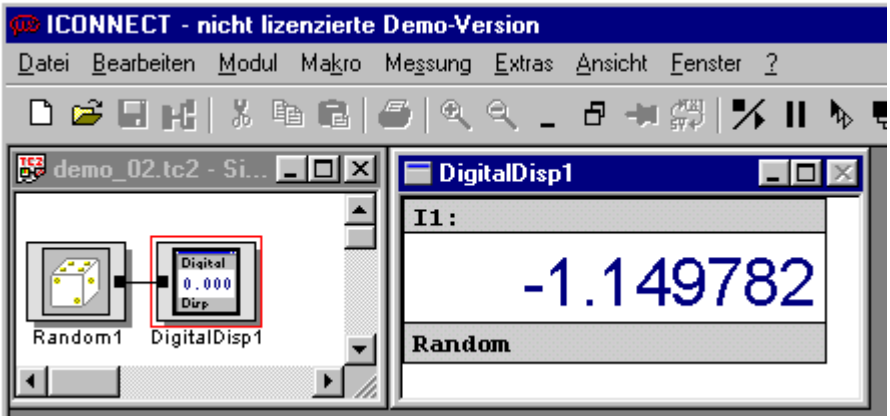


Abb. 2.6: Generierung einer Zufallszahl in ICONNECT

Als nächstes untersuchen Sie die Möglichkeiten des digitalen Displays. Gehen Sie wie folgt vor:

1. Öffnen Sie den **Parameterdialog** des Moduls **DigitalDisp** durch einen Doppel-Links-Klick auf das Modul-Icon (siehe Abb. 2.7). Sie können dies auch durch einen Rechts-Klick und Auswahl des Menüunterpunktes **Eigenschaften** erreichen.

Ein digitales Display kann in seinem Fenster bis zu acht Signale visualisieren. Als Defaultwert wird der Wert 1 angenommen, der in diesem Beispiel auch nicht verändert wird. Für jeden Eingang können Sie einen Signalnamen definieren. Wenn Sie nichts eintragen, wird im Display der Name angezeigt, den das Signal im TypeInfo mit sich trägt (siehe Kap. 4). Im Fall des Zufallszahlengenerators ist dies der Name Random. Wünschen Sie keinen Text, fügen Sie im Feld **Signalnamen** ein Leerzeichen ein.

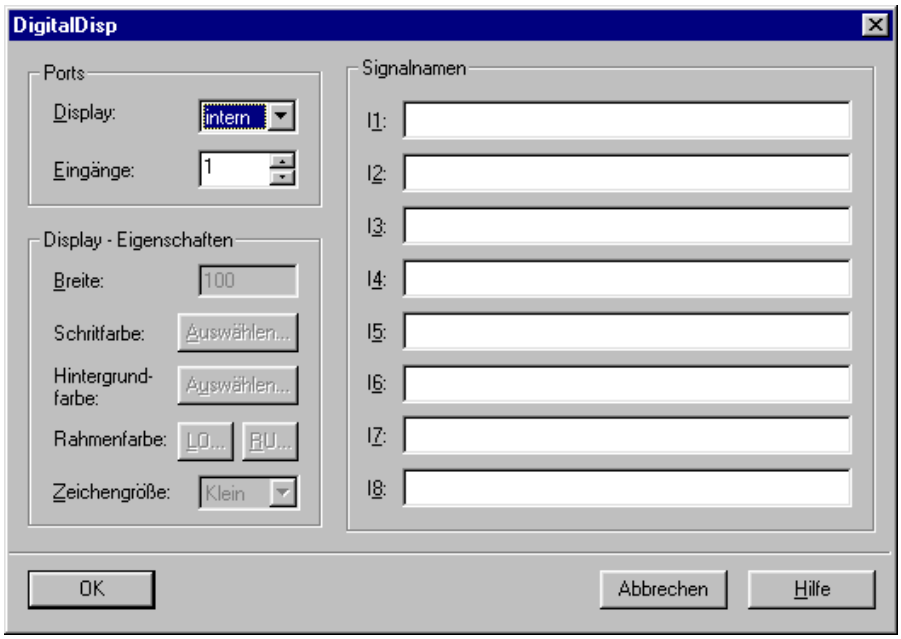


*Abb. 2.7: Dialog des Moduls **DigitalDisp***

2. Tragen Sie den Signalnamen ***Signal 1*** bei **I1** ein.

Mit den beiden Buttons **Schriftfarbe** und **Hintergrundfarbe** können Sie über den Farb-Dialog von WINDOWS die Farbgebung der Schrift und des Hintergrundes einstellen.

3. Setzen Sie die Schriftgröße in der Rubrik **Typ** auf **Large**.

Es fällt auf, daß durch die hohe Anzahl von Nachkommastellen der anzuzeigende Wert nicht vollständig in das Fenster paßt. Aus diesem Grund modifizieren Sie jetzt die Anzahl der Nachkommastellen.

4. Rechts-Klick auf die Leitung zwischen den beiden Modulen. Im aktivierten Menü wählen Sie in den Unterpunkt **Eigenschaften**.

Diese Aktion öffnet den Eigenschaftsdialog des Kommunikationskanals (siehe Abb. 2.8).

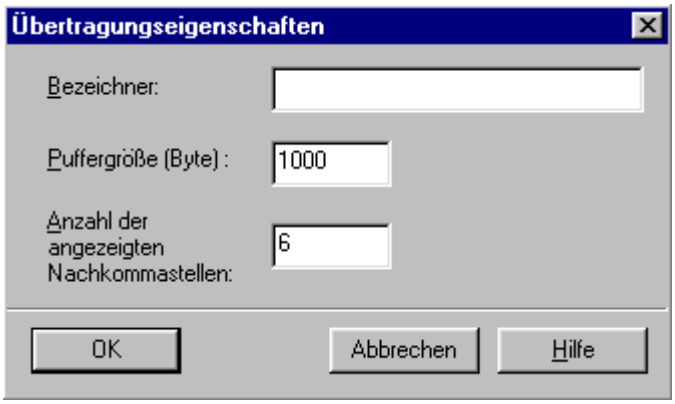


Abb. 2.8: Eigenschaften eines Kommunikationskanals

Sie können zwei Parameter variieren. Die Puffergröße gibt die Speicherkapazität des Kommunikationskanals an. Als Defaultwert ist hier 1000 Byte gewählt. Sollte mehr benötigt werden, so reallokiert das Programm den entsprechenden Speicherplatz. Deshalb kann es bei großen Datenmengen möglich sein, daß der erste Durchlauf des Signalgraphen etwas länger dauert, da eventuell die Speicherkapazität für die Kommunikationskanäle angepaßt werden muß. Die Genauigkeit des Kommunikationskanals, diese ist mit 6 vorinitalisiert, bestimmt die Anzahl der Nachkommastellen.

5. Reduzieren Sie die Leitungsgenauigkeit auf 3 Nachkommastellen und quittieren Sie die Eingaben. Die Änderungen werden erst mit Start des Programms wirksam.

Hinweis:
*Im Menü **Extras > Standardeinstellungen** können Sie die Leitungsgenauigkeit für künftige Kommunikationskanäle global festlegen.*

2.6 Zusammenfassung

In diesem ersten Kapitel haben Sie die Funktionen des Grapheditors in ICONNECT und die Grundlagen zur Erstellung von einfachen Signalgraphen kennengelernt. Dabei wurden Details in einigen Modulen diskutiert. Es wurde das Type-Info angesprochen, das jedes Datenpaket enthält, sowie die Möglichkeit der Parametrisierung von Kommunikationskanälen.

Im nächsten Kapitel wird die Modulbibliothek von ICONNECT in mehreren Beispielen erläutert. Dabei wird das Hauptaugenmerk auf diejenigen Module gelegt, die häufig gebraucht werden.

3. Die Modulbibliothek von ICONNECT

In diesem Kapitel erweitern Sie Ihre Kenntnisse zum Zeichnen von Signalgraphen durch die Vorstellung neuer Module aus der Modulbibliothek von ICONNECT. Sie

- lernen **arithmetische Module** und deren Eigenschaften bezüglich der Kommunikation kennen.
- diskutieren die Module aus der Gruppe **User Input**, mit deren Hilfe Sie interaktiv in den Signalgraphen eingreifen können.
- bauen mit den Modulen **Displaymanager** und **Inputmanager** Benutzeroberflächen auf.

3.1 Arithmetische Module

3.1.1 Das Modul VecOpVec

Als Beispiel addieren Sie die Signale der Module **FuncGen** und **Random** (siehe Abb. 3.5). Führen Sie dazu folgende Schritte aus:

1. Plazieren Sie das Modul **FuncGen** (**Module > Data Source**). Nehmen Sie folgende Eingaben im Dialog vor:
   Abtastrate 512 Hz
   Blockgröße 512 Werte
   Perioden pro Block 2 Hz.

Mit diese Einstellungen erhält man zwei Perioden pro Sekunde mit jeweils 256 Werten pro Periode.

2. Holen Sie das Modul **Random** in die Arbeitsfläche.
   Parameter: Blockgröße 512 Werte
   Abtastrate 512 Hz
   Verteilung Uniform
   Ausgangsgröße Paket

Uniform erzeugt Werte im Bereich zwischen 0 und 1 mit gleicher Häufigkeit.

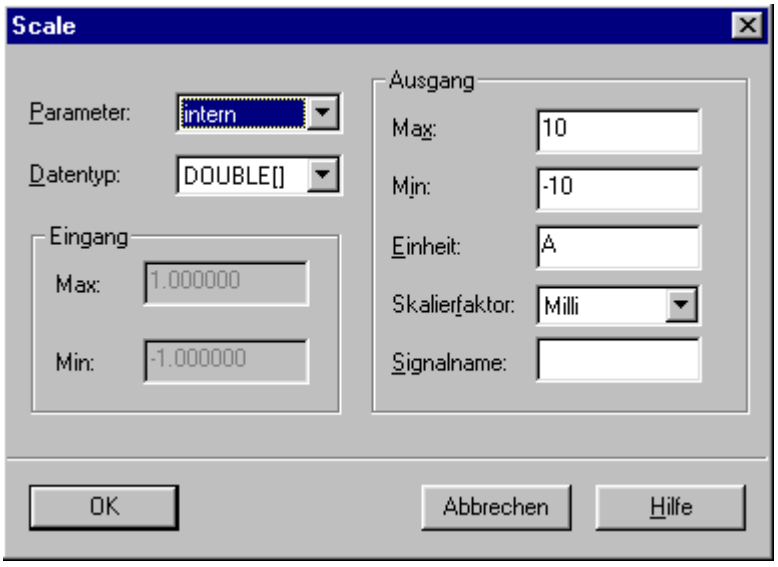


*Abb. 3.1: Parameter für das Modul **Scale1***

X975(

3. Wählen Sie das Modul **Scale1** (**Module > Scale**)und belegen Sie die Parameter lt. Abb. 3.1.

Für die Ausgänge werden als **Min**-Wert -10 und als **Max**-Wert 10 angegeben. Diese Information gibt den Bereich des Signals an. Wird in einem Display für den darzustellenden Bereich der Ordinate **Type-Info** gewählt, so wird diese Bereichsangabe als Skalierung benutzt.
Als Einheit wird **A** für Ampere und **Milli** ($= 10^{-3}$) für den Skalierungsfaktor angegeben. Wie bereits angesprochen wird diese Einheit, die ebenfalls zum Type-Info gehört durch den gesamten Signalgraphen mitgeführt und in den arithmetischen Modulen verrechnet und überprüft (siehe Beschreibung Modul „VecOpVec“).
Der Parameter **DOUBLE[]**[1] bestimmt den Datentyp für das Modul. Dies bedeutet, daß das Modul am Eingang einen Vektor von reellen Zahlen beliebiger Länge erwartet. Alternativ könnte der Typ **DOUBLE** definiert werden, welcher für eine einzelne reelle Zahl steht, die auch als reeller Skalar bezeichnet wird.
Mit Hilfe dieser Typbezeichnungen wird eine Validierung der Kommunikationsstruktur bereits bei der Verdrahtung von Modulen durchgeführt.

4. Wählen Sie das Modul **Scale2**. Die Parameter stimmen bis auf den Signalbereich, der mit -5 bis 5 spezifiziert werden soll, mit denen von Scale1 überein.

5. Fügen Sie das Modul **VecOpVec** aus dem Menü **Module > Math** der Arbeitsfläche hinzu.

Das Modul **VecOpVec** kann einfache arithmetische Verknüpfungen mit zwei reellen Vektoren durchführen. Die Operation Addition erfordert gleiche physikalische Einheiten. In den Schritten 3 und 4 wird dies realisiert, denn das Modul **VecOpVec** prüft die Einheiten auf Konsistenz. Unterschiedliche Einheiten haben bei additiven Verknüpfungen eine Fehlermeldung zur Folge. Bei multiplikativen Operationen wird eine entsprechende neue Einheit berechnet. Weitere Vorraussetzungen für eine erfolgreiche arithmetische Verknüpfung von Vektoren sind gleiche Blocklänge und Abtastrate.

---

1) Die Bezeichnung **DOUBLE[]** entspricht dem Arraytyp double in der Programmiersprache C.

6. Öffnen Sie den Dialog von **VecOpVec** (siehe Abb. 3.2) und ändern Sie den Signalnamen von **VecOpVec** auf **Sinus+Noise**.

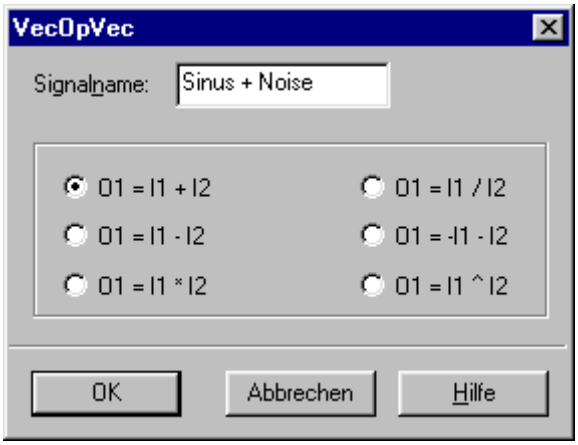


*Abb. 3.2: Dialog des Moduls **VecOpVec***

7. Holen Sie das Modul **AnalogChart** aus dem Menü **Module > Display** in die Arbeitsfläche.

Die Parametrisierung soll in diesem Beispiel etwas genauer besprochen werden. Der Einstiegsdialog für dieses Modul ist in Abb. 3.3 zu sehen. Mit dem Parameter **Anzahl** im Abschnitt **Kanäle** werden 3 Eingänge definiert und im darunter angeordneten Tree-View angezeigt. Durch einen Doppel-Links-Klick auf einen solchen Eintrag aktivieren Sie den Dialog für das Kanalsetup (siehe Abb. 3.4). Vergeben Sie für den **X-Bereich** einen Wert von 1 s. Signalname und y-Achsenbeschriftung werden aus dem **Type-Info** übernommen. Dies geschieht automatisch, wenn in die entsprechenden Eingabezeilen nichts eingetragen wird. Mit dem Button **Farbe** können Sie die Farbe für den Graph einstellen.

8. Verbinden Sie die Module gemäß Abb. 3.5 und betätigen Sie den Startbutton.

Abb. 3.6 zeigt die Visualisierung der einzelnen Signale. Neben den Einheiten, Wertebereichen und Signalnamen der einzelnen Datenströme wird auch das Datum und die Uhrzeit der Signalgenerierung dargestellt. Diese als Zeitstempel bezeichnete Information wird ebenfalls im **Type-Info** mitgeführt.

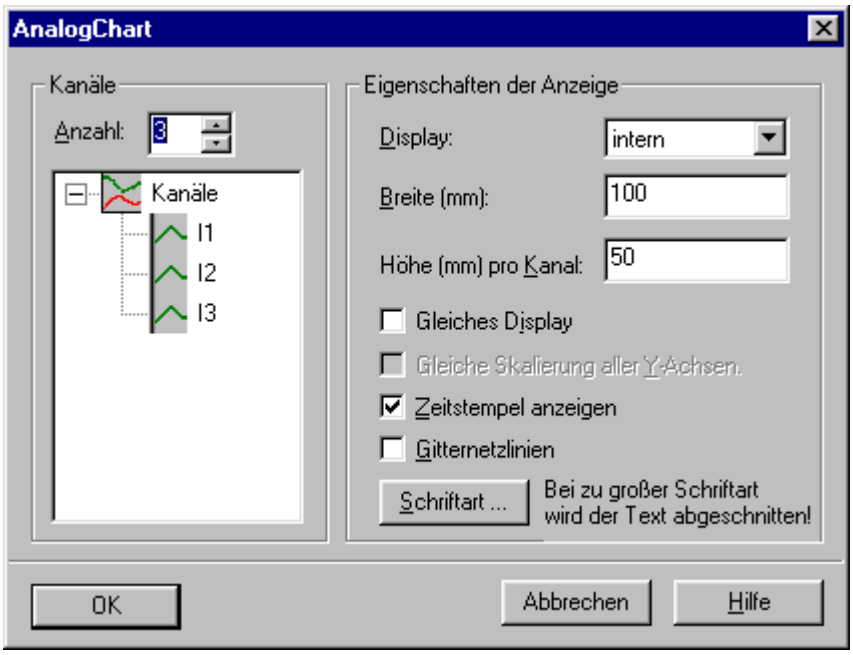


*Abb. 3.3: Dialog des Moduls **AnalogChart***

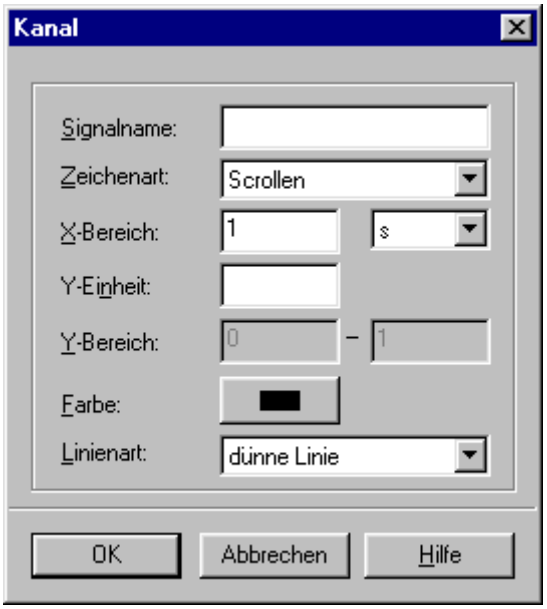


*Abb. 3.4: Kanalsetup des Moduls **AnalogChart***

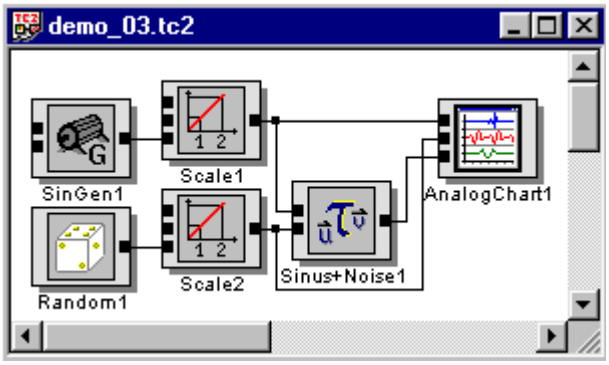


*Abb. 3.5: Signalgraph von **demo_03***

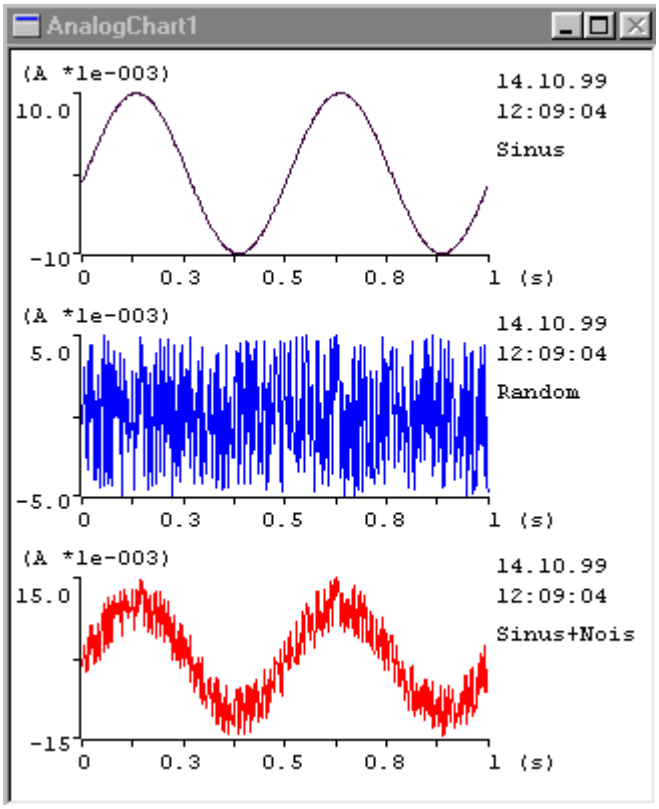


Abb. 3.6: Addition von Sinus-Signal mit weißem Rauschen

Hinweis:
Wählen Sie als Verknüpfung ***Multiplikation*** *(O1 = I1 * I2) anstatt* ***Addition*** *im Modul* ***VecOpVec*** *und ändern Sie die* ***Einheit*** *von* ***A*** *(Ampere) nach* ***V*** *(Volt) im Modul* ***Scale2****. Starten Sie das Programm erneut und beobachten Sie die Änderungen gegenüber Abb. 3.6. Im untersten Koordinatensystem wird als* ***Einheit W*** *also* ***A*V*** *angezeigt.*

3.1.2 Das Modul VecOpScal

Im vorhergehenden Beispiel wurde das Modul **VecOpVec** zur arithmetischen Verknüpfung zweier reeller Vektoren vorgestellt. In der Praxis besteht oft die Notwendigkeit, einen Vektor mit einem Skalar zu verknüpfen. Beispielsweise soll bei einer Meßaufgabe ein Meßsignal mit einem Korrekturwert zur thermischen Stabilisierung verrechnet werden. Das Signal liegt in der Regel als Vektor vor und der Korrekturfaktor als Skalar. Das Modul **VecOpScal** besitzt folgende Eigenschaften:

- Ein Skalar am Eingang **SCA** wird immer dann eingelesen, wenn sich der Wert des Skalars geändert hat.
- Liegen am Eingang **VEC** Daten an, so werden diese mit dem aktuellen Skalar verknüpft.
- Liegen am Eingang **VEC** Daten an, ohne daß zu diesem Zeitpunkt bereits Daten am Eingang SCA gelesen wurden, so wird eine Verknüpfung mit dem neutralen Element dieser Operation durchgeführt.

Im Beispiel (**demo_05**) ist die Amplitude des weißen Rauschens interaktiv einstellbar. Für die Realisierung des Programms sind die folgenden aufgelisteten Schritte notwendig:

1. Stellen Sie das Modul **SliderV** aus dem Menü **Module > User Input** in die Arbeitsfläche.

Module aus der Gruppe **User Input**, die Zugriffe auf den Signalgraphen während der Applikationslaufzeit zulassen, öffnen bei ihrer Instanziierung ein Fenster mit dem entsprechenden Windows-Dialogelement.

2. Tragen Sie im Dialog des Moduls **SliderV** (siehe Abb. 3.7) als **Signalname** die Bezeichnung **Amp** ein. Als Signaltyp wird mit **DOUBLE[1]** ein Vektor der Länge 1 definiert.

3. Holen Sie ein Modul **Scale** in die Arbeitsfläche. Skalieren Sie damit das Signal des Moduls **SliderV** , das Werte im Bereich von 0 bis 1 ausgibt. Als Skalierungsfaktor wählen Sie **Milli**. Geben Sie keine **Einheit** an, denn mit diesem Signal wird weißes Rauschen multipliziert. Der daraus resultierende Datenstrom muß wegen der Addition mit dem Signal Sinus die **Einheit** A besitzen.

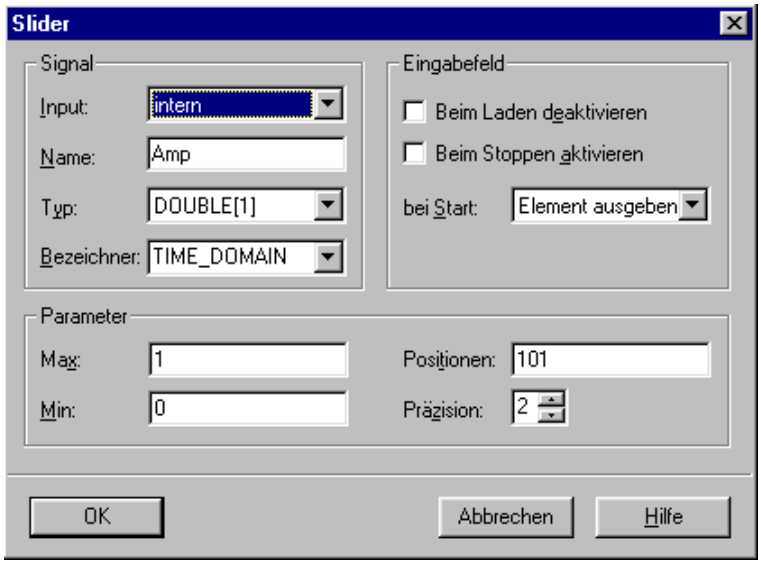


Abb. 3.7: Dialog des Moduls ***SliderV***

4. Fügen Sie das Modul **VecOpScal** aus **Module > Math** dem Signalgraphen hinzu. Der Dialog entspricht dem des Moduls **VecOpVec**. Wählen Sie als **Operation** die Funktion **Multiplikation**. Tragen Sie **Amp** als **Signalname** ein.

5. Vervollständigen Sie das Programm mit dem Signalgraphen aus dem Beispiel **demo_04** und verbinden Sie die Module miteinander (siehe Abb. 3.8).

Bei der Verdrahtung der Module **Scale3** und **VecOpScal** wird eine Gültigkeitsüberprüfung durchgeführt:

Übereinstimmung der Datentypen
Übereinstimmung der Bezeichner

Führt die Übereinstimmung der Bezeichner zu keiner positiven Entscheidung wird eine

Abfrage

durchgeführt. ICONNECT gibt eine Warnung aus, daß die Typbezeichner nicht übereinstimmen. Diese Warnung erfolgt zweimal. Sie ist in diesem Beispiel mit **Ja** zu bestätigen. Diese Warnung kann für jede Richtung bestätigt werden, falls Sie mit den Bezeichnern einverstanden sind.

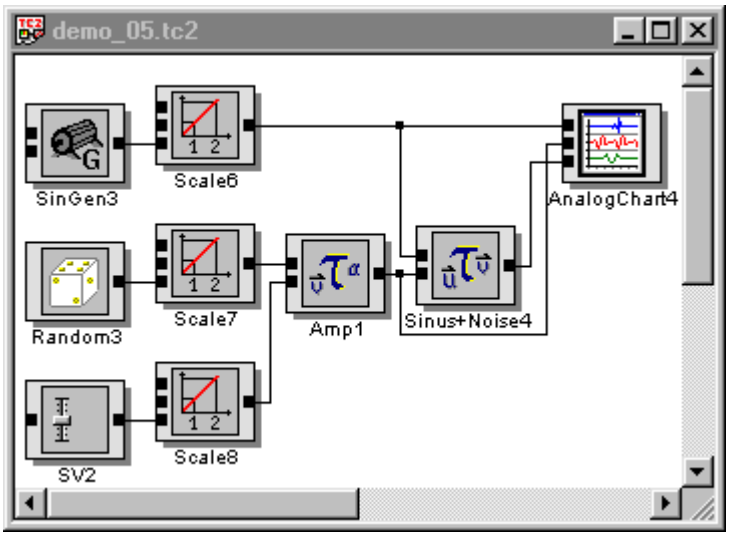


*Abb. 3.8: Signalgraph von **demo_05***

5. Betätigen Sie den Start-Button. Mit der Maus können Sie am Bedienelement **Amp** (siehe Abb. 3.9) die Amplitude des weißen Rauschens einstellen, die zum Sinussignal addiert wird.

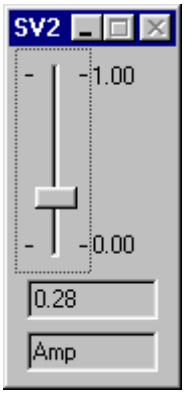


*Abb. 3.9: Bedienelement des Moduls **SliderV***

Hinweis:
Wird eine Verdrahtung zwischen zwei Modulen hergestellt, so wird ein Vergleich der entsprechenden Validierungsstrings durchgeführt. Dazu übergibt der Grapheditor den String des Quellmoduls an das Zielmodul. Dieses vollzieht einen Vergleich der Zeichenketten und bestätigt die Kompatibiltät der Ports, bzw. verneint diese. Als nächstes wird der String vom Zielmodul an das Quellmodul übergeben und dieses führt die Prüfung durch. Schlägt die Validierung bei einem der Module fehl, so wird der Kommunikationskanal nicht aufgebaut. Die doppelte Prüfung ist notwendig, da manche Module sich bei dieser Prüfung in Bezug auf ihre Arbeitsweise an das Partnermodul anpassen. Diese können sowohl Quell- als auch Zielmodul einer Verbindung sein.

Im Validierungsstring wird zuerst der Datentyp der Verbindung genannt. Im Falle des Moduls ***Scale*** *ist dies* ***DOUBLE[]****. Das Modul* ***VecOpScal*** *erwartet am Eingang* ***SCA*** *Daten des Typs* ***DOUBLE[1]****. Diese Typen sind miteinander kompatibel. Das Modul* ***VecOpScal*** *erkennt zur Applikationslaufzeit, ob* ***Scale*** *mehr als einen Datenwert pro Block überträgt.*

Zusätzlich erhält das Signal, das auf dem entstehenden Kommunikationskanal transportiert werden soll, eine Bezeichnung (= Typbezeichner oder Interpretation). Der Typ ***DOUBLE*** *sagt noch nichts darüber aus, ob es sich um ein Zeit-, Frequenz- oder Steuersignal handelt. Das Modul* ***Scale*** *legt an seinem Ausgang ein Zeitsignal mit der Bezeichnung* ***TIME_DOMAIN*** *an.* ***VecOpScal*** *erwartet an seinem Eingang einen Skalar dessen Interpretation mit* ***SCALAR*** *benannt ist. Wenn die Typbezeichner nicht übereinstimmen, so gibt das überprüfende Modul eine Warnung aus. Wenn der Benutzer entscheidet, daß die Interpretationen der Signale für seine Zwecke kompatibel sind, so bestätigt er die Frage des Systems mit* ***Ja*** *und der Kommunikationskanal wird aufgebaut. Weitere Details über Datentypen, Interpretationen sind im Kapitel 4 nachzulesen.*

3.1.3 Das Modul Formula

Als nächstes Modul aus der Gruppe **Math** wird im Beispiel **demo_06** (siehe Abb. 3.10) der Formelinterpreter **Formula** vorgestellt. Außer dem Formelinterpreter werden erneut die Module **FuncGen**, **Scale** und **AnalogChart** benutzt. Für diese Module gelten die Einstellungen analog zu Beispiel **demo_5**.

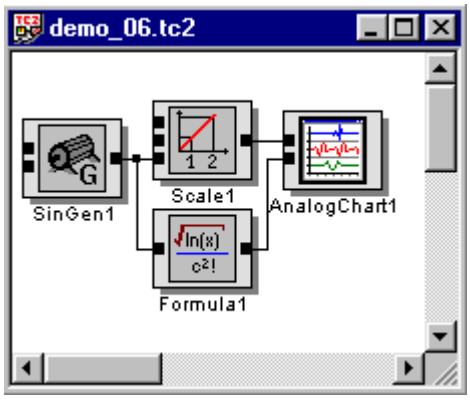


*Abb. 3.10: Signalgraph von **demo_06***

Der Dialog des Moduls **Formula** (siehe Abb. 3.11) besitzt ein Edit-Feld zur Eingabe der Formeln. Eine Formel besitzt folgende Struktur:

Modulausgang = f(Moduleingang a, Moduleingang n,...)

In einer Formel wird die Variable links des Gleichheitszeichens als Modulausgang interpretiert. Die sich rechts davon befindlichen Bezeichner werden als Eingänge verstanden. Im vorliegenden Beispiel ist die Formel

O1 = -I1;

eingetragen. Das am Eingang liegende Sinussignal wird invertiert.

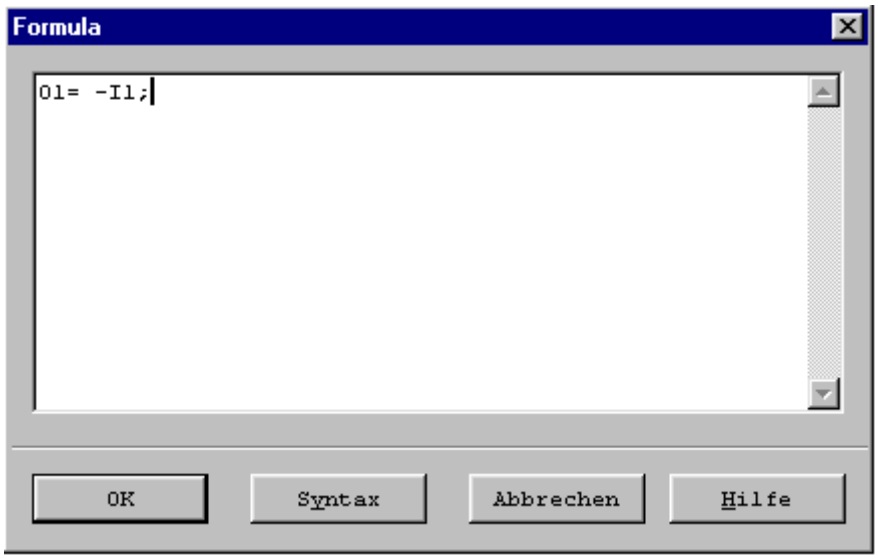


*Abb. 3.11: Dialog des Moduls **Formula***

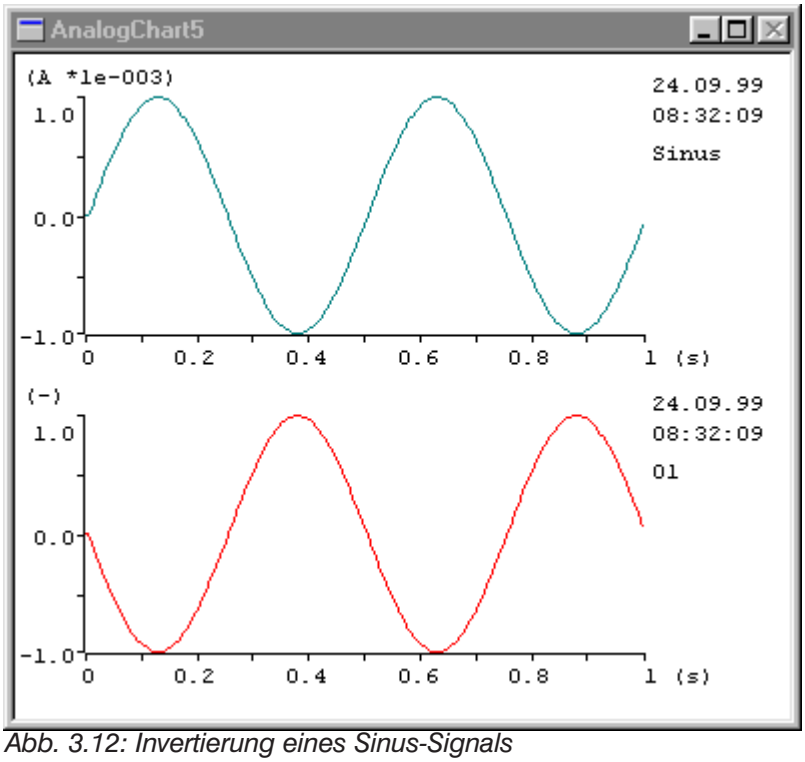


Abb. 3.12: Invertierung eines Sinus-Signals

Für die einzelnen Variablen in der Formel (= Ein/-Ausgänge des Moduls) müssen die Datentypen angegeben werden, außer es handelt sich um eine Variable, die den Defaulttyp besitzt (siehe Tab. 3.1). Der Variablentyp legt den Typ für den Ausgang fest. Ferner entscheidet er auch darüber, ob das Modul Formula die Berechnung durchführt und neue Daten auf die Ausgänge legt oder nicht. Diese Aktionen werden ausgeführt, wenn Daten an den Eingängen liegen, die einen Datentyp mit Triggereigenschaften besitzen. Liegen Daten an den Eingängen, die die Berechnung nicht auslösen, so werden die Daten in das Modul eingelesen, es erfolgt jedoch keine Berechnung.

Der Typbezeichner setzt sich aus den Buchstaben **d** (Datentyp = Ganze Zahl) oder **f** (Datentyp = Gleitkommazahl) und einem Präfix zusammen. Im Präfix ist die Charakteristik (Skalar oder Vektor) und die Triggereigenschaft festgelegt.

Charakteristik	Skalar				Vektor			
Präfix	'&'		'%'		'#'			
Typbezeichner	'&d'	'&f'	'%d'	'%f'	'#d'	'#f'	'd'	'f'
Datentyp	SWORD	DOUBLE	SWORD	DOUBLE	SWORD[1]	DOUBLE[1]	SWORD[]	DOUBLE[]
Trigger	JA	JA	NEIN	NEIN	NEIN	NEIN	JA	JA
								Default

*Tab. 3.1: Datentypen im Modul **Formula***

Die Typbezeichner werden in eckigen Klammern dem Variablennamen nachgestellt.

Beispiele:

Aufgabenstellung: Es soll ein Eingang vom Typ SWORD[1] (= Vektor aus ganzen Zahlen der Länge 1) festgelegt werden.
Lösung: ... = I1[#d]

Aufgabenstellung: Es soll ein Eingang vom Typ DOUBLE (= Skalar vom Typ Gleitkommazahl ohne Triggereigenschaft) definiert werden.
Lösung: ... = I2[%f]

In der Formel aus Abb. 3.11 wurde keine explizite Typdefinition für die Variablen vergeben, weil der Defaulttyp gewählt wurde. In dem Beispiel **demo_07** wurde die Formel zu

O1 = -(pow(I1,I2[%d]));

erweitert. Hiermit wird ein zweiter Eingang geschaffen. Als Typ wird ein Skalar erwartet, der als Datentyp eine ganze Zahl besitzt. Mit **pow(Basis, Exponent)** wird die Potenzfunktion verwendet. **Formula** enthält eine umfangreiche Funktionsbibliothek, deren Notation an die Syntax der Programmiersprache C angelehnt ist. Die komplette Funktionsreferenz kann in der Online-Hilfe nachgelesen werden.

Die Daten für den Eingang **I2** werden mit dem Modul **Spin** aus dem Menü **Module > User Input** erzeugt. Das Bedienelement dieses Moduls enthält ein sogenanntes **Spin-Control**. Mit Links-Klicks auf die Pfeile erhöhen bzw. erniedrigen Sie den Wert des Ausgangs um eins. In seinem Dialog (siehe Abb. 3.13) können Sie Signaltyp, sowie die Grenzen des Spinzählers einstellen. Außer dem Signalnamen, der im Beispiel als ***Exponent*** bezeichnet wird, werden die initialen Werte nicht verändert.

*Abb. 3.13: Dialog des Moduls **Spin***

Starten Sie das Programm. Variieren Sie mit **Spin** den Exponenten der Potenzfunktion und beobachten Sie die Auswirkungen auf die Signalverläufe.

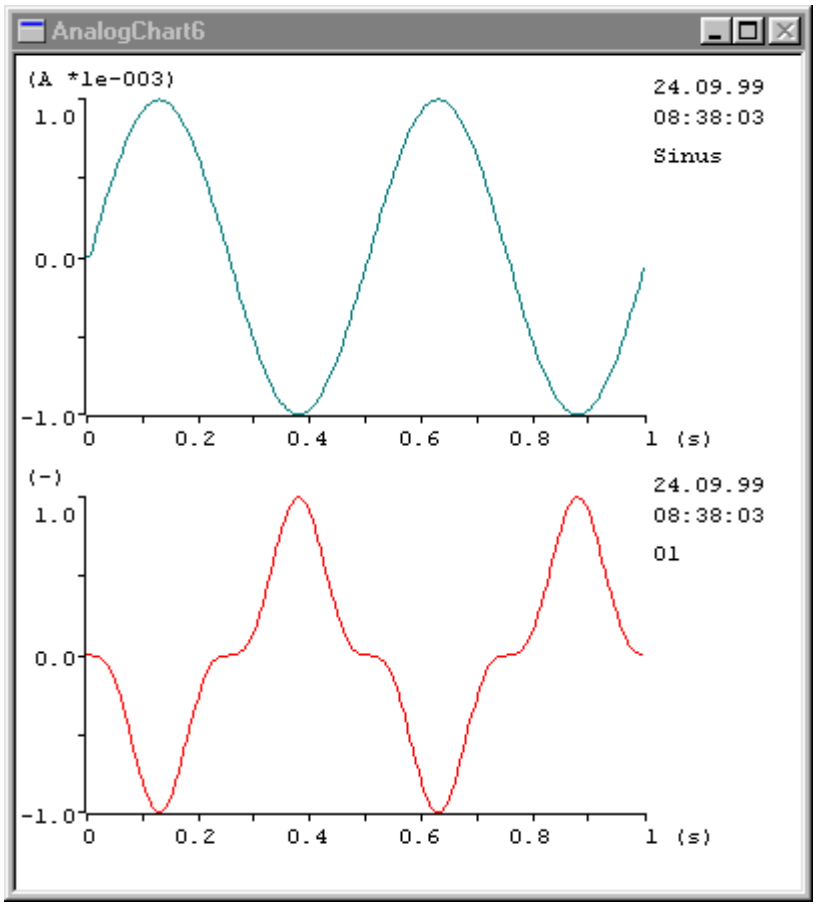


Abb. 3.14: Signalverlauf des Programms ***demo_07***

Hinweis:
Zusätzlich kann für den Ausgang neben dem Typ auch das gesamte ***Type-Info*** *angegeben werden. Die Sequenzen des Moduls Formula können beliebig kompliziert werden. Formula berechnet die Einheiten und Ranges[1] nicht automatisch.*

Das Programm ***demo_08*** *verwendet folgendes erweitertes* ***Type-Info****.*

O1[f,name = "P", min = -1, max =1, unit = "V", scale = 0.001] = pow(I1,I2[%d])* -1;

P *ist der Name für das Signal O1. Der Wertebereich des Signals ist von min = -1 bis max =1 definiert. Eine Einheit* ***V*** *(Volt) ist physikalisch nicht sinnvoll, aber zu Demonstrationszwecken geeignet. Der Skalierungsfaktor* ***scale*** *beträgt 0.001. Anzumerken ist ferner, daß bei Angabe des* ***Type-Info*** *auch der Typ des Ausgangssignal angegeben werden muß.*

Innerhalb eines ***Formula***-Moduls *können mehre Ausgänge, also mehrere Formeln, definiert werden.*

Beachten Sie, daß hier die Exponentenschreibweise (z.B. 1E-2 für 0,01) nicht möglich ist.

Im nächsten Abschnitt wird näher auf die Module der Gruppe **Display** eingegangen. Aus diesem Bereich wurden bereits die Module **DigitalDisp** und **AnalogChart** benutzt. Ferner wird erläutert, wie eine Benutzeroberfläche gestaltet werden kann.

1) Wertebereich des Signals

3.2 Module der Gruppe Display

In diesem Abschnitt lernen Sie Visualisierungsmöglichkeiten von ICONNECT kennen. Hier erfahren Sie, wie mittels der Module **DisplayManager** und **InputManager** eine Benutzeroberfläche aufgebaut werden kann.

3.2.1 Das Modul AnalogDisp

Zunächst soll mit dem Modul **AnalogDisp** eine weitere Möglichkeit aufgezeigt werden, Einzelwerte darzustellen. Das Modul kann entweder als Zeigerinstrument oder Balkenanzeige konfiguriert werden. Für den Anzeigebereich können Sie

- feste Grenzen, oder
- die Limits des **Type-Info**

festlegen.

Für beide Varianten ist eine Darstellung in Prozent möglich.

*Abb. 3.15: Dialog des Moduls **AnalogDisp***

Mit dem Parameter **Display**, des Moduls AnalogDisp (**Module > Display > Scalar**), legen Sie fest ob das Modul ein eigenes Anzeigefenster besitzt oder in den Displaymanager integriert wird. Wählen Sie für ein Visualisierungsmodul **Display > extern**, erhält das Modulicon im Signalgraph einen Ausgang, der am Modul **DisplayManager** angeschlossen werden kann.

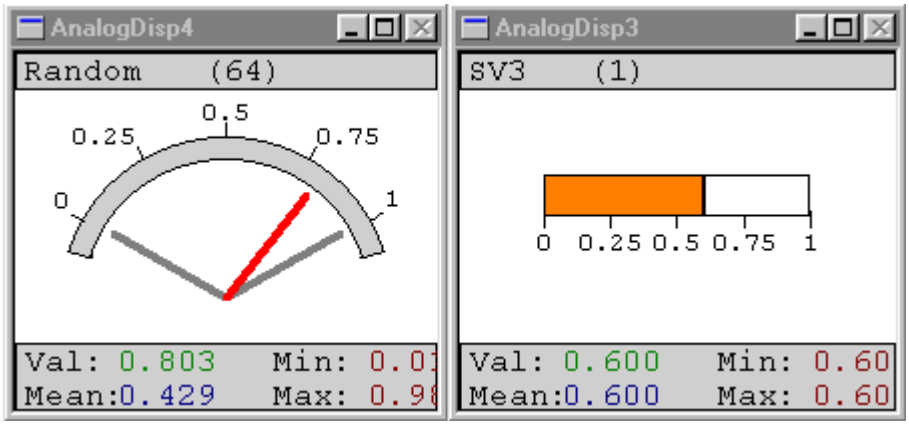
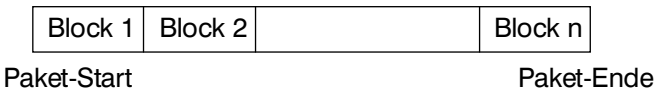


*Abb. 3.16: Das Modul **AnalogDisp** als Zeigerinstrument bzw. als Balkenanzeige*

Das Modul **AnalogDisp** erkennt das am Eingang liegende Datenformat. Es unterscheidet in seiner Arbeitsweise zwischen einzelnem Datum oder Datenblock mit mehreren Werten.



- **Val** Letzter Wert eines Blocks
- **Mean** Mittelwert eines Pakets
- **Min** Minimum eines Pakets
- **Max** Maximum eines Pakets

Arbeitet das Modul **AnalogDisp** als Zeigerinstrument, werden die Werte **Min** und **Max** in Form von zwei Schleppzeigern visualisiert.

Hinweis:
*Die Schleppzeiger beziehen sich nicht auf den gesamten Meßverlauf. Die in Abb. 3.16 eingeblendete Information **Random (64)** bzw. **P1 (1)** gibt den Typ der Datenquelle und in Klammern die Gesamtheit der Datenwerte (=Vielfaches der Blockgröße) an.*

3.2.2 Das Modul Plot

Das Modul **Plot** (**Module > Display > Array**) unterscheidet, abhängig von der Verdrahtung des X-Eingangs, zwei Arbeitsweisen:

- X-Eingang nicht verdrahtet — Y-Daten werden über die Zeit aufgetragen
- X-Eingang verdrahtet — Y-Daten werden über X aufgetragen

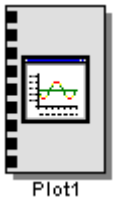


*Abb. 3.17: Das Modul **Plot***

Um Erfahrungen mit dem Modul **Plot** zu sammeln starten Sie das Programm **demo_11**. Das Modul **Plot** visualisiert das vom Modul **Random** gebildete Rauschen. Dieses Programm bildet die Basis für weitere Ergänzungen, die Sie in diesem Kapitel noch anfügen. Das Rauschen wird in unterschiedlichen Farben dargestellt. Sie haben damit die Möglichkeit **Warn**- bzw. **Alarmgrenzen** zu unterscheiden.

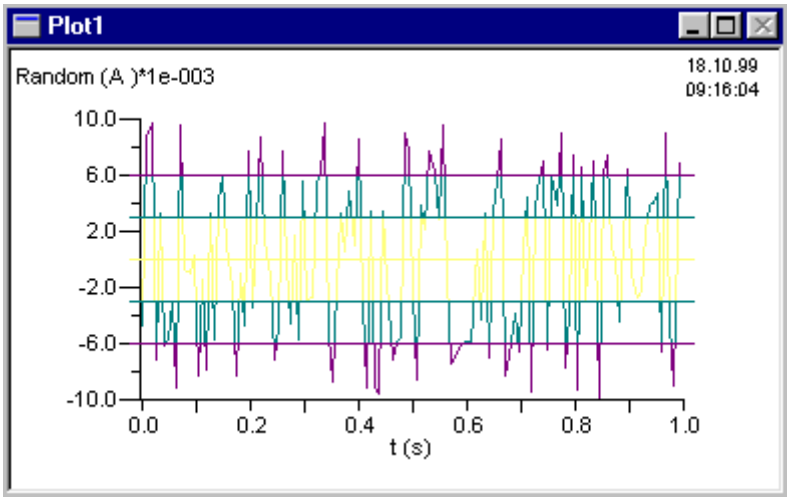


*Abb. 3.18: Anzeigefenster des Moduls **Plot***

Stoppen Sie das Programm **demo_11** und öffnen Sie den Dialog des Moduls **Plot** (siehe Abb. 3.19). Als Skalierung für die **Y-Achse** wurde **Type-Info** gewählt. Damit entsprechen Signalname, minimaler und maximaler Wert, sowie Einheit und Skalierungsfaktor den Informationen des Datenpakets, das am Y-Eingang anliegt. Für die X-Achse wurde die Achsenbeschriftung **auto** gewählt. Plot berechnet aus der Abtastrate die Zeitskale, über welcher die Y-Daten angezeigt werden. Im Auto-Range-Betrieb wird weder ein Signalname, noch eine Einheit generiert. Deshalb wurde für **Label *t*** und als **Einheit *s*** eingetragen.

Vergeben Sie für **Titel** bzw. **Untertitel** die Überschrift ***Weißes Rauschen***. Erhöhen Sie mit **Präzision** die Anzahl der Nachkommastellen der Achsenbeschriftung. Starten Sie zwischendurch das Programm um die Änderungen zu verfolgen.

Sie können **Warn**- und **Alarmgrenzen** definieren und so das Modul Plot als Visualisierungstool im Bereich der Qualitätskontrolle bzw. der Überwachung einsetzen. Ein Überschreiten der Grenzen wird mit einem Farbumschlag verdeutlicht. Im Beispiel sind die Grenzen auf -3, 3, -6 und 6 eingestellt.

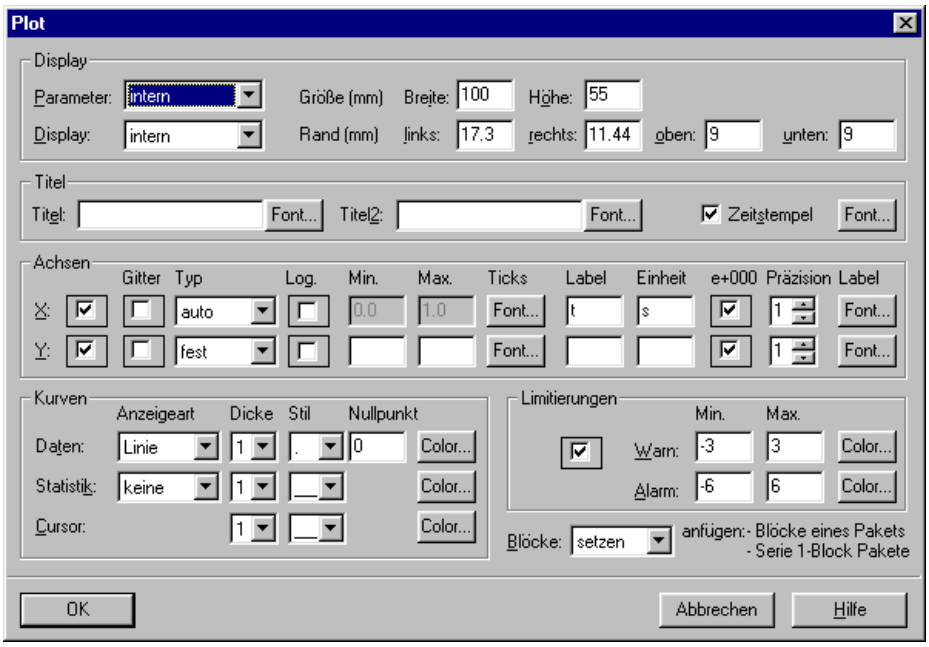


*Abb. 3.19: Dialog des Moduls **Plot***

Wie bereits in Kap. 2.1.4 angesprochen, können Sie Parameter zur Applikationslaufzeit modifizieren, wenn die Parameterquelle auf **extern** eingestellt ist. Dazu führen Sie folgende Schritte aus:

1. Wählen Sie im Modul **Plot** für die Parameterquelle **extern**

2. Holen Sie das Modul **ParamConv** aus dem Menü **Module > SignalProcessing > Convert** in die Arbeitsfläche. Verbinden Sie den Ausgang von **ParamConv** mit dem Eingang **EXT** des Moduls **Plot**.

Das Modul **ParamConv** arbeitet als Parallel-Seriell-Wandler. **Plot** erwartet die Parameter am Eingang in serieller Form.

Im Dialogfenster von **ParamConv** sind alle Signale angezeigt, die das Modul **Plot** erwartet. Für jedes Signal können Sie einen festen Wert oder einen Kontrolleingang definieren.

3. Öffnen Sie das Dialogfenster von **ParamConv**, wählen Sie die Zeile **UBYTE[]{Title}** aus und tragen Sie als Defaultwert ***Weißes Rauschen*** ein (siehe Abb. 3.20).

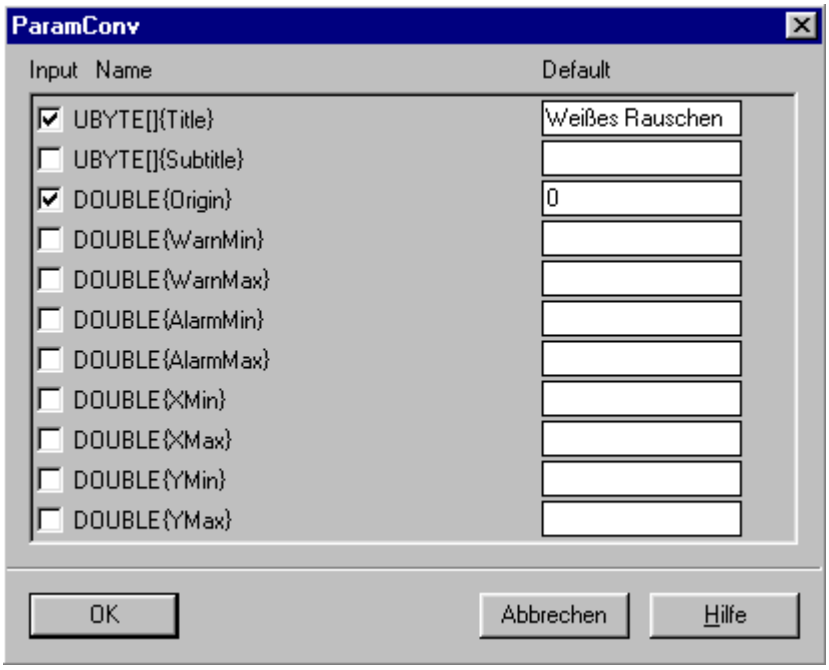


*Abb. 3.20: Dialog des Moduls **ParamConv***

4. Wählen Sie die Zeile **DOUBLE{Origin}** aus und tragen Sie als Defaultwert den Wert ***0*** ein. Quittieren Sie die Eingabe mit **OK**.

Mit den Schritten 3 (**Titel**) und 4 (**Nullpunkt**) haben Sie feste Werte für das Modul **Plot** vergeben. ICONNECT vermerkt dies im Dialog des Moduls **ParamConv**. In der Spalte **Kontrolle** finden Sie den Eintrag **Value**.

Die Warn- und Alarmgrenzen sollen während der Laufzeit variierbar sein. Das Modul **ParamConv** ist deshalb mit Kontrolleingängen auszustatten.

5. Öffnen Sie das Dialogfenster von **ParamConv.** Wählen Sie die Einträge **DOUBLE{WarnMin}, DOUBLE{WarnMax}, DOUBLE{AlarmMin}** und **DOUBLE{AlarmMax}** aus.

An die Eingänge **WarnMax** und **AlarmMax** werden zwei Sliders direkt angeschlossen. Die Eingänge für die negativen Werte (**WarnMax** und **AlarmMax** werden zur Umkehrung der Polarität über das Modul Formula geführt.

6. Vervollständigen Sie den Signalgraphen (siehe Abb. 3.21). Geben Sie für die Module **SLDERV** als Signaltyp **DOUBLE** und als Input **intern** an. Editieren Sie die Befehlszeile für die beiden Module **Formula** mit

O1[%f]=-I1[&f];

Nach dem Starten des Signalgraphen können Sie mit den vier SliderV die Warn- und Alarmgrenzen zur Applikationslaufzeit einstellen.

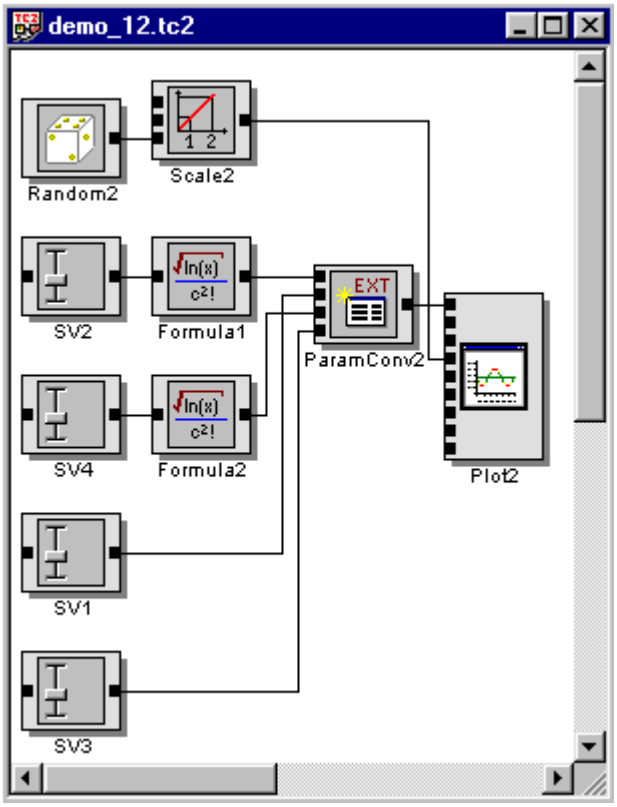


*Abb. 3.21: Signalgraph von **demo_12***

Außer den Signalen im Zeitbereich können in ICONNECT auch binäre Signale dargestellt werden. Für die Visualisierung der binären Signale stehen in ICONNECT zwei Module zur Verfügung. Der Funktionsumfang von **DigitalChart** entspricht im wesentlichen dem von **AnalogChart** und wird deswegen hier nicht weiter erläutert. **BinaryDisp** wird mit der Gruppe **Logic** besprochen.

3.2.3 Die Module InputManager und DisplayManager

Mit den Modulen **InputManager** und **DisplayManager** fassen Sie Bedienelemente bzw. Ausgabeelemente zusammen. Der Signalgraph von **demo_13** (siehe Abb. 3.22) verwendet ein einfaches Benutzerinterface. Wie bereits vorher angesprochen, können Sie Visualisierungsmodule so einstellen, daß sie mit dem Displaymanager verbunden werden können. Analog dazu können Sie Module mit Bedienelementen an den Inputmanager anschließen.

Die Anzahl der Ausgänge für den Inputmanager können Sie unter **Anzahl der Anzeigen** definieren (siehe Dialogfenster **InputManager** in Abb. 3.23). Sobald ein Ausgang von Modul **InputManager** mit dem Modul **SliderV** verbunden ist, wird das Bedienelement in den Inputmanager aufgenommen (siehe Abb. 3.24).

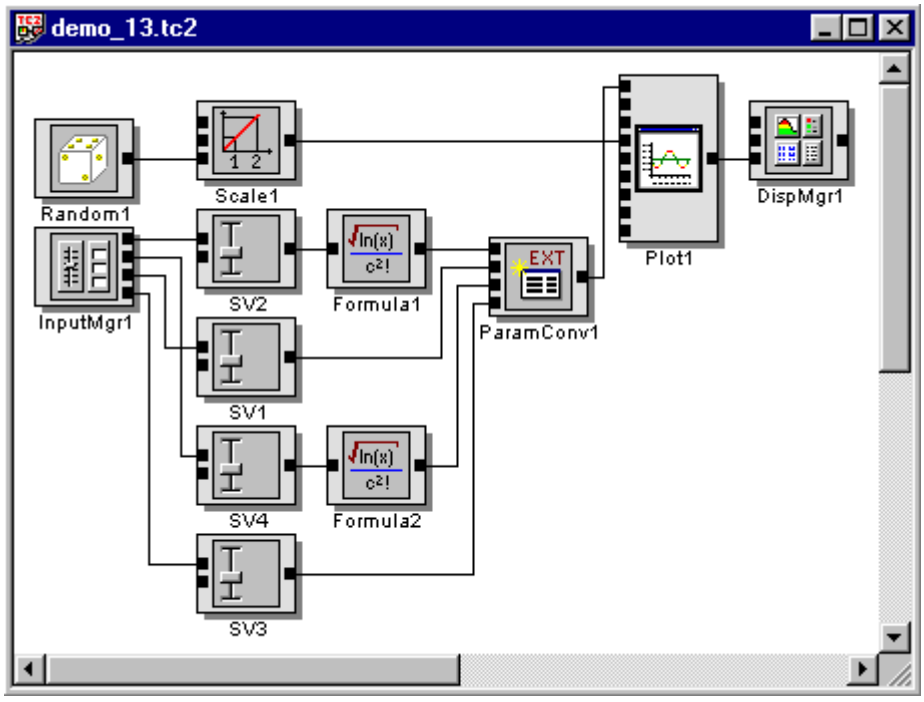


*Abb. 3.22: Signalgraph von **demo_13***

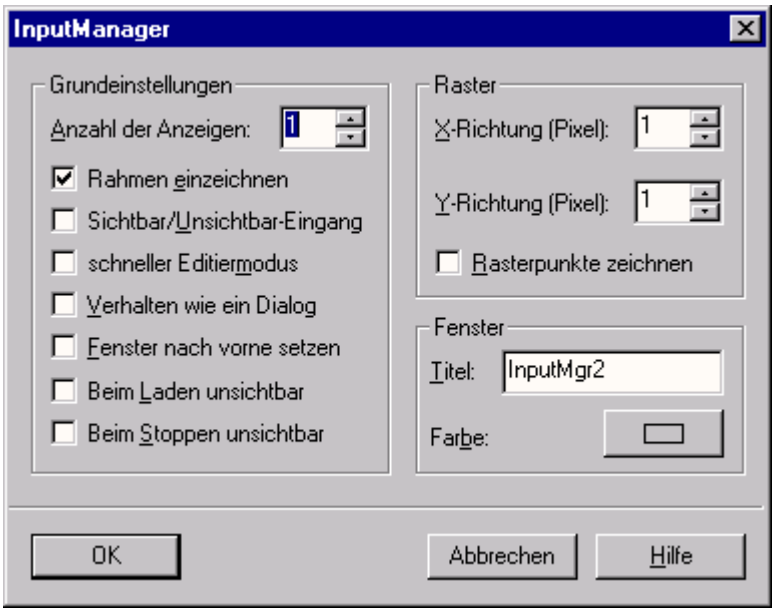


*Abb. 3.23: Dialog des Moduls **InputManager***

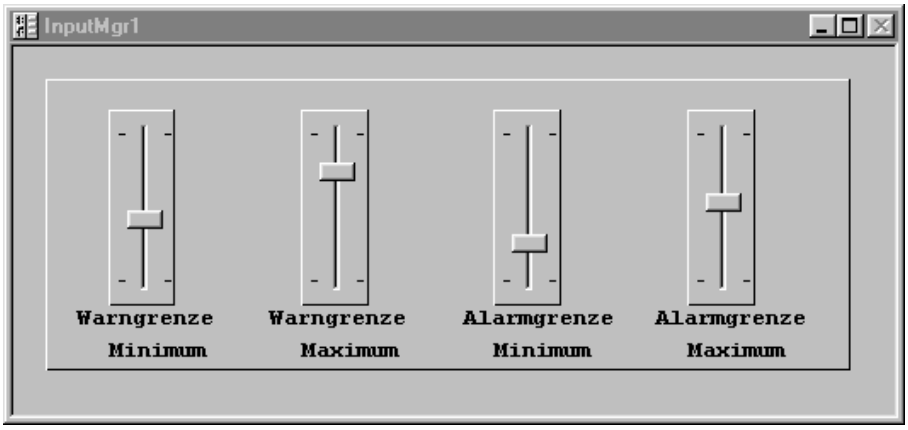


*Abb. 3.24: Bedienoberfläche des Moduls **InputManager***

Die Größe und Position eines Bedienelements können Sie mit der Maus einstellen. Dazu sind folgende Schritte notwendig:

1. Doppel-Links-Klick im Fenster des Inputmanagers.

Das weiß hinterlegte Fenster zeigt Ihnen an, daß Sie sich im Editiermodus befinden. Ein geeignetes Raster (siehe unter **Raster** im Dialog von **InputManager**) erleichtert das Gestalten des Eingabefensters.

2. Bewegen Sie den Mauszeiger an den Rand des Bedienelements (siehe Abb. 3.25a) und variieren Sie mit gedrückter linker Maustaste die Größe. Wenn Sie die Position eines Bedienelements verändern möchten, bewegen Sie den Mauszeiger in das Bedienelement (siehe Abb. 3.25b) und schieben mit gedrückter linker Maustaste das Bedienelement an die gewünschte Position.

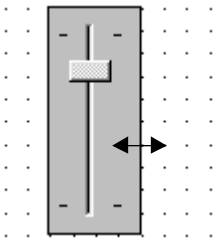


Abb. 3.25a: Dehnen

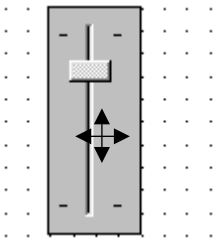


Abb. 3.25b: Schieben

Mit einem einfachen Rechts-Klick öffnen Sie ein PopUp-Menü, das Ihnen die Möglichkeit bietet, einen Text, eine Linie oder ein Rechteck einzufügen. Ein bereits gezeichnetes Objekt können Sie über das Kontext-Menü löschen. Der Editiermodus wird durch Doppel-Linksklick beendet.

ICONNECT bietet Ihnen die Möglichkeit das Fenster für den Inputmanager wahlweise auszublenden. Damit können Sie in Verbindung mit logischen Modulen und Steuersignalen (siehe Kap. 3.3) beliebige Dialogfolgen für die Benutzeroberfläche definieren.

Das Prinzip des Displaymanagers ist äquivalent zum Inputmanager. Auch die Vorgehensweise beim Gestalten von Oberflächen ist die gleiche. Der Displaymanager besitzt jedoch zusätzlich einen Eingang, der bei seiner Aktivierung durch ein Steuersignal (siehe Kap. 3.3) den momentanen Zustand des Displaymanagers auf einen Drucker ausgibt.

Im nächsten Kapitel werden die logischen Module erläutert. In diesem Zusammenhang wird auch der Unterschied zwischen Daten- und Steuersignalen diskutiert.

3.3 Module der Gruppe Logic

Anhand einiger Beispiele lernen Sie

- Steuerungsaufgaben in ICONNECT auszuführen
- Neue Module der Gruppe **User Input** und **Display** kennen

3.3.1 Das Modul Count

Count zählt entweder
- Blöcke oder
- Pakete (siehe Abb. 3.28)

Der Eingang **I1** des Moduls ist mit beliebigen Datentypen kompatibel.

*Abb. 3.26: Bedienelement des Moduls **Button***

Starten Sie das Programm **demo_14**. Mit einem Links-Klick auf den Button **B2** (siehe Abb. 3.26) aus dem Menü **Module > User Input** wird im Signalgraph von **demo_14** der Pulsgenerator gestartet. **PulseGen** liefert Pulse mit den Werten **Null** und **Eins**. Die Signale werden mit dem Zeitverhalten generiert, die im Dialog (siehe Abb. 3.27) eingestellt werden. Die Parameter sind so gewählt, daß **PulseGen** alle zwei Sekunden eine positive Flanke generiert. Das Signal wird an den Zähleingang I1 des Moduls **Count** weiter geleitet. Der Zählerstand des Moduls **Count** erhöht sich also mit jeder Flanke des Pulsgenerators und gibt den Zählerstand am entsprechenden Ausgang aus. Das Modul **DigitalDisp** visualisiert den Zählerstand.

Hinweis:
Schaltsignale haben grundsätzlich Paketstatus

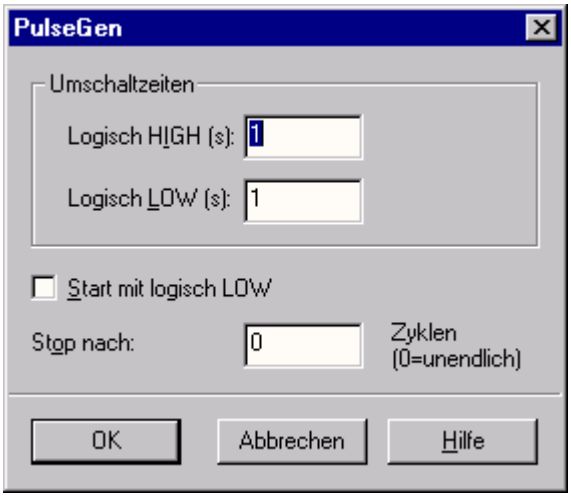


*Abb. 3.27: Dialog des Moduls **PulseGen***

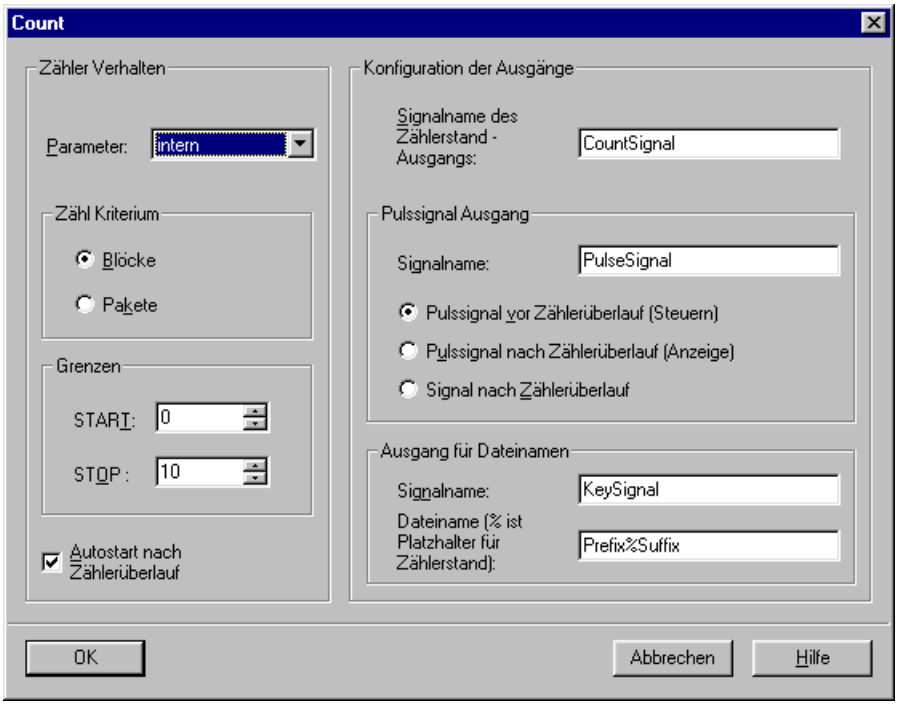


*Abb. 3.28: Dialog des Moduls **Count***

Im Dialog des Moduls **Count** (siehe Abb. 3.28) sind als Zählergrenzen die Werte 0 und 10 eingetragen. Nach einem Überlauf wird der Zähler neu gestartet. Das Zählermodul kann neben der Ausgabe des reinen Zählerstands auch Steuersignale generieren.

Im Signalgraphen des Beispiels **demo_15** werden die Steuersignale des Moduls **Count** mit dem Modul **BinaryDisp** dargestellt. **BinaryDisp** zeigt die Zustände von binären Signalen mit Leuchtdioden an. Wie beim digitalen Display können mehrere Eingänge spezifiziert werden. Für die beiden Zustände des Signals können Sie Namen vergeben.

Dateinamen verwenden Sie, um den Dateizugriff bestimmter Module zu regeln (siehe Kap. 3.4).

In den vorangehenden Ausführungen wurde vermehrt von Steuersignalen gesprochen. Beachten Sie den Unterschied zwischen Daten- und Steuersignalen:

- Steuersignale sind eindimensionale Vektoren vom Typ **SWORD[1]** und besitzen die Interpretation **BIN**. Die logischen Module, bzw. alle Module, die mit Steuersignalen kommunizieren, reagieren sofort auf eine Änderung an einem einzelnen Eingang.

- Datensignale sind in der Regel mehrdimensionale Vektoren, die mit einer bestimmten Abtastrate erzeugt worden sind. Die Module verarbeiten solche Daten erst, wenn an allen entsprechenden Eingängen Daten präsent sind. Es werden nur Zustandsänderungen übertragen.

3.3.2 Die Module UniGate

Die logischen Gatter befinden sich im Menü **Module > Logic**. Sie realisieren logische Verknüpfungen (Schaltalgebra). Bei allen Modulen der Gruppe **UniGate** können Sie sowohl die Ein- als auch die Ausgänge invertieren.

Typ	Eingänge	logische Verknüpfung		
		UND	ODER	EXOR
UniGate2	2	✓	✓	✓
UniGate3	3	✓	✓	
UniGate4	4	✓	✓	

Tab. 3.2: Mögliche Verknüpfungsarten der Modulgruppe UniGate

Abb. 3.29 zeigt die Konfiguration für eine logische UND-Verknüpfung mit zwei Eingängen, wobei der Eingang I1 invertiert ist.

Kurzschreibweise:

$$O1 = \overline{I1} \wedge I2$$

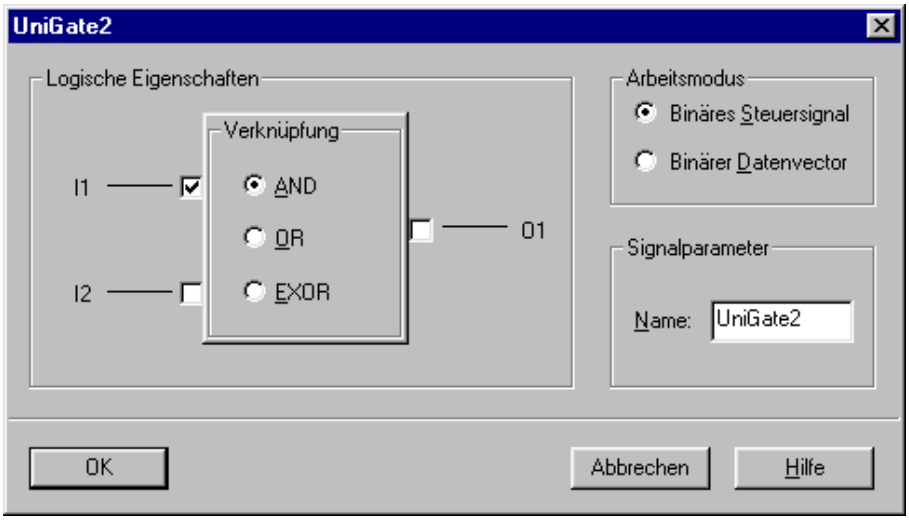


*Abb. 3.29: Dialog des Moduls **UniGate2***

Das Beispiel **demo_16** erzeugt alle möglichen Belegungen für das Modul **UniGate4**. Das Modul **BinDeCoder** wandelt eine Dezimalzahl in binäre Signale um. Für das Modul **BinDeCoder** können Sie folgende Codierungsarten vergeben:

- Dezimal ⟶ Binär
- Dezimal ⟶ BCD (8421-Code)
- Dezimal ⟶ Grey-Code

Starten Sie das Beispiel **demo_16**. Das Modul BinaryDisp stellt die Zustände der vier Eingangsleitungen und das Ergebnis der Verknüpfung dar.

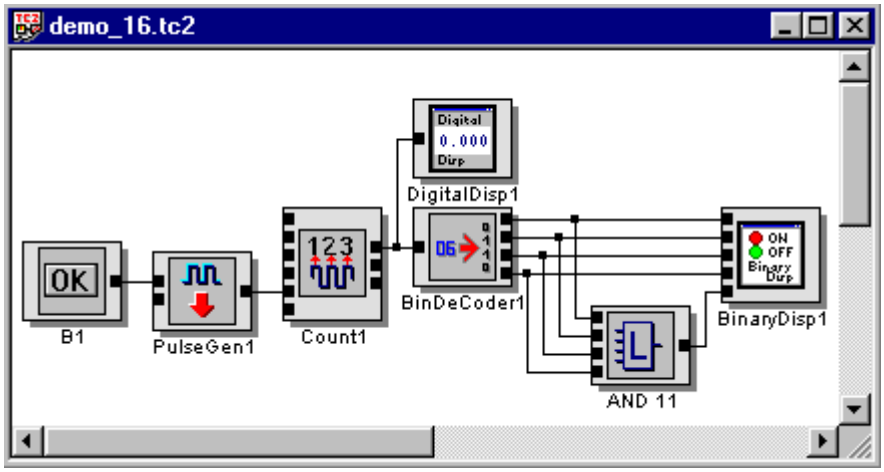


*Abb. 3.30: Signalgraph von **demo_16***

Hinweis:
*Die logischen Verknüpfungen NAND, NOR und XOR können mit den Grundfunktionen UND, ODER, EXOR und NICHT realisiert werden. Weiterführende Literatur zum Thema Schaltalgebra (Boolesche Algebra) finden Sie im Buch **Halbleiter-Schaltungstechnik** von **Tietze/Schenk**.*

3.3.3 Die Module FlipFlop, Mono-Flop und T-FlipFlop

ICONNECT FlipFlop's ahmen Schaltungen, wie sie in der Elektrotechnik Verwendung finden, nach. Sie besitzen die Fähigkeit als Speicherelement zu arbeiten. Sie sind in der Lage Zustände zeitlich unbegrenzt einzunehmen. Derzeit sind folgende FlipFlop-Typen realisiert:

- RS-FlipFlop — Set-Reset-FlipFlop, Basistyp der FlipFlop's
- Mono-Flop — Pos. oder neg. Flankentriggerung, Haltezeit variabel von 0,01s bis 10.000s, retriggerbar
- T-FlipFlop — Pos. oder neg. Flankentriggerung

Der Signalgraph in Abb. 3.31 zeigt die Ansteuerung von zwei Mono-Flops. Die FlipFlops werden mit der negativen Flanke des Moduls **PulseGen1** getriggert.

Beide Mono-Flops gehen synchron miteinander in den logischen Zustand **Eins**, fallen aber bedingt durch unterschiedliche Haltezeiten nacheinander ab (siehe Abb. 3.32). Das Modul **DigitalChart** visualisiert die Signale. Starten Sie das Beispiel **demo_36**, varieren Sie die Parameter der Module und beobachten Sie die Auswirkungen.

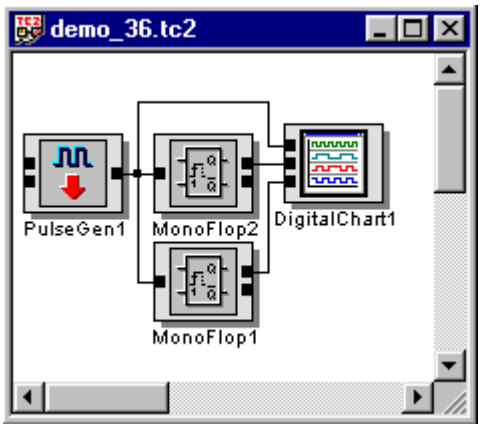


*Abb. 3.31: Signalgraph von **demo_36***

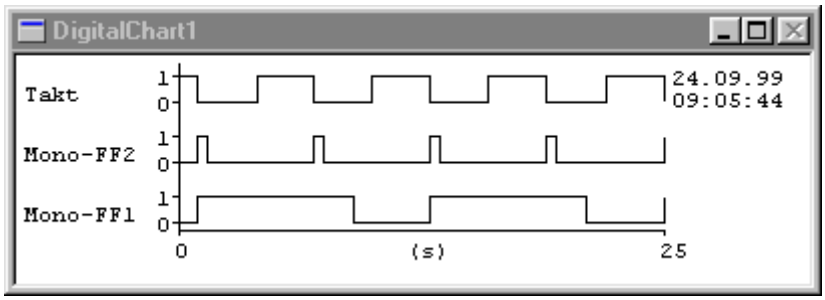


*Abb. 3.32: Signalverlauf von **demo_36***

Hinweis:
*Die Eingangsbelegung **Set[0]** und **Reset[0]** ist für das **RS-FlipFlop** nicht definiert. Das **RS-FlipFlop** liefert in diesem Fall den Ausgangszustand der letzten gültigen Eingangsbelegung.*

3.4 Speichern und Lesen von Daten

In diesem Abschnitt lernen Sie wie Sie

- Daten in Dateien ablegen bzw. aus Dateien lesen
- die unterschiedlichen Dateiformate **SaveAscii, LoadAscii** / **SaveTable**, **LoadTable** verwenden.

ICONNECT stellt zwei verschiedene Dateiformate zur Verfügung. Die Module **SaveAscii** und **LoadAscii** aus dem Menü **Module > Data IO > File** schreiben und lesen die Dateien im ASCII-Format. **SaveAscii** fügt zusätzlich Zeitinformationen ein. Somit können Sie Daten mit **LoadAscii** in Bezug auf das zeitliche Verhalten so einlesen, wie sie geschrieben wurden.

Die Module **SaveTable** und **LoadTable** schreiben und lesen Dateien in Form von Tabellen, wobei eine Spalte der Tabelle einem Eingang des Moduls entspricht. Das Format ist kompatibel zu Tabellenkalkulationen in Office Paketen, wie z.B. MS-Excel.

Da die Parametrisierung beider Typen fast äquivalent vor sich geht, wird hier nur das Schreiben und Lesen von ASCII-Dateien beschrieben.

Laden Sie das Beispiel **demo_17** und öffnen Sie das Dialogfenster von **SaveAscii** (siehe Abb. 3.33). Unter der Rubrik **Durchsuchen...** bestimmen Sie den Pfad und den Dateinamen der Datei. In der Regel wird der Dateiname während der Applikationslaufzeit bestimmt. Trotzdem ist ein initialer Dateiname zu definieren, da bei Applikationsstart eine Datei geöffnet wird, die den initialen Dateinamen besitzt.

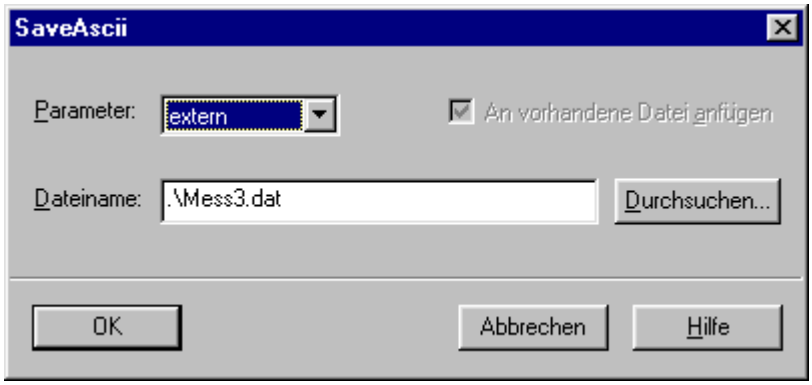


*Abb. 3.33: Dialog des Moduls **SaveAscii***

Mit dem Feld **an vorhandene Datei anfügen** entscheiden Sie, ob die Daten nach einem Neustart der Applikation an das Ende der bereits bestehenden Datei angefügt werden, oder ob die alte Datei überschrieben wird.

Sie können den Dateinamen auch zur Laufzeit über den **EXT**-Eingang (= externer Eingang) vergeben. Beispielsweise kann der Dateiname automatisch mit dem Modul **Count** erzeugt werden. In diesem Fall wird der Zählerstand mit einem Schlüsselwort verbunden. Dazu ist folgendes notwendig:

1. Öffnen Sie den Dialog des Moduls **Count**. Tragen Sie im Feld **Ausgang für Dateiname > Dateiname** den Pfad **c:\Temp\Mess%.dat** für die Ablage der Datei ein.

Die Pfadangabe setzt sich aus

- Prefix Pfad und Dateiname
- % Zählerstand
- Suffix Datei-Extension

zusammen. Im Beispiel wählen Sie für Prefix **c:\Daten\Mess** und **.dat** für das Suffix. Während der Generierung des Schlüsselworts wird an der Stelle des Prozentzeichens der Zählerstand eingetragen. Für jede Messung wird so ein eindeutiger Dateiname erzeugt.

2. Öffnen Sie das Dialogfeld des Moduls **SaveAscii** (siehe Abb. 2.34). Geben Sie den Pfad **c:\Daten\Mess** vor.

Starten Sie das Programm und kontrollieren Sie die angelegten Dateien.

Hinweis:
Pfade werden nicht von ICONNECT angelegt. Wählen Sie gegebenenfalls einen existierenden Pfad bzw. legen Sie einen an.

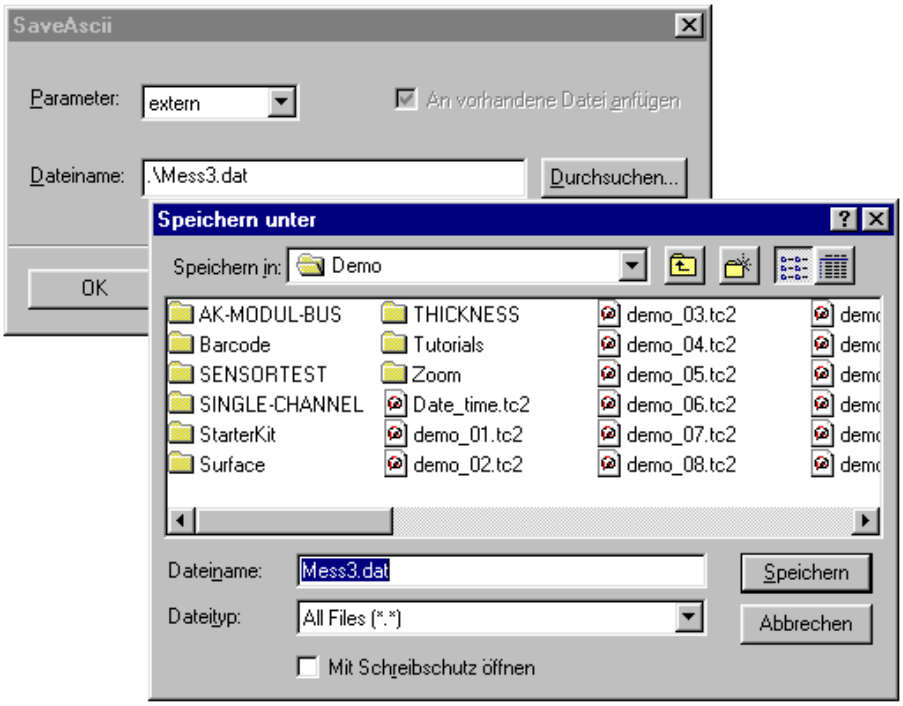


*Abb. 3.34: Dialoge des Moduls **SaveAscii***

Das Modul **FileDlg** ist eine weitere Möglichkeit, während der Laufzeit einen Dateinamen zu erzeugen. Dieses Modul ruft den File-Dialog von Windows auf und wird vor allem zum Lesen von Dateien eingesetzt.

Im Beispiel **demo_17** haben Sie Dateien angelegt. ICONNECT bietet Ihnen die Möglichkeit, gespeicherte Meßwerte zu einem späterem Zeitpunkt anzusehen. Diese Dateien sollen jetzt ausgelesen werden. Führen Sie dazu folgende Schritte durch:

1. Laden Sie das Beispiel **demo_19**

2. Öffnen Sie im Signalgraphen die Dialoge der Module **FileName** und **LoadAscii1**. Richten Sie den jeweils von Ihnen vergebenen Pfad zu den Meßdateien aus **demo_17** ein.

3. Starten Sie das Programm. Drücken Sie den Button **Enter** (siehe Abb. 3.35) im Bedienelement des Moduls **FileDg1**. Sie starten damit den Datei-Dialog von Windows zur Auswahl der gewünschten Meßdatei.

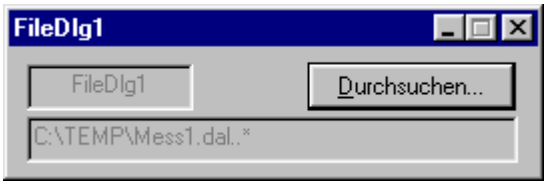


*Abb. 3.35: Bedienelement des Moduls **FileDlg***

Hinweis:
*Mehr zum Thema **Speichern und Lesen Von Daten** finden Sie im Anhang (siehe Kap. 9.2).*

3.5 Module der Gruppe Signalprocessing

In diesem Abschnitt lernen Sie Möglichkeiten

- zur Grenzwertüberwachung
- zum Auschneiden von Daten
- zur Erzeugung von Triggerereignissen
- zum Filtern und Glätten von Daten

kennen. Den Abschluß des Abschnitts bilden einige mathematisch komplexere Module wie FFT (Fourier-Transformation) und CFFT (Komplexe Fourier-Transformation).

Das Modul Limits

Das Grenzwertmodul Limits kann unterschiedliche Aufgabenstellungen lösen.

Grenzwertüberwachung

In Beispiel **demo_20** wird ein Signal auf die Überschreitung von Grenzwerten untersucht. Das Dialogfeld von **Limits** (siehe Abb. 3.36) erlaubt die Definition von zwei Grenzwerten. Zur Grenzwertüberwachung werden die beiden Ausgänge **S>** und **S<** verdrahtet. Diese liefern ein binäres Signal des Typs **SWORD[1]**, wenn der im Dialog angegebene Bereich nach oben **S>** oder nach unten **S<** verlassen wird.

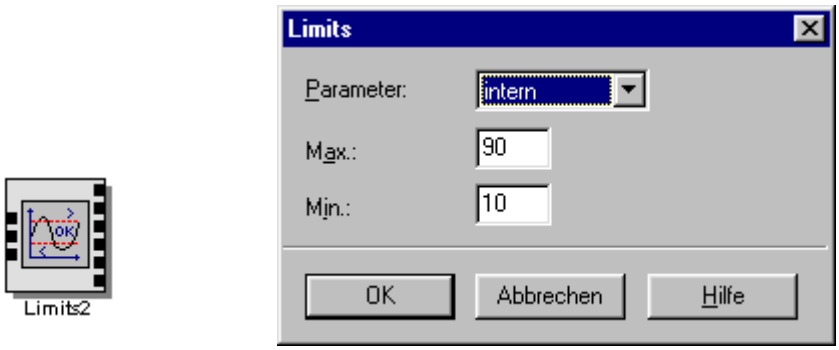


*Abb. 3.36: Icon und Dialog des Moduls **Limits***

Mit den Modulen **FuncGen** und **Scale** wird ein Wegsignal mit einem Meßbereich von 100 μm erzeugt. Im Grenzwertmodul ist ein Bereich von 10 μm bis 90 μm definiert. Werden diese Grenzwerte unter- bzw. überschritten, so wird dies mit zwei **BinaryDisp**-Modulen angezeigt. Die gesamte Visualisierung wird im Displaymanager durchgeführt.

Signalbegrenzung

Das Modul **Limits** kann Signale begrenzen. **Limits** hält beim Verlassen des erlaubten Bereichs solange den Grenzwert, bis sich das Orignalsignal wieder innerhalb der Grenzwerte befindet (siehe Abb. 3.37). In diesem Fall ist der Ausgang **LIM** (siehe Beispiel **demo_21**) zu verdrahten.

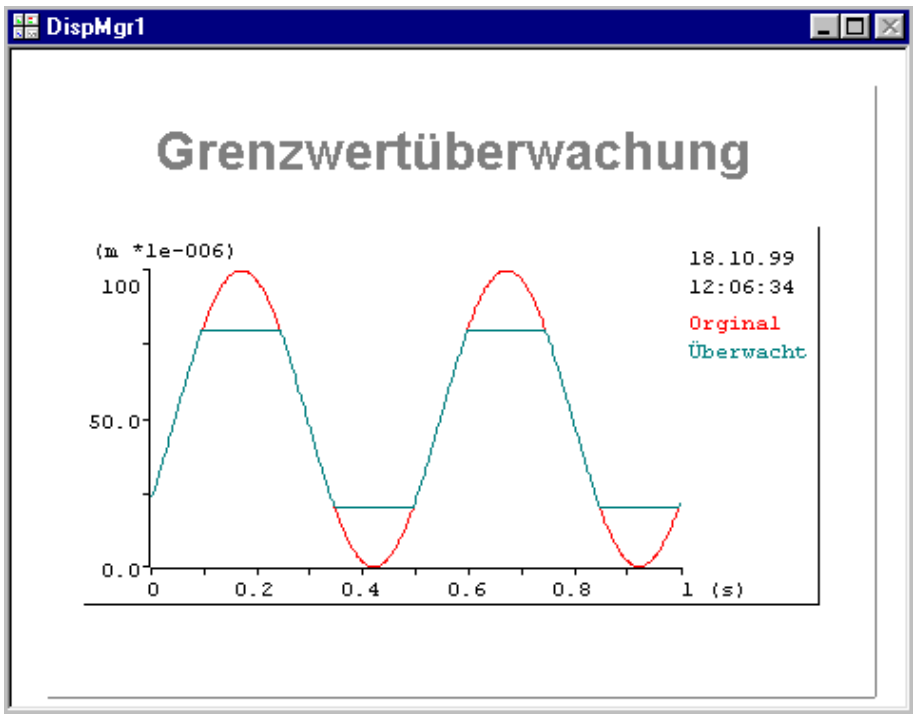


*Abb. 3.37: Signalbegrenzung mit **Limits***

Ausschneiden von Signalteilen

Mit dem Modul **Edges** können Sie zusammen mit **Limits** Teile von Signalen ausschneiden. Die Signale müssen dazu in Paketform vorliegen.

Limits erzeugt an den Ausgängen **>, OK** und **<** binäre Vektoren. Diese Vektoren charakterisieren die Korrelation zwischen den Grenzwerten und den Eingangsdaten. Im Beispiel **demo_22** ist der Ausgang **>** von **Limits** mit dem Eingang **BIN** von **Edges** verbunden. **Edges** manipuliert dann das Orginalsignal, das am Eingang **I1** liegt, mit Hilfe des binären Vektors. Liegt das Datum an der Indexposition x des Eingangsvektors im zugelassenen Bereich, so wird an dieser Stelle der Wert 0 in den Ausgangsvektor geschrieben. Liegt der Wert oberhalb des angegebenen Grenzwerts, so wird eine 1 eingetragen.

Für den auszuschneidenden Teil der Datenmenge ist der Bereich entsprechend im Dialog von **Limits** zu definieren. Das erzeugte binäre Signal wird an das Modul **Edges** weitergegeben.

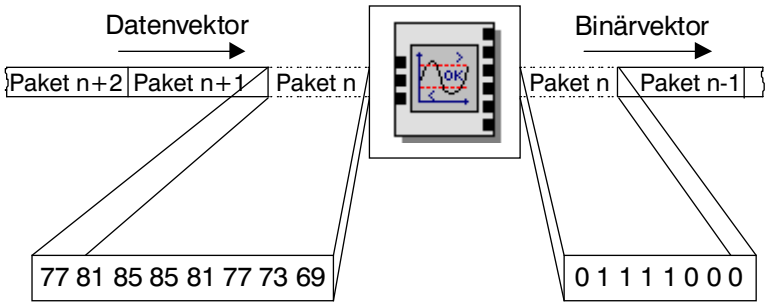


*Abb. 3.38: Generierung eines Binärvektors durch das Modul **Limits***

In **demo_22** ist eine Kantenerkennung implementiert. Das Programm simuliert Eingangsdaten im Bereich von 0 μm bis 100 μm. Alle Daten des Orginalsignals, die zwischen dem ersten Eintreten und dem letzten Austreten des Orginalsignals aus dem Bereich über 80 μm liegen, werden ausgeschnitten.

Für manche Aufgabenstellungen ist es notwendig, die Ränder eines ausgeschnittenen Bereichs zu entfernen, da Signalunregelmäßigkeiten ausgeblendet werden sollen. Dazu können Sie im Dialog von **Edges** eine Start- und Stopgrenze in Prozent angeben. Im Beispiel **demo_22** werden die ersten und letzten 5 % des Signals ausgeblendet.

Eine andere Möglichkeit besteht darin, nur den Teil des Signals, der sich oberhalb des Grenzwertes befindet auszuschneiden (siehe Beispiel **demo_23**). Dazu wird die Option **Block aus 1-Signalen** im Modul **Edges** gewählt und der zweite Block eingetragen. Mit diesem Parametersatz wird die Spitze der zweiten Sinusschwingung ausgeschnitten.

Hinweis:
Alternativ können Sie Signalteile ausschneiden, wenn Sie ***Edges*** *durch* ***Packet*** *und* ***Limits*** *durch* ***Trigger*** *ersetzen.*

Trigger *erwartet jedoch die Daten nicht in Paketform. Damit können Sie aus Daten, die im Dauermodus aufgenommen werden, bestimmte Ereignisse ableiten.*

3.6 Module der Gruppe Statistics

3.6.1 Das Modul DStatistics

DStatistics berechnet deskriptive statistische Werte. Derzeit sind realisiert:

- Anzahl, Minimum, Maximum
- Mittelwert, Standardabweichung
- Skewness, Kurtosis

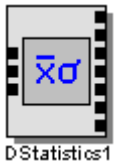


*Abb. 3.39: Das Modul **DStatistics***

Es stehen folgende Eingänge zur Verfügung:

- **I1** Dateneingang
- **Cal** Calculate
- **Res** Reset

Die statistischen Ausgangsgrößen werden berechnet und ausgegeben sobald

- ein **Eins**-Signal am **Cal**-Eingang anliegt oder
- ein **High**-Pegel am **Res**-Eingang anliegt oder
- ein Paketende erreicht wird

Die Statistikergebnisse werden solange kumuliert bis

- der **Res**-Eingang zurückgesetzt, oder
- die Messung gestoppt wird

Hinweis:
High am Reset-Eingang führt nicht zu einem Reset eines mit DStatistics verbundenen Displays.

Im Beispiel **demo_24** wird über das Modul **Count** bei jedem zweiten Block ein Pulssignal erzeugt, das die Berechnung der deskriptiven statistischen Größen bewirkt. Den Button **Reset** verwenden Sie, um den Puffer des Moduls **DStatistics** zurückzusetzen.

3.6.2 Das Modul Hist

Für die Überwachung der Güte z.B. eines Produktionsprozesses werden Häufigkeitsverteilungen als Entscheidungshilfe verwendet. Das Modul **Hist** wertet die Häufigkeiten für die einzelnen Ereignisse aus.

Abb. 3.40: Das Modul **Hist**

Hist verfügt über folgende Parameter:

- Start, Ende — Darzustellender Wertebereich auf der Abszisse
- Klassen — Anzahl der möglichen diskreten Ereignisse auf der Abszisse
- Skalierung — Einteilung der absoluten Häufigkeit (Summe, Maximum oder keine) auf der Ordinate

Das Beispiel **demo_25** wertet die Zahlenfolgen eines Zufallszahlen-generators aus. Das Modul **Plot** visualisiert die Ergebnisse von **Hist**.

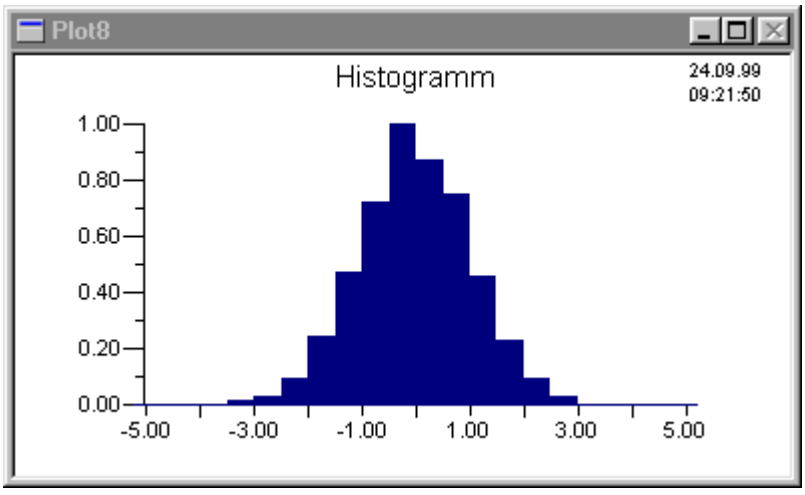


Abb. 3.41: Histogramm eines Zufallszahlengenerators

3.6.3 Das Modul Sorter

Das Modul **Sorter** verwenden Sie, um Steuerungsaufgaben zu lösen. Sie können im Dialog

- Parameterquelle
- Minimum und Maximum für den Wertebereich
- die Anzahl der Sortierklassen
- die Klassengrenzen
- die Art des Ausgangssignals

festlegen.

Sorter berechnet anhand des Wertebereichs und der Anzahl der Sortierklassen die Klassengrenzen. Um die Klassengrenzen manuell zu wählen, klicken Sie zweimal hintereinander auf die zu ändernde Klassengrenze und editieren diese.

Als Ausgangsgröße wählen Sie entweder einen Vektor vom Typ **DOUBLE[]**, der für jede Sortierklasse die Anzahl der Werte angibt, die in diesen Bereich gehören, oder ein binäres Steuersignal **SWORD[1]**, das z.B. eine Sortierstation bedienen kann.

Im Beispiel **demo_26** (siehe Abb. 3.42) werden Zufallszahlen dem Modul Sorter einer Sortierung zugeführt. Das Sortiermodul erzeugt für jeden Wert, entsprechend der Klasse, ein **Eins**-Signal am Ausgang.

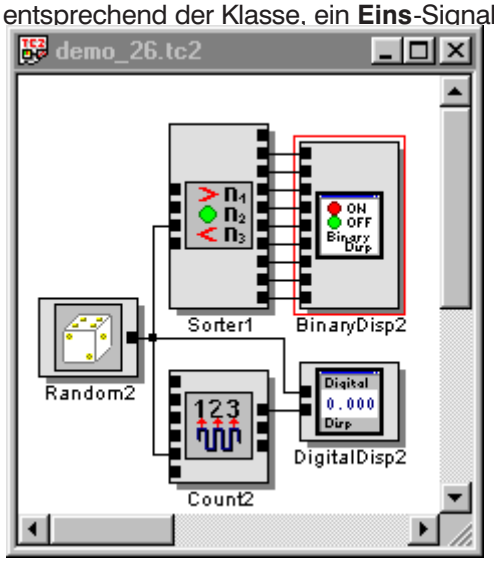


Abb. 3.42: Sortieren von Zufallszahlen

3.7 Module der Gruppe FlowControl

In diesem Kapitel lernen Sie einige Eigenschaften zur Steuerung des Programmablaufs in ICONNECT kennen.

- Software-Schleifen — Implementierung adaptiver Algorithmen
- Multiplexer — Zuteilung der Daten an unterschiedliche Teilgraphen

Damit können Sie eine Applikation in unterschiedliche Bearbeitungsphasen unterteilen.

3.7.1 Das Modul ForNext

Mit dem Modul **ForNext** bilden Sie eine Schleife mit definierten Grenzen. Das Beispiel **demo_31** (siehe Abb. 3.43) berechnet in einer Schleife den Wert

E= 100 * 4 + Startwert

Im Dialog des Moduls **ForNext** ist als untere Schleifengrenze 0, als obere Schleifengrenze 99 eingeben. Als Datentyp wird ein Vektor vom Typ **DOUBLE[]** verarbeitet. Im Formelinterpreter (Modul **Formula**) wird O1 = I1 + 4 berechnet, also die Multiplikation auf eine Addition zurückgeführt. Mit jedem Schleifendurchlauf wird der Wert 4 zu der bis dahin gebildeten Summe addiert.

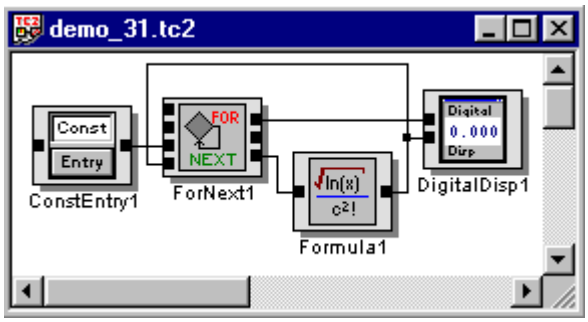


*Abb. 3.43: Signalgraph **demo_31***

3.7.2 Das Modul DEMUX

DEMUX kann Daten in unterschiedliche Teilgraphen lenken. In Beispiel **demo_32** (siehe Abb. 3.44) werden von zehn Datenpaketen neun auf das erste Display und eins auf das zweite Display geleitet. Der Zähler **Count1** erzeugt an seinem Pulsausgang, bevor der Zähler überläuft, ein 1-Signal. Dieses Signal schaltet den Demultiplexer um.

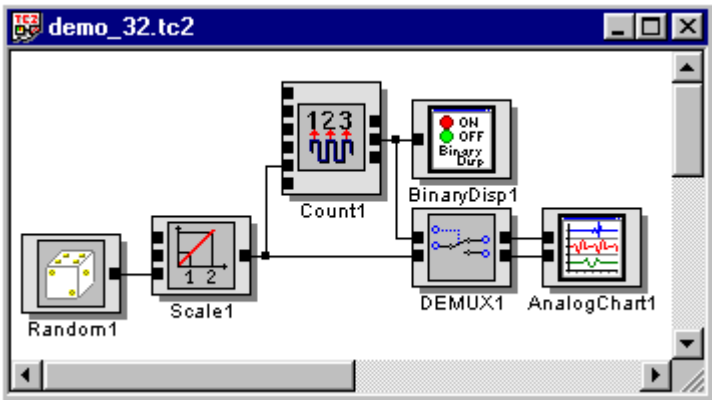


*Abb. 3.44: Signalgraph **demo_32***

Es muß sichergestellt werden, daß die Steuer-Daten (Schalten des Demultiplexers) und die Signal-Daten richtig synchronisiert werden.

Das Modul **DEMUX** meldet sich rechenwillig (siehe auch Kap. 6.2), wenn Daten am Eingang **I1** anliegen. Um sicherzustellen, daß der Demultiplexer zum richtigen Zeitpunkt schaltet, darf das Pulssignal nicht zeitlich nach den Daten des Zufallszahlengenerators am Modul ankommen. Sonst besteht die Möglichkeit, daß **DEMUX** diese Daten noch transportiert, bevor das Pulssignal ankommt. **DEMUX** meldet sich rechenwillig, nachdem die Daten vom Zufallszahlengenerator anliegen. Der Zähler ist ebenfalls rechenwillig, nachdem Random die Daten bereithält.

Das Modul mit der höherwertigen Priorität wird zuerst ausgeführt. Die Priorität ist mit 16-Bit kodiert, und kann Werte von 0 bis 65535 annehmen. Als Defaultwert wird 255 angenommen. Im Beispiel erhält das Modul **Count** die Priorität 256.

Mit einem Rechtsklick auf das Modulicon von **Count** öffnen Sie ein Popup-Menü, Linksklick auf **Priorität**, Sie können nun die Priorität des Moduls einstellen (siehe Abb. 3.45).

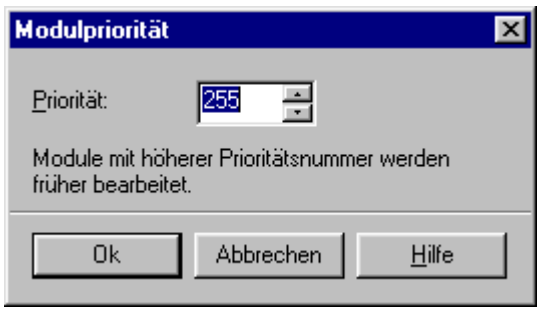


Abb. 3.45: Dialog für Prioritätsvergabe

Damit legen Sie fest, daß der Zähler vor dem Demultiplexer ausgeführt wird. **DEMUX** rechnet erst, wenn sowohl die Daten von **Random**, als auch das Pulssignal vom Zähler anliegen.

Hinweis:
Dieses Beipiel ist der Prinzipaufbau für eine Meßanordnung, die innerhalb von zehn Messungen einmal eine Kalibriermessung durchführt.

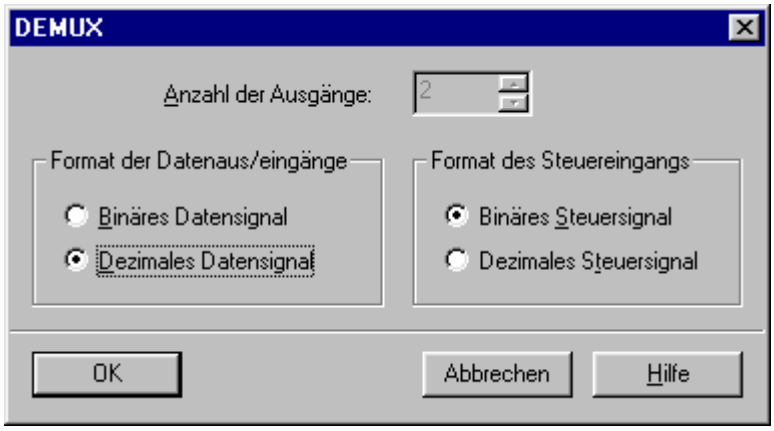


*Abb. 3.46: Dialog des Moduls **DEMUX***

Der Demuliplexer verarbeitet die Datentypen **SWORD[]** und **DOUBLE[]**. Somit kann der Multiplexer sowohl mit binären als auch dezimalen Signalen gesteuert werden. Ebenso können diese Datentypen über den Demultiplexer transferiert werden. Den Datentyp wählen Sie im Dialog (siehe Abb. 3.46).

Mit einem dezimalen Steuersignal können Sie die Anzahl der Datenausgänge variieren. Ein Demultipexen über eine beliebige Anzahl von Ausgängen ist dann möglich.

Im Beispiel **demo_33** wird der Multiplexer **MUX** eingesetzt. In diesem Beispiel werden zwei Datenquellen auf ein Display zusammen synchronisiert. Dies ist der Prinzipaufbau eines Signalgraphen, der zwei alternative Meßstationen auf eine Verarbeitungseinheit zusammenführt.

In diesem Kapitel wurden die wichtigsten Prinzipien von ICONNECT anhand von Beispielen eingeführt. Außerdem wurde ein Überblick über die Modulbibiliothek gegeben. Der Leser sollte nun in der Lage sein, selbst kleine Applikationem mit ICONNECT zu erstellen. Im nächsten Kapitel wird das Typeinfo und die Kommunikation zwischen den Modulen noch einmal im Überblick zusammengefaßt.

4. Kommunikation und Typinformation

Die Kommunikation ist in ICONNECT blockweise orientiert. Die kleinste Einheit ist ein einzelner Datenwert, der von einem Modul erzeugt wird, das eine Datenquelle verwaltet. Eine solche Datenquelle kann eine HW-Schnittstelle wie z. B. ein I/O-Board oder ein File-I/O Modul oder ähnliches sein.

Der Aufruf eines Moduls verursacht einen gewissen Verwaltungsaufwand. Der Aufruf durch einen einzelnen Datenwert verbraucht unnötig Rechenleistung. Um die Anzahl der Aufrufe gering zu halten, werden deshalb die Daten zu Blöcken zusammengefaßt. Die Größe eines solchen Blocks muß immer applikationsspezifisch eingestellt werden.

Zur Synchronisation von Datenblöcken im Graphen wurde der Mechanismus des Pakets eingeführt. Unter einem Paket wird eine Menge von Blöcken verstanden, die eine abgeschlossene Einheit darstellen. Dies können z.B. alle Blöcke sein, die zu einem einzelnen Teil gehören.

Treffen auf einem Kommunikationskanal zwei Pakete aufeinander, fallen mehr Daten an, als vom Rechner verarbeitet werden können. ICONNECT generiert eine Fehlermeldung und unterbricht den Ablauf. Mit dieser „Echtzeitbedingung“ werden die Daten im Graphen synchronisiert, da sich Pakete nicht überholen können. Ferner werden bei Modulen, die Pakete nur im ganzen verarbeiten, die einzelnen Blöcke (Block 0 bis Block n+1, Abb. 4.1) anhand von zusätzlichen Informationen zusammengefaßt, die im Kommunikationskanal enthalten sind.

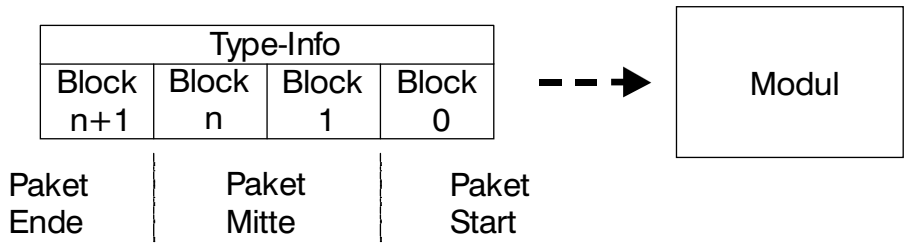


Abb. 4.1: Informations- und Datenblöcke in ICONNECT

Zu jedem Paket werden im Kanal Typinformationen (= **Type-Info)** hinzugefügt, die für jedes Paket neu bestimmt werden. Im Beispiel **demo_34** (siehe Abb. 4.2) wird an den Funktionsgenerator das TypeInfo-Display (**Module > Display > Debug > TIDisp**) angeschlossen. Dieses kann zum Testen von Signalgraphen benutzt werden. **TIDisp** stellt Type-Info und Paketstatus des Kommunikationskanals dar (siehe Abb. 4.3).

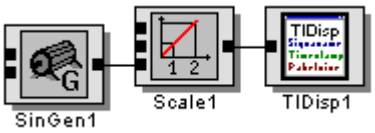


*Abb. 4.2: Signalgraph **demo_34***

TIDisp enthält als Informationen:

- Signalname — Name des Signals
- Min.Range — Untergrenze des Meßbereichs
- Max.Range — Obergrenze des Meßbereichs
- Samplerate — Abtastrate, mit der die Daten erzeugt werden
- Unit — Physikalische Einheit
- Paketstatus — Start, Mitte, Stop, Komplett
- TimeStamp — Zeitstempel, wann das Signal erzeugt wurde

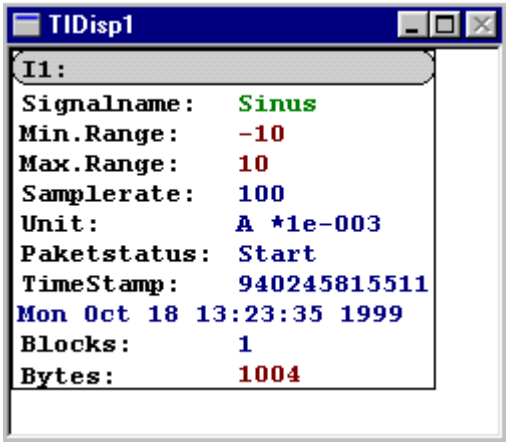


*Abb. 4.3: Type-Info von **demo_34***

Im Beispiel kann im Dialog des Funktionsgenerators die Anzahl der Blöcke im Paket variiert werden. Die Veränderungen, die sich dadurch bei der Kommunikation ergeben, sind im TIDisp sichtbar. Der Zeitstempel wird nur einmal pro Paket modifiziert, unabhängig von der Anzahl der Blöcke pro Paket.

ICONNECT modifiziert in allen Modulen, bei denen es sinnvoll ist, die Einheiten entsprechend den Algorithmen (siehe auch Kap. 3.1).
ICONNECT kennt zunächst alle SI-Einheiten entsprechend Tab. 4.1.

Größe	**Einheitenname**	**Einheitenzeichen**
Länge	Meter	m
Masse	Kilogramm	kg
Zeit	Sekunde	s
elektrische Stromstärke	Ampere	A
thermodynamische Temperatur	Kelvin	K
Stoffmenge	Mol	mol
Lichtstärke	Candela	cd

Tab. 4.1: Internationale Einheiten

ICONNECT leitet folgende Einheiten, entsprechend Tab. 4.2 ab.

Größe	**Einheitenzeichen**	**Ableitung**
Aktivität	Bq	s ^ -1
Beleuchtungsstärke	lx	lm m ^ -2
Druck	Pa	N m ^ -2
elektr. Kapazität	F	C V ^ -1
elektr. Leitwert	S	A V ^ -1
elektr. Spannung	V	W A ^ -1
elektr. Widerstand	Ω	V A ^ -1
Energie	J	N m
Energiedosis	Gy	J kg ^ -1
Frequenz	Hz	s ^ -1
Induktivität	H	V s A ^ -1
Kraft	N	m kg s ^ -2
Ladung	C	A s
Leistung	W	J s ^ -1
Lichtstrom	lm	cd steradian
magn. Fluß	Wb	V s
magn. Flußdichte	T	Wb m ^ -2

Tab. 4.2: Abgeleitete Einheiten in ICONNECT

Neben dem Paketstatus und den Typinformationen ist für die Kommunikation zweier Module der Datentyp (z.B. DOUBLE[], SWORD, ...) und dessen Bezeichner (z.B. TIME_DOMAIN, BIN,...) wichtig. Als Datentypen gibt es für die Modulkommunikation:

- DOUBLE 8 Byte Gleitkommazahl
- FLOAT 4 Byte Gleitkommazahl
- SWORD 4 Byte Integer (vorzeichenbehaftet)
- SDWORD 4 Byte Integer (vorzeichenbehaftet)
- UBYTE 1 Byte Integer, Ascii codiert
- UWORD 4 Byte Integer
- UDWORD 4 Byte Integer
- TYPEINFO Type Info

Aus diesen Typen können Arrays (= Vektoren) gebildet werden, die bestimmte Länge z.B **DOUBLE[3]** bzw. unbestimmte Länge **DOUBLE[]** besitzten. Arrays mit unterschiedlicher bestimmter Länge sind prinzipiell inkompatibel. Ein Vektor mit bestimmter Länge ist mit einem Vektor unbestimmter Länge kompatibel. Es wird dann in den Modulen zur Laufzeit überprüft, ob die Daten miteinander verknüpft werden können. Dazu wird in jedem Datenblock seine Länge (**Type-Info**, Abb. 4.1) als erster Wert mitübertragen. Das Modul stellt so fest, wieviele Daten am Eingang zur Verarbeitung anstehen.

Im Beispiel **demo_35** wird ein Datenvektor explizit mit dem Modul **Spy** dargestellt. Dieses Modul ist zu allen Datenausgängen, die oben angeführt sind, kompatibel und kann damit zum Testen eines Signalgraphen eingesetzt werden.

Hinweis:
Mehr zum Thema Datentypen finden Sie im Anhang (siehe Kap. 9.1).

5. Benutzerverwaltung

In ICONNECT ist eine Benutzerverwaltung mit individuellen Rechten und Paßwörtern eingebaut. Die Rechte gliedern sich in Aktionsrechte und Administrationsrechte.

- Aktionen — Ausführen eines Signalgraphen, Parametrisieren (Einstellen von Parametern in den Moduldialogen) sowie das Editieren (Erstellen von Signalgraphen)
- Administration — Verwalten von Benutzern

Das Ausführrecht erlaubt es dem Anwender, Signalgraphen zu starten, stoppen, bzw. pausieren. Mit dem Parametrisierrecht kann der Benutzer durch einen Rechtsklick mit der Maus auf ein Modul, eine Leitung oder den Fensterhintergrund (identisch mit Messung - Eigenschaften) Einstellungen vornehmen. Das Editierrecht gibt Ihnen die Möglichkeit, Module in den Signalgraphen einzufügen, zu löschen und zu verschieben. Außerdem erlaubt es alle weiteren Editierfunktionen aus dem Menüpunkt **Bearbeiten**.

Der Benutzer sieht in der Benutzerverwaltung nur die Einträge, die er laut seinen Administrationsrechten verändern darf. Ein Benutzer mit z. B. drei Administrationsrechten sieht alle Einträge. Ein Benutzer ohne Administrationsrecht kann die Benutzerverwaltung nicht öffnen. Der Anwender kann nur Rechte einsehen und verändern, die er selbst besitzt. Ein Benutzer ohne Editierrecht kann die Benutzerrechte weder einsehen noch verändern. Hat ein Benutzer zwar ein Administrationsrecht für eine Gruppe, ist aber selbst nicht in dieser Gruppe (z.B. darf er Gruppe 2 administrieren, aber nicht selbst editieren), so ist zusätzlich das Aktionsrecht ausgeblendet. Er kann das Recht in einem Eintrag weder ansehen noch ändern.

Den Dialog für die Benutzerverwaltung finden Sie im Menü **Extras > Benutzerverwaltung**. Links wird der gewählte Eintrag angezeigt, rechts die zugehörigen Rechte. Sind ein oder mehrere Rechte ausgeblendet (grau hinterlegt), so besitzt der angemeldete Benutzer nicht selbst das entsprechende Administrations- oder Aktionsrecht. Für jeden Benutzer muß ein Paßwort festgelegt werden (mindestens ein Zeichen). Dieses ist in der zweiten Zeile zu bestätigen.

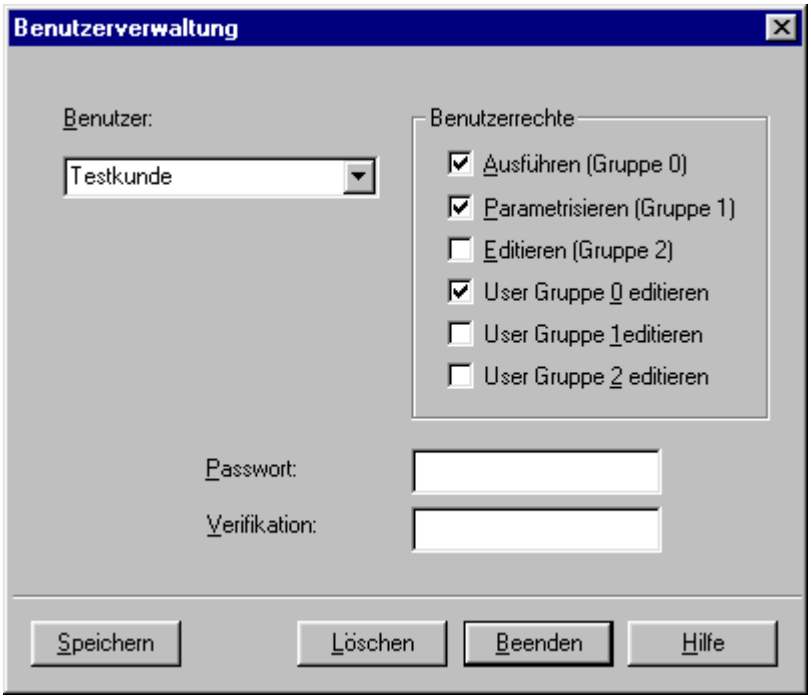


Abb. 5.1: Dialog Benutzerverwaltung

Um einen neuen Benutzer anzulegen, geben Sie den Namen im Feld **Benutzer** ein und legen Sie seine Rechte sowie das Paßwort fest. Haben Sie alle Einstellungen vorgenommen quittieren Sie die Eingaben mit Links-Klick auf den Button **Speichern**.

Hinweis:
*Durch ein Beenden des Dialogs **Benutzerverwaltung** ohne Speichern verliert man die gerade eingegebenen Benutzerdaten.*

Um die Einstellungen für einen bestehenden Benutzer zu ändern, ist dieser im Feld **Benutzer** auszuwählen. Anschließend können Sie die Rechte neu definieren. Die neuen Einstellungen müssen, unter erneuter Angabe des Paßworts, gespeichert werden.

Sie löschen einen Benutzer, indem er in der Auswahlbox ausgewählt wird und durch den Befehl **Löschen** gelöscht wird. Es erfolgt dabei keine Warnung. Der Benutzer wird unwiderruflich gelöscht.

Mit Links-Klick auf den Button **Beenden** schließen Sie den Dialog.

Jeder Benutzer hat das Recht, sein Paßwort zu ändern. Es muß das alte Kennwort eingegeben werden, anschließend das neue Paßwort und die Bestätigung. Hier ist auch ein leeres Paßwort erlaubt. Dazu wird lediglich das alte Paßwort eingegeben und mit OK bestätigt. Den Dialog finden Sie im Menü **Extras > Paßwort ändern**.

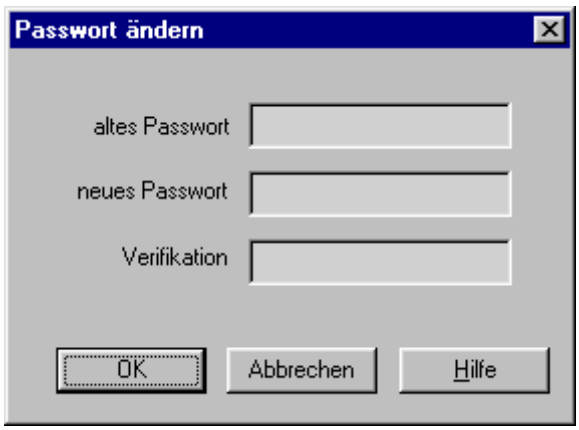


Abb. 5.2: Dialog Passwort ändern

6. Tips & Tricks

6.1 Ablaufsteuerung

In ICONNECT können Sie mehrere Signalgraphen öffnen und zugleich starten. In dieser parallelen Ausführungsart teilen sich mehrere Signalgraphen die Prozessorzeit (preemptives Multithreading). Bei geringer CPU-Auslastung laufen damit alle Graphen scheinbar gleichzeitig und ohne gegenseitige Beeinflussung. Steigt die Auslastung, so ist es sinnvoll, die Rechenzeit nicht gleichmäßig zu verteilen, sondern zeitkritische Abläufe zu bevorzugen. Dies erreichen Sie durch eine Höherpriorisierung des kritischeren Threads.

Öffnen Sie dazu den Parameterdialog der Ablaufsteuerung (**Messung > Eigenschaften**). Die Default-Einstellung der Priorität ist **hoch** (siehe Abb. 6.1) und kann bei Bedarf auf **niedrig** umgestellt werden. Damit wird der Signalgraph nur bearbeitet, wenn freie Rechenzeit zur Verfügung steht.

Die niedrige Prioritätsstufe wird dennoch gegenüber anderen Anwendungen bevorzugt. Wollen Sie CPU-Zeit an andere Windows-Applikationen abgeben, so müssen Sie dies im Zeitverhalten spezifizieren. Editieren Sie dazu das Feld **Wartezeit nach Schleifendurchlauf** (Default: 50ms). Vor der zyklischen Behandlung der ICONNECT-Quellmodule durch die Ablaufsteuerung wird Rechenzeit abgegeben. Dabei wird die minimal eingestellte Wartezeit an konkurrierende Applikationen vergeben. Ist die Solldurchlaufzeit größer als die eingestellte Wartezeit, so wird solange gewartet, bis diese erreicht ist. Damit können Sie eine konstante, also von den Modulrechenzeiten unabhängige Schleifendurchlaufzeit einstellen.

Hinweis:
Da Windows kein Echtzeit-Betriebssystem ist, kann für die korrekte Einhaltung von Zeitvorgaben keine Garantie übernommen werden! Alle Zeitangaben unterliegen vor allem bei höherer Rechner- oder Speicherauslastung relativ großen Schwankungen. Arbeiten Sie deshalb nie an der Auslastungsgrenze des Rechners. Um ein Auslagern von virtuellem Speicher zu vermeiden, deaktivieren Sie den virtuellen Arbeitsspeicher in der Systemsteuerung.

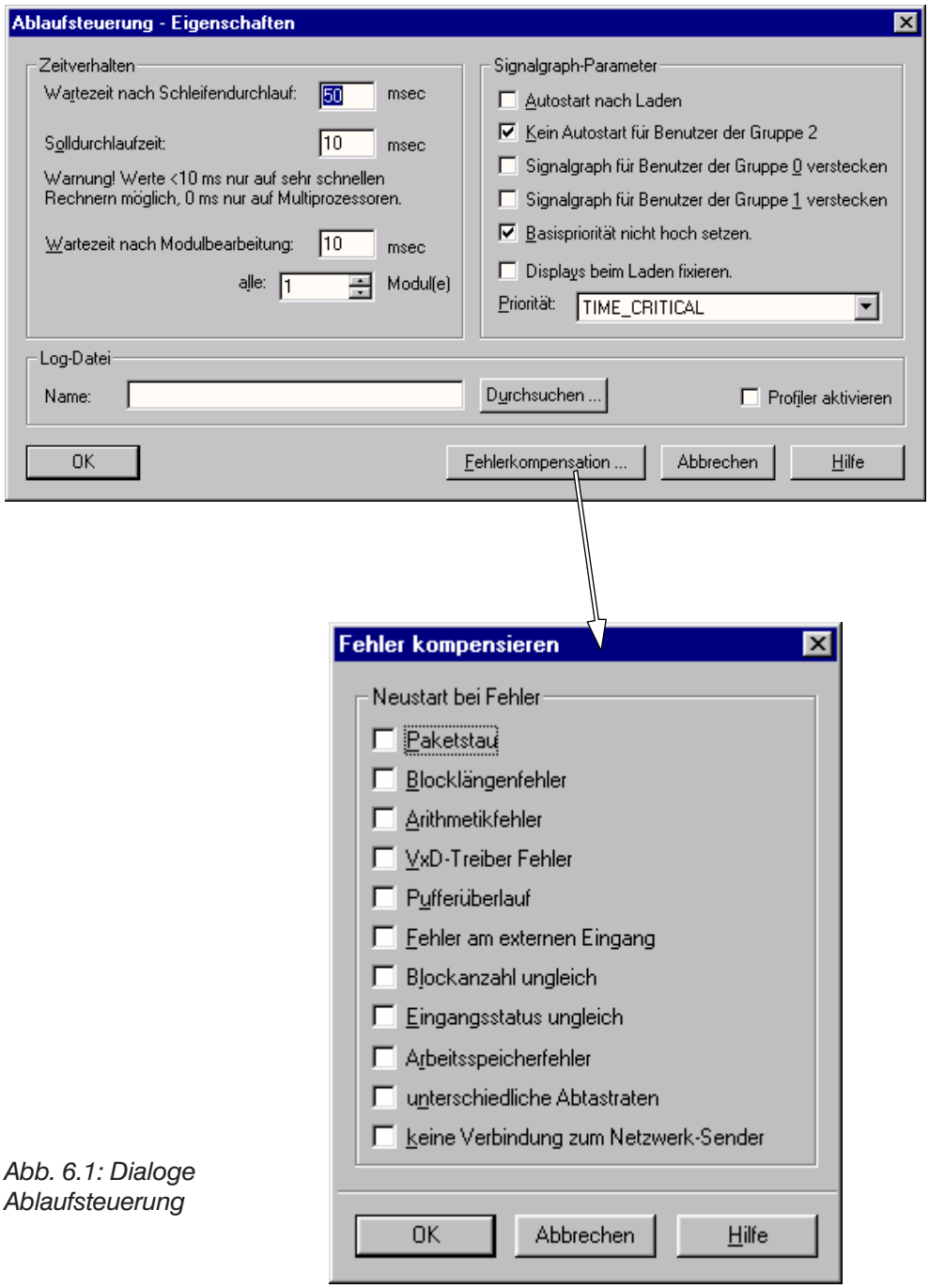


Abb. 6.1: Dialoge Ablaufsteuerung

X9750025-D010080MSC

Wollen Sie einen Signalgraphen beim Start von ICONNECT automatisch ausführen, so wählen Sie **Autostart nach Laden**. In der Berechtigungsebene **Ausführen (Gruppe 0)** können Sie den Signalgraphen verbergen. Damit erstellen Sie einen Signalgraphen für eine Runtime-Version eines ICONNECT-Programms.

Die unter **Log-Datei** einstellbare Datei protokolliert Fehlermeldungen (sofern **Neustart bei Fehler**, siehe Abb. 6.1, aktiviert wurde) und Modullaufzeiten im Debug-Modus:

```
Signalgraph FFT:***  Init        ***
Ausfuehrungzeiten fuer Signalgraph FFT
Modul: Random1     Count:15        Avg:10       Max:60
Modul: FFT1        Count:15        Avg:26       Max:60
Modul: Plot1       Count:15        Avg:0        Max:0

Signalgraph FFT:***  Done        ***
Signalgraph FFT:***  No Restart  ***
```

Alle Zeitangaben sind in ms (mit einer zeitlichen Auflösung der Messung von 10 ms). Neben den durchschnittlichen Rechenzeiten der Module werden auch die Zahl der Aufrufe (Count) und die Maximalzeiten angegeben.

6.2 Interpret-Programmierung

Das Modul **Interpret** stellt einen einfachen Interpreter dar und ermöglicht die Programmierung von individuellen Aufgaben und Funktionen.
Mit Aufruf des Dialogs erscheint ein Editor zum Eingeben der Programmabfolge. Die vordefinierten Funktionen finden Sie in der Online-Hilfe.

Ein Interpret-Programm hat folgenden Aufbau:

1. Ein-/Ausgänge und Variablen deklarieren
2. Initialisierungsphase (init) — Durchführung von Initialisierungsarbeiten, wie z.B. Speicherallokation, oder das Laden von Treibern
3. Ausführungsphase (execute) — Ausführen des Algorithmus
4. Abschlußphase (done) — Signalgraph wurde gestoppt. Abschlußarbeiten, wie z.B. Freigabe von Speicher, Hardwareressourcen

6.2.1 Ein-/Ausgangsdeklaration

Zur Festlegung von Moduleingangs- bzw. Modulausgangsports werden die Schlüsselwörter **input** bzw. **output** benutzt. Der Aufbau einer Portdeklaration erfolgt nach folgendem Muster:

Portrichtung [Triggerattribut] Portname (Validierungsinformation);

Portrichtung:	input | output
Triggerattribut:	trigger
Portname:	Mit einem Buchstaben beginnende Zeichenkette aus Buchstaben, Zahlen und '_'
Validierungsinformation:	"Datentyp", "Bezeichner" [, "Datentyp", "Bezeichner"]
Datentyp:	SWORD | SWORD[1] | SWORD[] | DOUBLE | DOUBLE[1] | DOUBLE[] | UBYTE[] | TYPEINFO
Bezeichner:	BIN | TIME_DOMAIN | TypeInfo | beliebige Zeichenkette

6.2.2 Variablendeklaration

Der Geltungsbereich von Variablen im Interpreter-Modul ist global. Eine Initialisierung während der Deklaration ist nicht möglich.

Beispiel:
falsch: int i=0; richtig: int i; i=0;

Variablen werden einzeln deklariert, Kommalisten sind nicht erlaubt.

Beispiel:
falsch: int i, j, k; richtig: int i; int j; int k;

Variablentyp Variablenname ['[' Arraygröße ']'];
Variablentyp: char | int | double
Variablenname: Mit einem Buchstaben beginnende Zeichenkette aus Buchstaben, Zahlen und '_'
Arraygröße: Positive ganze Zahl > 0

6.2.3 Aufruf vordefinierter Funktionen

Alle Variablen der Parameterliste von Funktionen, werden als Variablen-Referenzen (call-by-reference) übergeben. Damit sind im Interpret-Modul keine Zeiger notwendig.

Beispiel:

```
int Len; int Address; int Status; int erg;
char recv[41];
erg = enter( recv, 40, Len, Address, Status);
```

Dabei sind Len und Status Rückgabewerte (Referenzvariablen).

6.2.4 Programm-Sektionen

Die **Initialisierungsphase** (init) wird nur einmal, vor der ersten **Ausführungsphase** (d.h. nach Messung | Start), durchlaufen. Sie dient zum Initialisieren von Variablen oder Hardwarekomponenten. Während der **Initialisierungsphase** kann noch nicht auf Eingangsdaten zugegriffen werden.
Die **Ausführungsphase** (execute) wird bei jedem Programmlauf ausgeführt.
Die **Abschlußphase** (done) wird nur einmal - nach der letzten Ausführungsphase (d.h. nach Messung | Stop) - abgearbeitet.

Konventionen:

- Alle Sektionen sind optional.
- Ein leeres Programm wird akzeptiert.
- Vor der ersten Sektion muß mindestens eine Variable oder ein Port deklariert werden.
- Deklarierte Sektionen dürfen nicht leer sein.
- Parameterübergaben an Funktionen und Rückgabewerte sind nicht möglich (globale Variablen verwenden!).

Beispiel 1 (Gerüst eines Interpret-Programms):

```
input trigger i1 ( “TYPEINFO“, “TypeInfo“, “DOUBLE[]“,
“TIME_DOMAIN“);
output o1 ( “TYPEINFO“, “TypeInfo“, “DOUBLE[]“,
“TIME_DOMAIN“);
int i;
init
{
;
}
execute
{
;
}
done
{
;
}
```

6.2.5 Schreib-/Leseoperationen auf Streams

Auf Eingangsdaten kann nur innerhalb der execute-Sektion zugegriffen werden. Der Zugriff auf die Eingangsvariable i1 erfolgt analog dem Zugriff auf (Array-) Variablen. Um sicherzustellen, daß die Ausgangsvektorvariable o komplett gefüllt wird, ist der Zugriff auf Einzelelemente von o nicht erlaubt. Deshalb ist zur Ausgabe nur der << Operator erlaubt:

Beispiel 2 (Ein-/Ausgabe):

```
input trigger i1 ( “TYPEINFO“, “TypeInfo“, “DOUBLE[]“,
“TIME_DOMAIN“);
output o1 ( “TYPEINFO“, “TypeInfo“, “DOUBLE[]“,
“TIME_DOMAIN“);
int s;
int i;
execute
 {
 // o1 << i1; ist erlaubt, Zugriff auf Arrayelemente
damit jedoch nicht
 // möglich
 s = size( i1);
 for ( i=0; i<s; ++i) // ++i ist schneller als i++ (i =
i + 1)
  {
  o1 << i1[i] * 3.0; // Ausgabe
  }
 }
```

6.2.6 Behandlung der Type-Info-Struktur

In Beispiel 2 werden nur Daten ohne Typ-Informationen behandelt und ausgegeben. Da die zur Skalierung notwendigen Wertebereiche des Signals fehlen, kann eine Anzeige z.B. im Modul **Plot** nicht erfolgen. In Beispiel 3 werden die notwendigen Schritte zur Behandlung des **Type-Info** gezeigt:

Beispiel 3 (Kopieren des **Type-Info**):

```
input  trigger
i1("TYPEINFO","TypeInfo","DOUBLE[]","TIME_DOMAIN");
output
o1("TYPEINFO","TypeInfo","DOUBLE[]","TIME_DOMAIN");
char name[20]; char unit[20];
int erg; int status;
double d; double timestamp; double samplerate; double
rangemin;
double rangemax; double scale;

typeinfo_kopieren()
 {
 ti_copy( o1, i1);
 // kopiert name, timestamp, samplerate, rangemin,
rangemax, unit, scale,
//   jedoch keinen Status
 status = ti_getstatus( i1);
 ti_setstatus( status, o1);
 }

typeinfo_lesen()
 {
 ti_getname( name, i1);
 timestamp = ti_gettimestamp( i1);
 samplerate =  ti_getsamplerate( i1);
 rangemin =  ti_getrangemin( i1);
 rangemax = ti_getrangemax( i1);
 erg = ti_getunit( unit, i1);
 scale =  ti_getscale( i1);
 status = ti_getstatus( i1);
 }
```

```
typeinfo_setzen()
 {
 ti_setname( name, o1);
 ti_settimestamp( timestamp, o1);
 ti_setsamplerate( samplerate, o1);
 ti_setrangemin( rangemin, o1);
 ti_setrangemax( rangemax, o1);
 erg = ti_setunit( unit, o1);
 ti_setscale( scale, o1);
 ti_setstatus( status, o1);
 }

execute
 {
 typeinfo_kopieren();
 typeinfo_lesen();
 typeinfo_setzen();
 // sonstige Aktionen
 }
```

6.2.7 Interpret als Datenquelle

Wird kein Eingang mit dem Attribut **trigger** deklariert, so wird das Interpret-Modul in jedem Zyklus der Ablaufsteuerung aufgerufen. Damit lassen sich Datenquellen modellieren.

Beispiel 4 (Interpret als Quelle):

```
output o1("TYPEINFO","TypeInfo","SWORD[]","BIN");
int i;
init
 {
 i=0;
 }
execute
 {
 typeinfo_setzen();
 o1 << i;
 i  = ~i & 1;
 }
```

Das Beispiel erzeugt ein Steuersignal (0 1 0 1 0 ... Folge).

6.2.8 Trigger

Triggereingänge, d.h. Eingänge, für die das Attribut **trigger** spezifiziert ist, sind defaultmäßig ODER-verknüpft. Liegt an einem der Eingänge ein Signal an, so wird die execute-Sektion aufgerufen. Mit der Variablen TRIGGER = T_AND kann eine UND-Verknüpfung der Triggerbedingung festgelegt werden. Damit meldet sich das Modul rechenbereit, wenn an allen Triggereingängen Signale anliegen.

Beispiel 5 (Trigger):

```
input  trigger
i1("TYPEINFO","TypeInfo","DOUBLE[]","TIME_DOMAIN");
input  trigger
i2("TYPEINFO","TypeInfo","DOUBLE[]","TIME_DOMAIN");
output
o1("TYPEINFO","TypeInfo","DOUBLE[]","TIME_DOMAIN");
TRIGGER = T_AND;
int s1; int s2; int i;
execute
 {
 s1 = size( i1);
 s2 = size( i2);
// Laufzeitfehler bei unterschiedlichen Blocklängen
abfangen
 if ( s1 != s2)
  print("Blocklängenfehler");
 else
  {
  typeinfo_setzen();
  for ( i=0; i<s1; i=i+1)
   {
// bei falschem Index -> Laufzeitfehler
   o1 << i1[i] * i2[i];
   }
  }
 }
```

6.2.9 Beispiele

Lineare Regression:

Die Polynomkoeffizienten einer linearen Regression y = A0 + A1*x beschreiben eine Regressionsgerade durch ein Signal (Paket). Das Signal y liegt am Ausgang des **Formula**-Moduls. Mit **Sync** wird ein zugehöriges Zeitsignal x generiert. Die Koeffizienten A0, A1 und x werden zusätzlich zum Signal y an die Eingänge von Interpret gelegt. Am Ausgang wird die Abweichung von der Geraden dargestellt.

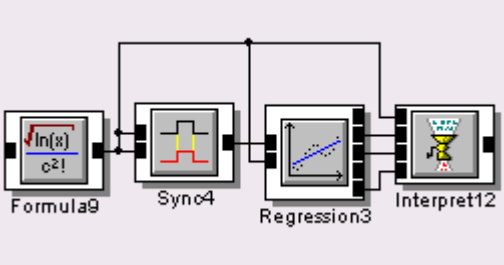


Abb. 6.2: Signalgraph Regression

```
input trigger
y("TYPEINFO","TypeInfo","DOUBLE[]","TIME_DOMAIN");
input trigger
A0("TYPEINFO","TypeInfo","DOUBLE[1]","TIME_DOMAIN");
input trigger
A1("TYPEINFO","TypeInfo","DOUBLE[1]","TIME_DOMAIN");
input trigger
dx("TYPEINFO","TypeInfo","DOUBLE[1]","TIME_DOMAIN");
output
out("TYPEINFO","TypeInfo","DOUBLE[]","TIME_DOMAIN");
int s;
int i;
execute
 {
 s = size(y);
 ti_copy(out, y);
 for ( i = 0 ;  i < s; ++i )
  {
  out << y[i]- A0[0] / dx[0] + A1[0];
  }
 }
```

Automatisierung einer EMV-Messung:

Zur Untersuchung der HF-Einstreuung in das Stromversorgungskabel eines Prüflings soll ein Meßaufbau automatisiert werden. Eine Liste von Frequenz- und Leistungswerten wird dazu aus einer Datei eingelesen und über IEEE488-BUS an einen Oszillator (Hameg HM 8133-2) und einen HF-Leistungsverstärker (Frankonia FLL 75) zur Steuerung ausgegeben. Der Verstärker gibt seine HF-Leistung an eine Stromzange ab. Hiermit werden in der Leitung Ströme induziert, die den Prüfling stören. Die Abweichung vom Sollwert des Prüflings bei der jeweiligen Störung wird mit einem Multimeter (Keithley 2000) erfaßt. Die Störung als Funktion der Frequenz wird in einer Datei abgelegt und grafisch dargestellt.

```
input I_Frq ("TYPEINFO", "TypeInfo", "DOUBLE[]",
"TIME_DOMAIN");
input I_Dbm ("TYPEINFO", "TypeInfo", "DOUBLE[]",
"TIME_DOMAIN");
input I_Schalter ("TYPEINFO", "TypeInfo", "SWORD",
"BIN");
input I_Start ("TYPEINFO", "TypeInfo", "SWORD", "BIN");
output O_Frq ("TYPEINFO", "TypeInfo", "DOUBLE[]",
"TIME_DOMAIN");
output O_U ("TYPEINFO", "TypeInfo", "DOUBLE[]",
"TIME_DOMAIN");

int status;
int s; int l; int t;
double f; double p; double v;
char beffrq[60];
char wertfrq[40];
char befdbm[60];
char wertdbm[40];
char volt_werte[80];
```

```
typeinfo_setzen_f()
 {
 ti_setname("Frequenz",O_Frq);
 ti_settimestamp(time(),O_Frq);
 ti_setsamplerate(1.5,O_Frq);
 ti_setrangemin(0.0,O_Frq);
 ti_setrangemax(100000000.0,O_Frq);
 ti_setscale(1.0,O_Frq);
 ti_setstatus(3,O_Frq);
 }

typeinfo_setzen_v()
 {
 ti_setname("Volt",O_U);
 ti_settimestamp(time(),O_U);
 ti_setsamplerate(1.5,O_U);
 ti_setrangemin(0.0,O_U);
 ti_setrangemax(1000.0,O_U);
 ti_setscale(1.0,O_U);
 ti_setstatus(3,O_U);
 }
```

```
init
 {
// Initialisierung des IEEE488-Interfaces (Keithley
KPC-488.2)
 initialize (21, 0);
 t=1000000;
 // DMM Funktion V, Range Auto
 send(16,":volt:dc:rang:auto on",status);
// Hameg Frequenz, Abschwächung
 send(7,"frq:150000,dbm:-50,am2:80",status);
 }

execute
 {
 if ( I_Start == 1) t = 0;
 // Anzahl der Meßfrequenzen
 s = size(I_Frq);
 if ( I_Schalter == 1 && t<s)
  {
  // Erstellen des Befehls zum Einstellen der Frequenz
am
  //Hameg HM 8133-2
  beffrq = "frq:";
  f = I_Frq[t];
  ftoa( wertfrq, f, 20);
  strcat( beffrq, wertfrq);

  // Erstellen des Befehls zum Einstellen der Abschwä-
chung am
  // Hameg HM 8133-2
  befdbm = "dbm:";
  p = I_Dbm[t];
  ftoa( wertdbm, p, 20);
  strcat( befdbm, wertdbm);

  // Senden der Einstellungen an Hameg HM8133-2
(IEEE488-Geräte
  // adresse 7)
  send ( 7, "op1:", status);
  send ( 7, beffrq, status);
  send ( 7, befdbm, status);

  // Wartezeit von 1.5 Sekunden
  Sleep(1500);
```

```
// Senden von Einstellungen an Keithley 2000 (IEEE488-
Geräteadresse
// 16)
// kontinuierliche Messung ausschalten
  send ( 16, "init:cont 0", status);
  send ( 16, "meas:volt:dc?", status);
// Einlesen eines Spannungswerts (mit maximal 40 Zei-
chen)
// vom Keithley 2000
// l = tatsächliche Anzahl Zeichen des Strings
  enter ( volt_werte, 40, l, 16, status);
  send ( 16, "init:cont 1", status); // kontinuierliche
Messung ein
  v = atof( volt_werte );
  O_Frq<<f;
  O_U<<v;
  typeinfo_setzen_f();
  typeinfo_setzen_v();
  t++;
  }
 }
```

7. Debugger

Der grafische Debugger in ICONNECT dient zur Fehlersuche in Signalgraphen, die zur Laufzeit des Programmes auftreten. Damit läßt sich der Signalfluß steuern (unterbrechen, Einzelschritt) und überwachen (Watchfenster). Der Debugger unterbricht die Ausführung der Programme an festgelegten Stellen (Breakpoint) und

- erlaubt die Fehlersuche im Signalgraphen,
- gibt einen Hinweis auf den Fehlerort und
- führt zu einer schnellen Programmprüfung.

7.1 Breakpoint

An einem Breakpoint wird das Programm angehalten.

Breakpoint setzen:
- Plazieren Sie den Mauszeiger über dem Modul im Signalgraphen
- Drücken Sie die rechte Maustaste
- Wählen Sie **Breakpoint setzen**/**löschen** aus dem Pull-down Menü

Hinweis:
Sie können einen Breakpoint auch im laufenden Signalgraphen setzen. Breakpoints sind in unbegrenzter Anzahl möglich und werden zusammen mit dem Signalgraphen gespeichert.

Ein Modul mit gesetztem Breakpoint wird im Signalgraphen mit einem Rahmen gekennzeichnet (siehe Abb. 7.1). ICONNECT bleibt vor der Ausführung des Breakpoints stehen.

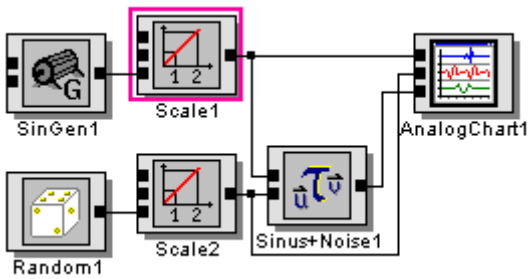


*Abb. 7.1: Signalgraph mit Breakpoint für das Modul **Scale1***

Im Debugger-Modus stehen die Befehle

- **Einzelschritt** (Strg+E) Signalgraph schrittweise ausführen
- **Bearbeitung fortfahren** (Strg+W) Signalgraph bis zum nächsten Breakpoint ausführen

im Menü Messung zur Verfügung.

Breakpoints löschen	Watchfenster löschen	Unter-brechen	Einzel-schritt	Bearbeitung fortfahren

Abb. 7.2: Ausschnitt aus der Symbolleiste mit Debug-Befehlen

Breakpoint löschen:

- Plazieren Sie den Mauszeiger über dem Modul im Signalgraphen
- Drücken Sie die rechte Maustaste
- Wählen Sie **Breakpoint setzen/löschen** aus dem Pull-down Menü oder wählen aus dem Menü **Messen > Breakpoints löschen**, um alle Breakpoints zu löschen

Wenn Sie den Signalgraphen unterbrechen (Menü **Messung > Unterbrechen**), können Sie den Signalgraphen ebenfalls in Einzelschritten ausführen.

7.2 Watchfenster

ICONNECT zeigt in Watchfenstern (siehe Abb. 7.3) den Inhalt eines Kommunikationskanals an. Mit zwei Watchfenstern, vor und nach einem Breakpoint, können Sie die Eingangs- und Ausgangsdaten eines Moduls kontrollieren.

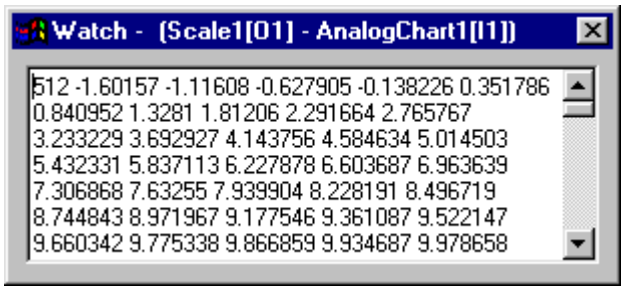


Abb. 7.3: Watchfenster

Watchfenster öffnen:
- Signalgraph unterbrechen
- Doppelclick auf Leitung öffnet das Watchfenster

Ist der Signalgraph unterbrochen, gibt es eine weitere Möglichkeit, sich den Inhalt eines Kommunikationskanals anzeigen zu lassen. Positionieren Sie dazu den Mauszeiger im Signalgraphen über die gewünschte Linie.

Der Titel im Watchfenster listet die beiden Module, die mit dem beobachteten Kanal miteinander verbunden sind. Watchfenster sind in unbegrenzter Anzahl möglich.

Hinweis:
Watchfenster zeigen Daten nur im unterbrochenen Signalgraphen an.

Watchfenster löschen:
Wählen Sie **Watchfenster löschen** aus dem Menü Messen, um alle offenen Watchfenster zu entfernen.

9. Anhang

9.1 Datentypen in ICONNECT

ICONNECT-Datentyp	C-Datentyp	Bytes	Bereich	Auflösung
DOUBLE DOUBLE[1] DOUBLE[]	double	8	± 1,7E308	15 Digits
FLOAT FLOAT[1] FLOAT[]	float	4	± 3,4E38	7 Digits
SDWORD SDWORD[1] SDWORD[]	long	4	-2 147 483 648 ... 2 147 483 647	
UDWORD UDWORD[1] UDWORD[]	unsigned long	4	0 ... 4 294 967 295	
SWORD SWORD[1] SWORD[]	int (32 Bit)	4	-2 147 483 648 ... 2 147 483 647	
UWORD UWORD[1] UWORD[]	unsigned int	4	0 ... 4 294 967 295	
UBYTE UBYTE[1]	unsigned char	1	0 ... 255	
UBYTE[]	CString			

9.2 Module in Verbindung mit externen Datenquellen

Modulname	Eigenschaften
LoadTable/ **SaveTable**	max.16 Spalten (=16 Eingänge); kein Zeitverhalten, sondern schnellstmögliche Datenübertragung (für Ascii, Excel etc.);
LoadAscii/ ***S*aveAsci*i***	ICONNECT - Datenformat mit Zeit und Einheit Zwei Kanäle (x,y oder x,t); wahlweise incl. Zeit verhalten, d.h. Daten werden beim Laden mit der gleichen Geschwindigkeit übertragen, mit der sie ursprünglich gespeichert wurden.
DBLoad/ **DBSave**	Verbindung zu Datenbank (Access)
FileDlg	Erlaubt interaktive Eingabe von Dateiname und Pfad (für LoadTable, LoadAscii und Dbload)

9.3 Fehlermeldungen

• Arithmetikfehler
Dieser Fehler tritt bei einer Division durch 0 auf, oder wenn das Ergebnis einer Rechnung den Wertebereich überschreitet (z.B. beim Versuch die Zahl 4096 mit 12 Bit zu codieren).

• Blockanzahl stimmt nicht überein
Die Vorgängermodule eines Moduls mit mehreren Eingängen liefern unterschiedlich viele Blöcke. Das Modul arbeitet mit Datenblöcken, weshalb diese in gleicher Anzahl geliefert werden müssen.

• Blocklängen an den Eingängen stimmen nicht überein!
Die Vorgängermodule eines Moduls mit mehreren Eingängen liefern Datenblöcke mit unterschiedlicher Anzahl von Daten. Für eine korrekte Verarbeitung müssen diese aber gleich sein.
Behebung: Das kürzen von Blöcken kann z.B. mit dem Modul Resampling erreicht werden (Datenreduzierung).
Eine Synchronisierung von Daten ist mit dem Modul **Sync** möglich.

• Dateifehler:
Eine Datei konnte nicht geladen oder gespeichert werden. Mögliche Ursachen: Datei existiert nicht, befindet sich nicht am spezifizierten Ort (Pfad), ist beschädigt, besitzt ein falsches Format (alte Version), ist schreibgeschützt oder auf dem Speichermedium (Festplatte) ist nicht genügend freier Platz.

• Das Modul ist in der Demoversion nicht funktionstüchtig.
Bestimmte Module können nur in der lizenzierten Version verwendet werden. Für eine Lizenzierung wenden Sie sich bitte an Ihren Händler.

• Das Modul konnte nicht erzeugt werden
Ursachen: Modul ist nicht lizensiert, beschädigt oder hat falsche Version.
Behebung: Falls es sich um eine Demo-Version handelt, diese neu installieren, andernfalls Modul lizenzieren lassen (Wenden sich an Ihren Händler).

• Das Zeitsignal auf Kanal I1 ist falsch! Überprüfen Sie Ihren Signalgraphen.
Modul das Daten mit Hilfe des Zeitsignals (=Timestamp) verarbeitet, erhält ungültigen Timestamp.
Fehlersuche: Mit dem Modul TIDisp kann das Zeitsignal überprüft werden. Liefert das Modul kein oder kein gültiges Zeitsignal so zeigt TIDisp „-„ als Timestamp an.
Behebung: Modul, das falsches Zeitsignal erzeugt neu konfigurieren oder durch anderes Modul ersetzen.

• Der Signalgraph enthält eine rekursive Schachtelung von Makros!
Eine rekursive Schachtelung von Makros ist nicht sinnvoll. Ein entsprechender Signalgraph kann nicht gestartet werden .

• Der Signalgraph konnte nicht geöffnet werden.
Überprüfen sie ob alle benötigten Module vorhanden sind.
Mögliche Fehler: Modulpfad falsch eingestellt, Module fehlen oder sind für diesen Dongle nicht lizenziert.

• Die Datei ist kein DisplayManager Signalgraph
Das gewählte Dokument entspricht nicht dem benötigten Format und wurde deshalb nicht geladen.
Mögliche Ursache: Beim Versuch einen Signalgraphen (*.tc2) oder eine andere Datei zu laden war ein DisplayManager-Fenster aktiv. Ein aktives Ausgabefenster eines DisplayManagers erlaubt nur das Laden des eigenen speziellen Dateiformats.
Um einen Signalgraphen laden zu können, ist vorher das Fenster des Signalgraphen zu aktivieren.

• Die Resource ID eines Moduls ist doppelt vorhanden
Ein Modul ist unter verschiedenen Namen mehrfach im Modulverzeichis vorhanden.
Behebung: Module löschen und Programm neu installieren.

• Dieser Eingang des Moduls ist schon benutzt.
Eingänge können nur von einer Datenquelle Daten erhalten.

• Dieser Kanal wird bereits von einem anderen Sender Benutzt.
Es können nicht zwei Module Communicate auf den selben Kanal schreiben.
Wählen sie einen andern Kanal

· Es sind keine Quellmodule im Signalgraph vorhanden
Ein Signalgraph ohne Datenquelle ist nicht sinnvoll und kann deshalb nicht gestartet werden.

• Einheiten stimmen nicht überein:
Es ist nicht sinnvoll Daten mit unterschiedlichen Einheiten zu addieren oder zu subtrahieren.

• Falsche ICONNECT Version
Der Signalgraph, der geladen werden sollte besitzt eine Versionsnummer die mit der ICONNECT-Version nicht kompatibel ist.

• Fehler bei der Parameterübergabe am EXT-Eingang
Am EXT-Eingang werden ungültige Parameter geliefert.
Behebung: Konfigurieren Sie das Modul, das die Parameter liefert, neu .

• Fehler bei der Parameterübergabe am Extern - Eingang:
Die Parameter, die über den EXT-Eingang an das Modul geliefert werden, entsprechen nicht dem erwarteten Format. Ist das Modul ParamConv die Parameterquelle, so muß dieses neu eingestellt werden. (Siehe ParmConv.hlp)
Überschreiten die Parameter bestimmte Grenzen, kann diese Fehlermeldung auftreten. (Beispiel: Untergrenze für Darstellung eines Sinussignals liegt bei 2, oder Obergrenze liegt unter Untergrenze).

• Kein ICONNECT Signalgraph
Das ausgewählte Dokument enthält keinen gültigen Signalgraphen oder ist beschädigt.

• Kein Speicher mehr verfügbar:
Mögliche Ursachen sind zu viele offene Applikationen, zu wenig Arbeitsspeicher, oder eine zu kleine Festplatte. Auch können Module bei übertriebenen Puffergrößen viel Arbeitsspeicher allokieren.
Behebung: Als erstes alle nicht benötigten Applikationen (Word, etc.) beenden. Der virtuelle Arbeitsspeicher sollte auf einer Festplatte liegen, die genügend freien Platz bietet (mindestens 64MB). Reicht dies nicht aus, ist der Kauf einer größeren Festplatte und evtl. auch von mehr Arbeitsspeicher empfehlenswert.
Wenden Sie sich an den System-/Signalgraphentwickler falls diese Problem wiederholt auftritt.

· Modul xxx liefert Fehler bei der Initialisierung
Es wurde versucht Mit einer ICONNECT Entwickler-Version Module zu benutzen, die zu der ICONNECT Runtime-Version gehören oder umgekehrt.
Behebung: Modulpfad zu den korrekten Modulen anlegen. Andernfalls ICONNECT neu installieren.

· Obligatorischer Eingang nicht verdrahtet
Ein Eingang der vom Modul zur Verarbeitung benötigt wird, ist nicht angeschlossen.
Behebung: Verdrahtung überprüfen, entsprechende Hilfe-Datei zum Modul lesen.

· Paketstatus stimmt nicht überein!
Ursache: Datenpakete treffen asynchron ein (Beispiel: Ein Eingang erhält Pakete mit mehreren Blöcken, ein weiterer Eingang des selben Moduls aber nur Datenpakete mit einem Block).
Mögliche Behebung: Synchronisation durch das Modul Sync.

· Paketstau
Tritt auf, wenn das Vorgängermodul schneller Datenpakete liefert, als es dieses Modul verarbeiten kann oder darf.
Tritt auch auf, wenn ein Modul mehrere Vorgängermodule besitzt, die parallel Datenpakete in unterschiedlichen zeitlichen Abständen liefern.
Behebung: Durch das Modul Sync können überzählige Pakete übersprungen werden (Datenverlust) und somit Paketstau vermieden werden.

· Pufferüberlauf:
Der von diesem Modul verwendete Datenpuffer konnte nicht alle Daten speichern. Tritt auf, wenn der Datenpuffer zu klein ist, oder die Verarbeitung der Daten zu aufwendig ist.

· Sind folgende Typbezeichner kompatibel?
Kein Fehler sondern ein Hinweis: Die Datentypen sind kompatibel, aber die Typbezeichner sind unterschiedlich. Da Typbezeichner auch manuell vergeben werden können, kann hier der Benutzer (Signalgraph-Entwickler) entscheiden, ob eine Verknüpfung dieser Ein/Ausgänge sinnvoll ist.

· Status von zwei Eingängen stimmt nicht überein:
Tritt auf, wenn Vorgängermodule Datenpakete asynchron liefern, das Modul aber die Daten paketweise verarbeitet. Abhilfe kann das Modul Sync schaffen (Synchronisation von Daten).

· Treiberfehler:
Ein virtueller Gerätetreiber (VxD) kann nicht korrekt arbeiten, weil z.B. keine unterstützte Hardware (A/D-Wandlerkarte o.ä.) vorhanden ist, die eingestellten I/O Adressen, DMAs, IRQs usw. ungültig oder bereits benutzt sind.
Behebung: Hardware überprüfen (beschädigt, falsch eingebaut), Treibereinstellungen an die Systemkonfiguration (Gerätemanager > Einstellungen in der Systemsteuerung) des Rechners anpassen.

· Unterschiedliche Abtastraten an den Eingängen
Modul benötigt an einigen Eingängen die gleiche Sample Rate (siehe entsprechende Modul-Hilfe)
Die Abtastraten können mit dem Modul TIDisp visualisiert und verglichen werden.

· Unterschiedliche Anzahl von Validierungsparametern! 2 vs. 1.
Ein Port der mehrere Parameter benötigt kann nicht mit einen Port verbunden werden der nur einen Parameter liefert.

· Unterschiedliche Basistypen
(Datentypen nicht kompatibel)
Diese Module können nicht miteinander verdrahtet werden.
Mögliche Lösung: Typkonvertierung mit den Modul TypeCast kann oft eine Verbindung trotz unterschiedlicher Datentypen ermöglichen. Dies ist aber i.A. nicht zu empfehlen, da hierbei je nach Konvertierungsart Rundungsfehler auftreten, ganze Datenvektoren weggeschnitten werden oder eine solche Konvertierung einfach nicht sinnvoll ist.

· Validierung ist fehlgeschlagen. Unterschiedliche Typen
Diese Module können auf keinen Fall miteinander verdrahtet werden. Eine Verknüpfung diese Module ist niemals sinnvoll.

· Warnung. Der Signalgraph enthält einen Zyklus! (Die Messung kann evtl. nur durch einen Abbruch gestoppt werden.)
Kein Fehler, aber falls der Zyklus keine Abbruchbedingung enthält, kann der Signalgraph nur noch über den Not-Aus (Knopf/Menüeintrag) gestoppt werden.