DIE GRUNDLAGEN

MESSEN & STEUERN

TESTPOINT REFERENZ

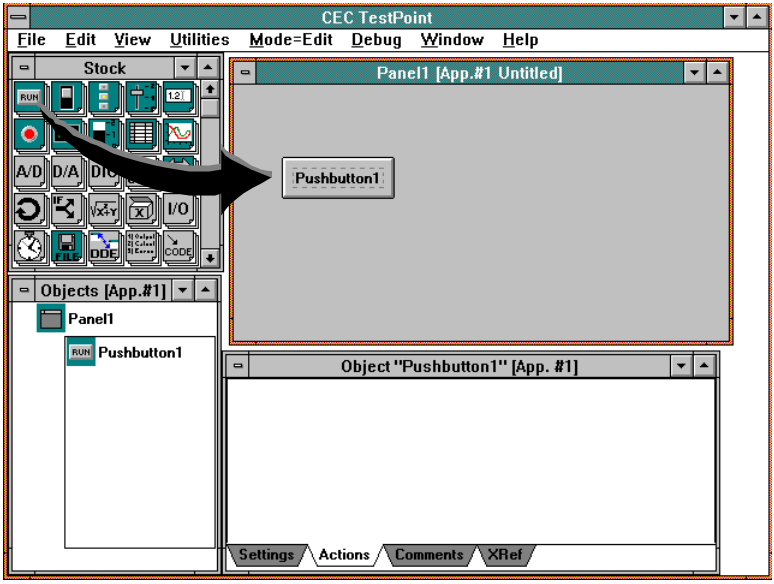
Die Grundlagen

Kapitel 1. Vorstellung

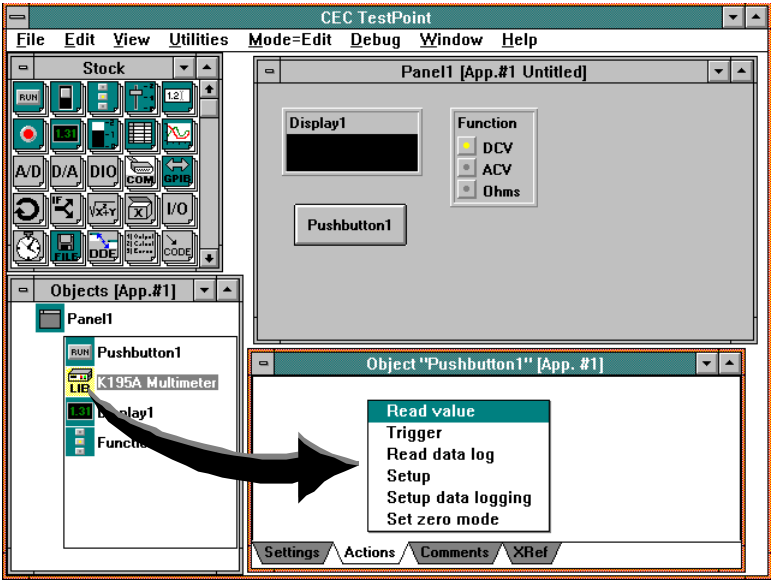
Wenn Sie „**TestPoint - Schnell erlernt**“ noch nicht gelesen haben, ist es sinnvoll, für eine Vorstellung des TestPoint-Konzepts dort zu beginnen.

Wie Sie TestPoint benutzen

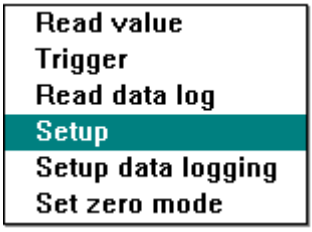
Sie beginnen damit, daß Sie TestPoint Objekte, wie Schalter, Anzeigen, Grafiken, GPIB Geräte, A/D-Karten, mathematische Formeln und Datendateien einfach vom Stock Fenster in Ihre Anwendung ziehen:



Dann ziehen Sie die Objekte in die Action List:



und wählen die Aktionen aus einem Menü aus, um TestPoint die Schritte mitzuteilen, die für die Ausführung Ihrer Anwendung erforderlich sind:



TestPoint erzeugt daraus Action Zeilen in einem klaren, lesbaren Format. Diese beschreiben die Aktionen, die Sie ausgewählt haben:

1)	Setup	K195A Multimeter	Function="DCV"
2)	Read value	K195A Multimeter	
3)	Set	Display1	to K195A Multimeter

Grundlegende Konzepte

Es gibt wirklich nur wenige Grundlagen, die man beachten muß, um TestPoint erfolgreich einzusetzen.

+ TestPoint hat Panels, Objekte mit Daten, Action Listen und Actions.

+ Das Panel ist die Benutzerschnittstelle für Ihre Anwendungen.

Sie können mehrere Panels (Fenster) verwenden, indem Sie Panel Objekte benutzen.

+ Übernehmen Sie Objekte durch Ziehen vom Stock in ein Panel oder Objects Fenster.

Sie können Objekte innerhalb einer oder aus weiteren Anwendungen (z. B. aus Beispielen oder Gerätebibliotheken) über die Zwischenablage kopieren und einfügen.

+ Fast alle Objekte haben Action Listen.

Es gibt viele Action Listen für verschiedene Objekte. TestPoint kann mehrere Action Listen zur gleichen Zeit ausführen (Multitasking).

+ Action Listen werden ausgeführt, wenn Ereignisse eintreten.

Ereignisse können Mausklicks (auf Schalter, Knöpfe, etc.), Timer und hardwaremäßige Ereignisse sein (z. B. A/D-Wandlung durchgeführt oder Daten der seriellen Schnittstelle liegen vor).

+ Objekte haben Daten.

Objekte verhalten sich wie **Variablen**. Sie beinhalten Daten und können als Parameter für andere TestPoint Aktionen benutzt werden.

+ Objekte haben Aktionen, die Sie ausführen können.

Diese kontrollieren Hardware, führen Berechnungen aus, usw. Sie ändern außerdem die Datenwerte der Objekte.

+ Sie erzeugen Action Zeilen durch Ziehen der Objekte in die Action Listen.

Sie können weiterhin die Einstellungen der Objekte in Action Listen verändern, indem Sie diese ins Action List Fenster ziehen.

Nutzen Sie die Beispiele

Es gibt viele Beispiele, die zum Lieferumfang Ihres TestPoint-Paketes gehören. Nutzen Sie diese. Schauen Sie diese an. Wenn Sie einen guten Ansatzpunkt für Ihre Anwendung gefunden haben, speichern Sie diese unter neuem Namen ab und benützen sie.

Kapitel 2. Objekte – Übersicht

Objekte mit Benutzerschnittstelle


Bar Indicator


Bitmap Switch


Data-Entry


Display


Grid


Indicator


OLE


Panel


Picture


Pushbutton


Selector


Slider


Switch


Text


VBX Controls

Ein-/Ausgabe Objekte


A/D I/O


D/A


Digital


File


GPIB


Port I/O


RS232

Präsentations Objekte

 Graph

 Report

Other Objects

 Action

 Case

 Code

 Conditional

 Container

 DDE

 Error Handler

 Loop

 Math

 Task

 Timer

 User-defined

Action Objekt

Das Action Objekt verhält sich wie eine Unterroutine und kann aus jeder Action List aufgerufen werden. Sie definieren beliebige Action Objekte durch eine Phrase und Parameter für dieses Objekt in dessen Einstellungen.

Analog/Digital (A/D) Objekt

Das A/D Objekt steuert analog/digitale Eingabe Hardware, mit der Erfassung eines oder mehrerer Datenkanäle bei spezifizierten Erfassungsraten.

Bar Indicator Objekt

Das Bar Objekte zeigt numerische Werte als farbigen Balken mit optionaler Skalierung an.

Bitmap Switch Objekt

Das Bitmap Switch Objekt erlaubt es Ihnen, eigene Schalter mit Hilfe von eigenen Grafiken zu definieren. Somit können Sie Taster, Schalter, Schieberegler und sogar Drehschalter definieren.

Case Objekt

Das Case Objekt führt eine von vielen alternativen Gruppen von Aktionen aufgrund eines definierbaren mathematischen Ausdrucks aus.

Code Objekt

Das Code Objekt gibt Ihnen die Möglichkeit, externe Routinen (DLLs), die in C, Pascal oder anderen Programmiersprachen geschrieben wurden, aus TestPoint auszuführen.

Conditional Objekt

Das Conditional Objekt führt eine von zwei alternativen Gruppen von Aktionen aufgrund eines definierbaren mathematischen Ausdrucks aus.

Container Objekt

Das Container Objekt stellt die Möglichkeit der temporären Zwischenspeicherung (wie auch andere Objekte) und die Akkumulierung der angefallenen Daten zur Verfügung.

Data Entry Objekt

Das Data-Entry Objekt erlaubt eine Benutzereingabe variabler Zahlenwerte oder Strings.

DDE Objekt

Das DDE Objekt unterstützt den dynamischen Datenaustausch (DDE). Dadurch ist es möglich, von TestPoint aus auf Daten anderer Windows-Anwendungen zuzugreifen und diese Anwendungen sogar zu steuern. Zu diesen Anwendungen gehören Tabellenkalkulationsprogramme, Datenbanken, Textverarbeitungen, Mathematikprogramme oder Grafikanwendungen.

Digital I/O (DIO) Objekt

Das DIO Objekt steuert digitale Ein-/Ausgabe Hardware (Einsteckkarten), die Intel 8255-kompatibel sind.

Digital/Analog (D/A) Objekt

Das D/A Objekt steuert digital/analog Ausgabe Hardware und erlaubt die Ausgabe von Gleichspannungen oder Kurvenzügen.

Display Objekt

Das Display Objekt kann numerische Werte oder Zeichenketten in wählbarer Größe, Farbe und Format darstellen.

Error Handler Objekt

Das Error-Handler Objekt kann benutzt werden, um auf Fehlermeldungen des Systems mit benutzerdefinierten Bedingungen zu reagieren oder eigene Fehlermeldungen zu erzeugen.

File Objekt

Das File Objekt erlaubt den Schreib- und Lesezugriff auf Dateien, einschließlich formatierter Ein-/Ausgabe einzelner Werte oder ganzer Vektoren oder Felder.

GPIB Objekt

Das GPIB Objekt steuert externe Meßgeräte, die über die GPIB (IEEE-488, HPIB, IEC-625) Schnittstelle mit dem PC verbunden sind.

Graph Objekt

Das Graph Objekt zeichnet Grafiken von Daten eines oder mehrerer Kurvenzüge im Y-n-Modus, X-Y-Modus, als Balken-, Linien- oder Linienschreiber-Diagramm.

Grid Objekt

Das Grid Objekt unterstützt die numerische Eingabe und Darstellung ganzer Vektoren oder Felder. (Für Tabellen mit Funktionen wie Tabellenkalkulationen benutzen Sie das VBX Objekt mit Produkten von Drittanbietern).

Gruppen von Objekten

Jedes Objekt kann in Gruppen plaziert werden. Gruppen (Groups) sind Werkzeuge zur besseren Organisation Ihrer Anwendung und erlauben den indizierten Zugriff auf mehrere Objekte als Feld.

Indicator Objekt

Das Indicator Objekt zeigt Ein/Aus Werte an. Dabei sind Farben und Texte frei wählbar.

Indirekte Objekte

Auf jedes Objekt kann man durch die indirekte Version desselben zugreifen, das durch ein Menükommando erzeugt wird. Indirekte Objekte verhalten sich wie Zeiger in konventionellen Programmiersprachen wie C oder Pascal.

 *Loop Objekt*

Das Loop Objekt wiederholt eine Gruppe von Aktionen, basierend auf einem vorgegebenen Wert; entweder schrittweise durch einen Bereich von Zahlen (For/Next) oder als Test auf eine bestimmte Bedingung (Do/Loop).

 *Math Objekt*

Das Math Objekt stellt Berechnungsfunktionen, von einfacher Arithmetik oder String-Funktionen, bis zu erweiterten Funktionen, wie FFT´s oder Filterung, zur Verfügung. Dabei stellt das Objekt eine konfigurierbare Formelzeile zur Verfügung, in der auch variable Parameter für die Berechnung definiert werden können.

OLE Objekt

Das OLE Objekt stellt das ***O****bject* ***L****inking and* ***E****mbedding* zur Verfügung, eine Windows Funktion, die es erlaubt, Informationen von anderen Programmen direkt in einem TestPoint Fenster darzustellen. Dabei können Grafiken, Multimedia Videos, Klänge oder Bilder dargestellt werden.

 *Panel Objekt*

Das Panel Objekt wird verwendet, um mehrere Fenster in einer Anwendung darzustellen und zu steuern.

 *Picture Objekt*

Das Picture Objekt kann Bilder aus einer Vielzahl von Formaten darstellen (BMP, PCX, TIF, usw.).

Port I/O Objekt

Das Port I/O Objekt steuert jede PC-Einsteckkarte auf Registerebene durch Schreib- und Lesezugriffe. Dies erlaubt die Verwendung fast aller PC-Einsteckkarten unter TestPoint.

Pushbutton Objekt

Das Pushbutton Objekt ist ein Standard Windows Schalter, der benutzt werden kann, um Action Listen zu starten.

Report Objekt

Das Report Objekt erzeugt Berichte auf dem Drucker. Diese können Bilder, Texte, Grafiken und Formatierungen enthalten. Seitennumerierung, frei wählbare Schriftart und Größe, Firmenlogos usw. können benutzt werden.

RS232 Objekt

Das RS232 Objekt steuert die an die seriellen Schnittstellen des PC´s angeschlossenen Geräte. Diese können sowohl RS-232, als auch RS-422 oder RS-485 Standards benutzen.

Selector Objekt

Das Selector Objekt läßt den Anwender aus einer Vielzahl von Möglichkeiten eine auswählen.

Slider Objekt

Das Slider Objekt stellt dem Anwender eine komfortable Eingabemöglichkeit für numerische Werte durch einen Schieberegler zur Verfügung.

Switch Objekt

Das Switch Objekt läßt den Anwender Ein/Aus Werte eingeben. Dabei sind die Texte frei wählbar.

Task Objekt

Das Task Objekt kann die Multitasking Fähigkeiten von TestPoint steuern und Initialisierungen und Abbrüche vornehmen.

Text Objekt

Das Text Objekt kann benutzt werden, um beliebige Texte in beliebiger Farbe, Schriftart und Größe am Bildschirm darzustellen.

Time Objekt

Das Time Objekt beinhaltet die jeweils aktuelle Tageszeit und verschiedene Möglichkeiten, um Aktionen periodisch, bei gegebener Zeitbasis, auszuführen.

User-defined Objekte

User-defined (Benutzerdefinierte) Objekte können mit TestPoint erzeugt werden, um damit wiederkehrende Vorgänge (wie z. B. GPIB-Geräteeinstellungen) zu automatisieren.

VBX Objekt

Das VBX Objekt erlaubt es, dem TestPoint System neue Objekte hinzuzufügen durch das Laden von Visual Basic Extenstions (VBX-Dateien). VBX sind Software-Komponenten, die von vielen Anbietern zur Verfügung gestellt werden. Ursprünglich wurden VBX als Erweiterung zum Microsoft Visual Basic verstanden, können nun aber mit TestPoint, wie „Originale“ Objekte verwendet werden. Typische VBX stellen Knöpfe, Schieberegler, Schalter, Balkengrafiken, 3-D Steuerelemente, Tabellenfunktionen, usw. zur Verfügung.

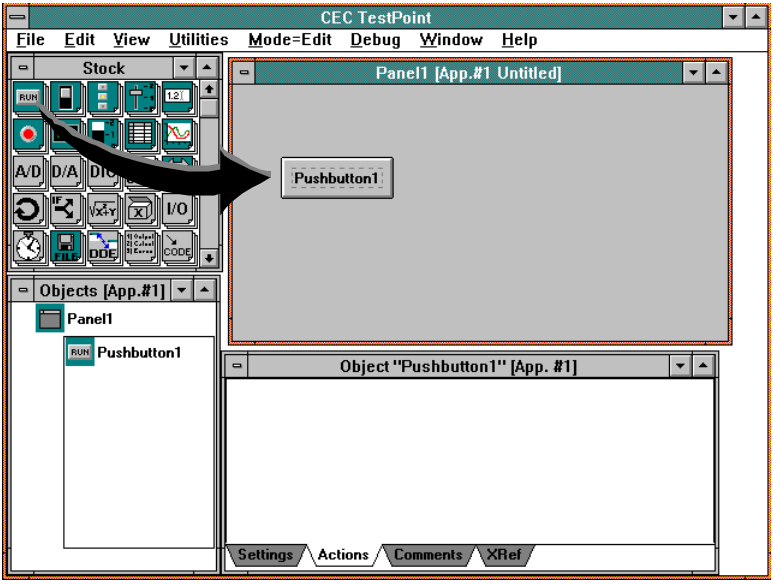
Kapitel 3. Editieren

Was in diesem Kapitel behandelt wird:

- Die TestPoint Benutzeroberfläche: Das Ein-/Ausgabefenster (Panel), das Objektfenster (Stock), die Objects List (Objekts) und die Objekt-Detailfenster.
- Techniken zur Anwendungserstellung.
- Die TestPoint Menükommandos.

Ziehen und Ablegen

Maßgeblich für die Entwicklung von TestPoint Anwendungen ist die Ziehen und Ablegen Funktion:



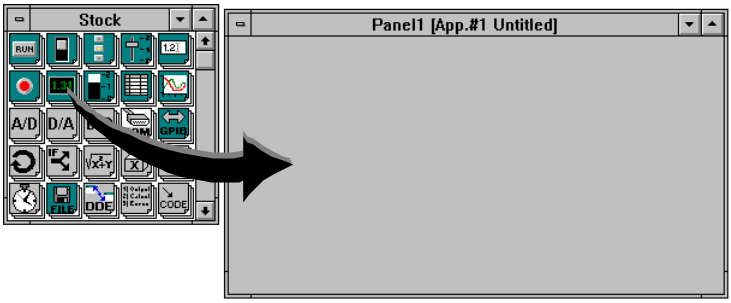
Wie man die Funktion „Ziehen und Ablegen“ anwendet

- Beginnen Sie damit, daß Sie den Mauszeiger auf das Objekt bewegen, das Sie verwenden wollen, wie z. B. ein Objekt aus dem Stock Fenster.
- Drücken Sie dann die linke Maustaste und halten sie sie gedrückt.
- Während Sie die Taste festhalten, bewegen Sie die Maus zum Zielort, wo Sie das Objekt plazieren wollen.

- Lassen Sie die Maustaste los.

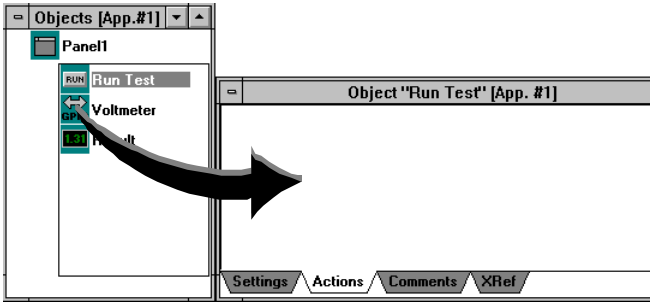
Wo sind Drag and Drop-Funktionen möglich

Vom Stock ins Panel oder Objects Fenster



Ziehen Sie ein Symbol für ein Objekt vom Stock Fenster ins Panel, um ein neues Objekt verwenden zu können.

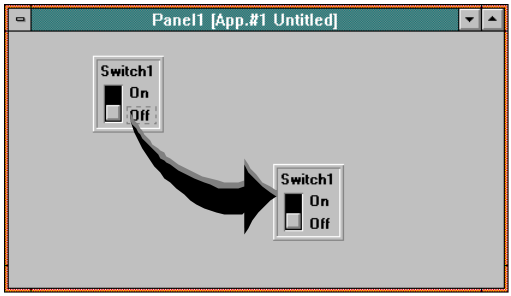
Vom Panel oder Objects Fenster in die Action List



Ziehen Sie das Symbol für ein Objekt in das Action List Fenster. Wenn Sie von dem Objects Fenster aus arbeiten, sollten Sie das Symbol anklicken, nicht den Namen des Objektes.

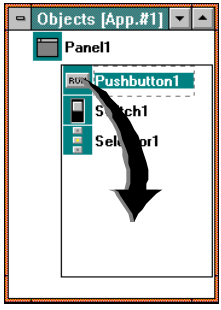
+ Nicht jedes Objekt (z. B. Loop, Math) hat auf dem Panel ein Icon. Diese Objekte werden nur im Objekt Fenster der Anwendung angezeigt.

Innerhalb eines Panels



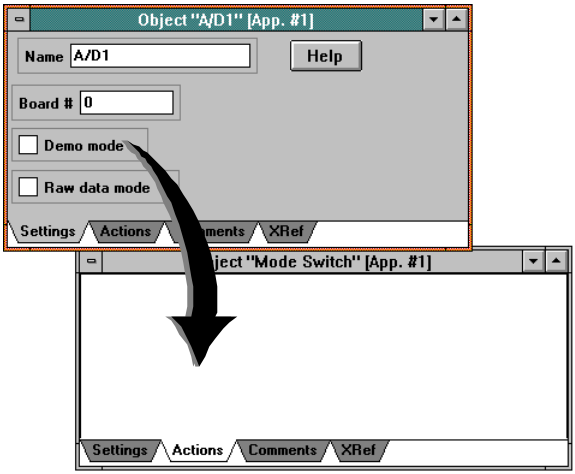
Um Objekte innerhalb eines Panels zu bewegen, ziehen Sie es einfach.

Innerhalb des Objects Fensters



Die Reihenfolge der Objekte im Objects Fenster kann durch Ziehen der Objekte an eine andere Stelle verändert werden. In welcher Reihenfolge die Objekte aufgelistet sind, ist für die Ausführung der Anwendung unerheblich.

Von Settings (Einstellungen) in die Action List



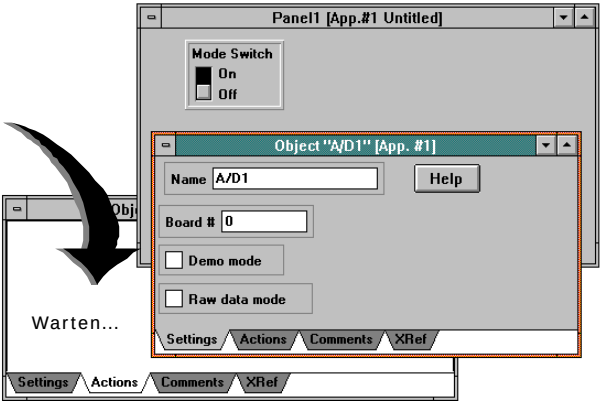
Einstellungen (Settings) der Objekte können in Action Listen verändert werden, indem Sie die entsprechende Einstellung vom Settings Fenster ins Action List Fenster ziehen, wie oben dargestellt.

Kopieren, Ausschneiden und Einfügen

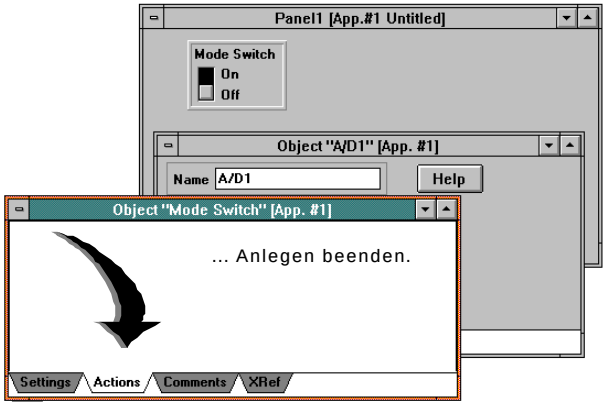
Zusätzlich zur „Ziehen und Ablegen“ Funktion können Sie die Windows Funktionen Kopieren, Ausschneiden und Einfügen (Copy, Cut und Paste) benutzen, um Objekte zu duplizieren. Zuerst wählen Sie die gewünschten Objekte durch anklicken aus. Dann verwenden Sie aus dem Menükommando Edit einen der Befehle Copy/Cut/Paste aus.

Wie Sie ins richtige Fenster gelangen

Was wollen Sie machen, wenn das Fenster, in dem Sie ein Objekt ablegen wollen, teilweise verdeckt ist? Sie können das Objekt nicht an der richtigen Stelle plazieren? Halten Sie den Mauszeiger einfach einen Augenblick ruhig über dem gewünschten Fenster und dieses kommt automatisch in den Vordergrund.



...dann...

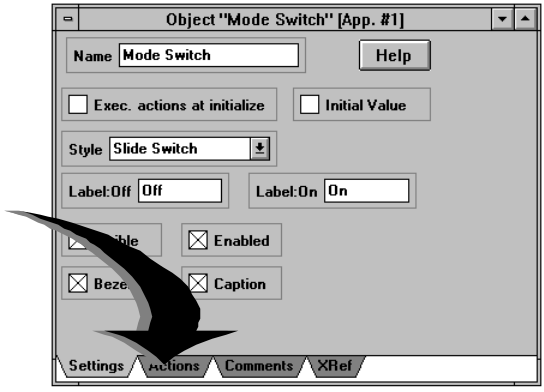


Die voreingestellte Wartezeit für diese Prozedur ist 1 Sekunde. Sie kann als Einstellung in der TESTPT.INI Datei verändert werden.

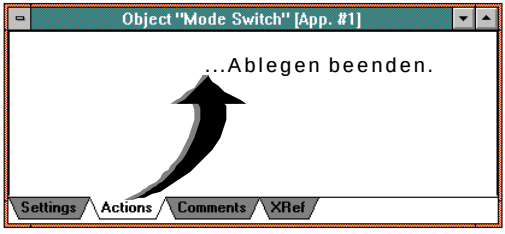
Zwischen Fenstern mit Karteireitern schalten

Normalerweise können Sie zwischen Einstellungen, Aktionen und Kommentaren schalten, indem Sie auf die Karteireiter am unteren Fensterrand des Objekt Detailfensters klicken.

Während einer Ziehen und Ablegen Operation können Sie jedoch nicht auf diesen Karteireiter klicken. Warten Sie mit dem Mauszeiger einfach einen Augenblick über dem gewünschten Karteireiter und das Fenster wechselt automatisch:

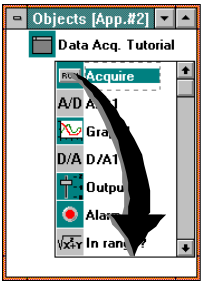


Warten . . . dann

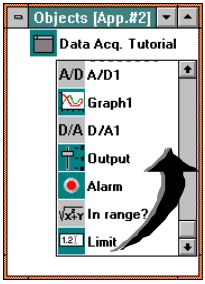


Ziehen und rollen

Um im Objects Fenster oder der Action List auf- und abwärts zu rollen, warten Sie einfach mit dem Mauszeiger in der Nähe des Fensterrandes und der Inhalt rollt automatisch nach oben oder unten.



Warten am Rand des Fensters...



auf das Rollen warten,

Mauszeiger vom Rand weg bewegen und...

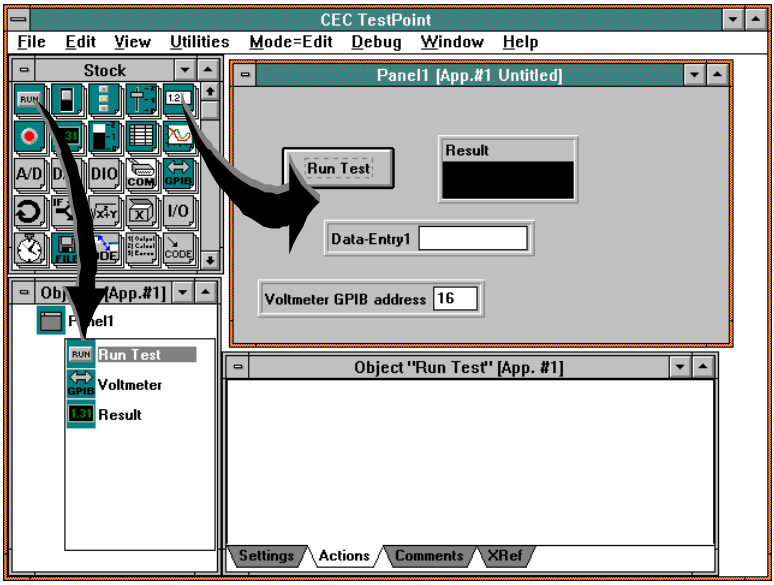
...das Ablegen beenden.

Beides, die Wartezeit und Wiederholrate beim Rollen kann durch Einstellungen in der TESTPT.INI Datei eingestellt werden.

Stock Fenster

Das Stock Fenster ist die Quelle für die Objekte, die in einer Anwendung durch die „Ziehen und Ablegen“ Funktion verwendet werden.

Jeder notizblockartige Stapel im Stock Bereich, zeigt ein Symbol für die Kategorie eines Objektes. Das Stock Fenster startet mit einer voreingestellten Größe, die verändert werden kann, um mehr Zeilen von Objekten gleichzeitig sichtbar zu machen. Die Anzahl der Spalten ist fest vorgegeben. Wenn nicht alle Objekte sichtbar sind, erscheint ein Rollbalken an der rechten Seite des Fensters, mit dem man die weiteren Objekte sichtbar machen kann.



Hilfe für Stock Objekte

Zur Identifizierung eines Symbols im Stock, erscheint, durch Klicken der rechten Maustaste auf dem Objekt, ein Text zur Erklärung dieses Objekts.

Modifzieren des Stocks

TestPoint startet mit einem voreingestelltem Aufbau des Stock Fensters. Durch Drücken und Festhalten der Ctrl- (oder Strg-) und der Maustaste können Objekte angewählt und verschoben werden. Ebenso kann ein Objekt auf einen freien Platz verschoben werden. Wird es auf ein anderes Objekt bewegt, tauschen beide Ihre Plätze.

Es können zusätzlich Objekte zum Stock **hinzugefügt** werden. Dies ist nützlich, wenn ein spezielles Objekt mit verschiedenen Einstellungen verwendet werden soll. Ein solches neu definiertes Objekt kann aus mehreren Einzelobjekten zusammengefügt und unter einem neuen Symbol abgelegt werden (siehe 'User-defined Objects - benutzerdefinierte Objekte'). Sie bewegen dazu das neu erstellte Symbol aus der Objects List (Objects) in den „Stock". Vor dem Ablegen drücken Sie die Strg - (Ctrl-) Taste auf der Tastatur.

+ **Sollen Objekte in den Stock übernommen werden, nutzen Sie einen freien Platz im Stock Window. (Verschieben Sie es auf ein bereits vorhandenes Objekt, wird dieses durch das neue ersetzt). Für das Erstellen einer neuen Reihe bewegen Sie den Scroll Bar abwärts und bringen Sie ein neues Objekt auf die Grundlinie.**

Laden und Speichern des Stock

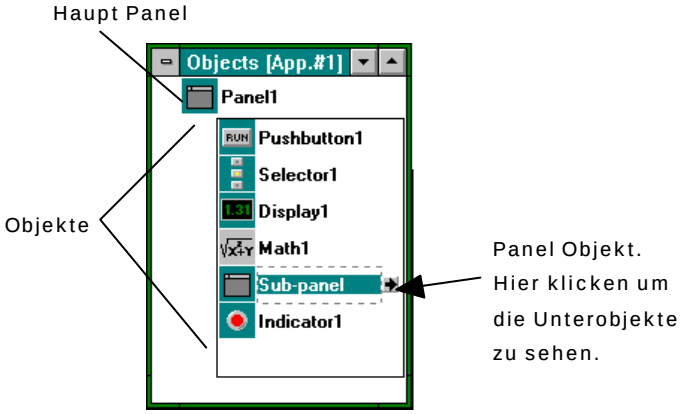
TestPoint speichert die Grundeinstellung des 'Stock' im File TPEDIT.STK und legt eine read-only Kopie im File ORIGINAL.STK an. Wenn TestPoint startet, lädt es TPEDIT.STK, wird TestPoint verlassen , speichert es den aktuellen Stock in der Datei.

Es können viele Stock Konfigurationen in mehreren Files gespeichert sein, weil der Stock, wie bereits beschrieben, vom Anwender verändert werden kann. Die Menübefehle der 'Utilities' 'Save Stock' und 'Load Stock' erlauben die Veränderung der Stock Dateien. Der aktuelle Stock

wird beim Verlassen von TestPoint immer in TPEDIT.STK abgespeichert und beim erneuten Start wieder verwendet.

Objects Fenster

Das Objects Fenster zeigt das Symbol eines Arbeitsfensters (Panels) und die Unterobjekte, die in diesem enthalten sind:



Der obere Teil der Objects List zeigt das Hauptarbeitsfenster der Anwendung (Panel1). Enthält eine Applikation weitere Panels, wird eine Folge von Panels und deren Unterobjekten gebildet. In die Objects Listen der Unter-Panel gelangt man durch Anklicken des neben dem Panelnamen befindlichen Pfeiles oder durch Markieren des Symbols und einer anschließenden Wahl des Menüpunktes ‘VIEW’ ‘Objects’. Um in das Hauptarbeitsfenster zurückzugelangen, klickt man auf den links neben dem Symbol des Unterpanels erscheinenden Pfeil.

Ziehen und Ablegen

Objekte in der Objects List können durch Anwählen mittels des Mauszeigers auf das **Symbol** des Objekts und bei gedrückter linker Maustaste bewegt (gezogen) werden. Wird ein Objekt auf diese Weise in einer Action Liste abgelegt (Maustaste loslassen), kann es ein Parameterfeld füllen oder eine neue Aktionszeile erstellen.

Durch das Ziehen und Ablegen der Objekte in der Objects List kann deren Reihenfolge innerhalb des Fensters geändert werden. Die Reihenfolge ist nicht relevant und dient nur zur besseren Übersicht der verwendeten Objekte.

Auswählen

Objekte im Objects Fenster können ausgewählt werden zum Kopieren, Verschieben oder Löschen der Objekte. Ein ausgewähltes Objekt wird durch invertiertem Text dargestellt.

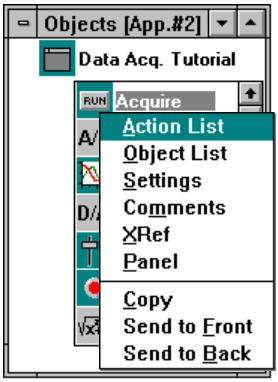
Ein einzelnes Objekt kann durch Anklicken des **Namens** oder des **Symbols** ausgewählt werden.

Um mehrere Objekte auszuwählen, klicken Sie die mit der linken Maustaste über dem Namen eines Objektes und ziehen den Zeiger über den gewünschten Bereich. Sollten die Objekte nicht hintereinander stehen, können Sie diese auswählen, indem Sie die Strg- (Ctrl-)Taste während der Auswahl gedrückt halten.

Action Listen und andere Objekt Informationen

Jedes Objekt kann Einstellungen, eine Action Liste, und Kommentare enthalten.

Die Anwahl eines Objektsymbols mit der rechten Maustaste öffnet ein Popup-Menü mit den möglichen Funktionen dieses Objektes:

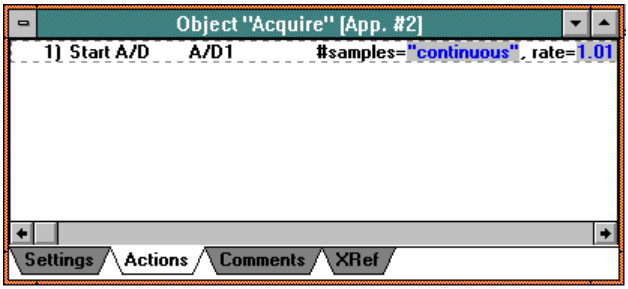


Die Wahl ‘Action-List’ ist hell markiert, so daß es ohne Mausbewegung ausgewählt werden kann.

+ Zur Anzeige der zugehörigen Action-List eines Objektes klicken Sie das Icon mit der rechten Maustaste an.

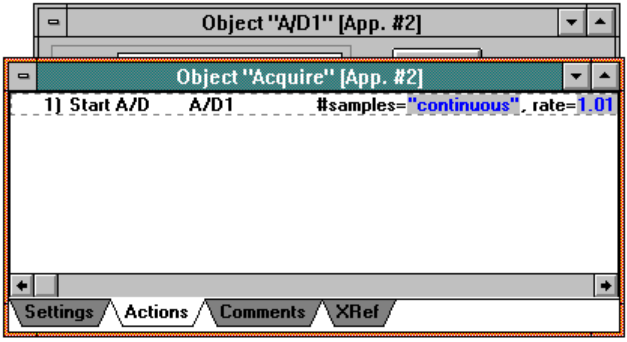
Objekt Detail Fenster

Individuelle Objekt Fenster erscheinen, wenn neue Objekte verwendet werden oder wenn die „Settings“, Action-Liste, usw., über die Menükommandos oder die rechte Maustaste ausgewählt werden.



Die Kartenreiter am unteren Fensterrand erlauben das Umschalten der einzelnen Möglichkeiten der Objektansicht: Settings, Action List, Comments oder Cross Referenz. (Wenn ein Objekt keine Action List besitzt, ist der Kartenreiter grau hinterlegt und kann nicht verwendet werden)

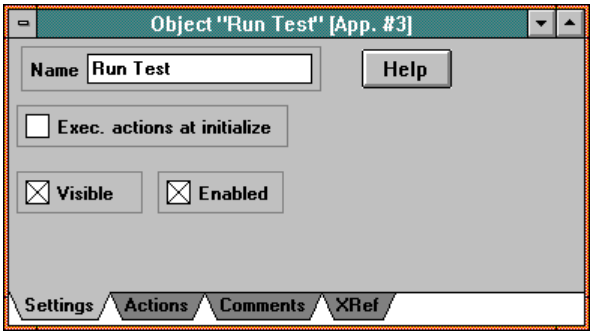
Es können mehrere Object Detail Fenster gleichzeitig geöffnet sein. Dies ist hilfreich bei der Ansicht oder beim Vergleich verschiedener Action Listen. Die Möglichkeit, Einstellungen von einem Fenster ins andere zu übernehmen, erhöht den Bedienkomfort.



Die maximale Anzahl dieser Fenster ist normalerweise auf zwei begrenzt (zuzüglich einem temporären für ein weiteres Settings Fenster). Diese Anzahl kann jedoch verändert werden, um Ihren Bedürfnissen gerecht zu werden. Dazu wird der entsprechende Parameter (MaxObjectWindows=n) in der TESTPT.INI Datei angepaßt.

Einstellungen

Das Settings Fenster erscheint automatisch, wenn ein neues Objekt verwendet wird oder ein Doppelklick auf ein bestehendes ausgeführt wird.



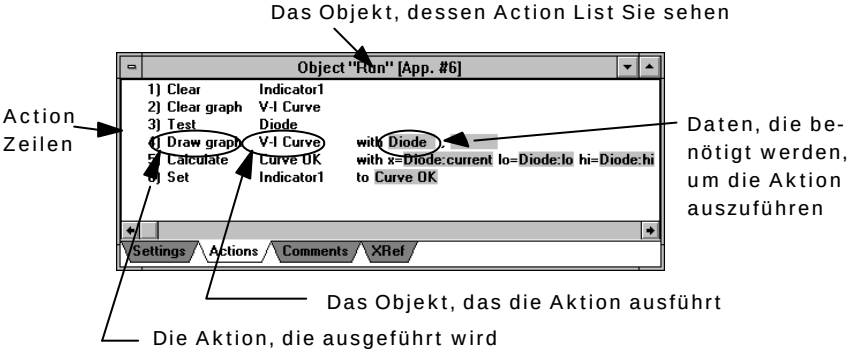
Die Einstellungen im Settings Fenster erlauben es, die Objekte an Ihre Anwendung anzupassen.

Die Einstellungen können im Editor geändert, in die Action List übernommen und gespeichert werden. Während der Ausführung der Anwendung kann man keine "Settings" in die "Action List" übernehmen.

Action Liste

Die Action-List eines ausgewählten Objektes zeigt die durch das Objekt abzuarbeitende Befehlsfolge. Die Action Liste wird durch die Auswahl eines Objektes in der Objects List oder dem Arbeitsfenster und dem sich anschließenden Menübefehl ‘VIEW’ ‘Action-List’ (oder auch Ctrl A bzw. Strg A) dargestellt.

+ Der einfachste Weg zur Darstellung der Action Liste eines Objektes ist, das Objekt selbst mit der rechten Maustaste anzuwählen.



Gehen auszuführende Aktionen über den rechten Rand der Action Liste hinaus, erscheint ein Rollbalken am unteren Rand, mit dem auch diese Informationen noch angezeigt werden können.

Parameter

Schattierte Bereiche in der Action Liste zeigen an, daß hier Parameter eingegeben werden können. Durch Anwahl dieses schattierten Bereiches und Eingabe eines Wertes kann dieser Parameter gesetzt werden. Die Eingabe wird durch Drücken der ENTER- oder Eingabetaste oder durch Klicken mit der Maus an eine beliebige Stelle abgeschlossen.. Ein Parameter kann auch ein Objekt aus der Objects List, einem Arbeitsfenster oder eine Objekt Referenz einer anderen Stelle der Action Liste sein.

Objekt Referenzen

Werden Objekte in das schattierte Parameterfeld gezogen und dort abgelegt, so wird der Datenwert des Objektes zu einem Parameter. Der Name des Objektes, welches die Action Liste ausführt, ist ebenfalls eine Objektreferenz, welche verschoben werden kann. Objekt Referenzen können auf Wunsch auch den Datentyp spezifizieren. Der Zugriff auf Datentypen geschieht durch einen Doppelklick auf das Objekt. Ist die Aktion eine Ein-/Ausgabe Operation, kann auch das Datenformat bestimmt werden.

Konstanten als Parameter

Wenn Sie auf das schattierte Parameterfeld in der Action List klicken, können Sie den Wert einer Konstanten eingeben. Wie erkennen Sie, ob die Konstante ein String oder eine Zahl ist?

Numerische Konstanten

Sollten Sie eine Zahl, bestehend aus Ziffern mit Dezimalpunkt, Vorzeichen und Exponenten eingeben, erhalten Sie eine numerische Konstante. Zum Beispiel:

34.56
+3.45E1

Hexadezimale Konstanten

Um einen hexadezimalen Wert zu spezifizieren, geben Sie den Wert, gefolgt von einem „H“, ein. Groß- oder Kleinschreibung brauchen Sie nicht zu beachten. Der Hexadezimalwert muß immer mit einer Ziffer beginnen! Beispiel:

```
1C3H
0f7baH
```

Falls die oben genannten Bedingungen nicht zutreffen, handelt es sich um einen String. Auch, wenn er mit einer Zahl beginnt, solange er sich nicht im korrekten Format für eine Zahl befindet. Beispiel:

```
ABC
0,1,2
```

Wenn Sie einen String wünschen, können Sie diesen durch Anführungszeichen definieren:

```
"  34  "
```

Wenn ein Konstanten Parameter auf dem Bildschirm oder Drucker dargestellt wird, ist er in Anführungszeichen eingeschlossen, wenn es sich um einen String handelt. Die Anführungszeichen sind kein Bestandteil der Konstanten (Anführungszeichen innerhalb des Strings werden als zwei Anführungszeichen hintereinander dargestellt).

Spezielle Zeichen Codes

Soll ein String nicht druckbare Sonderzeichen enthalten, wie Carriage Return, Line Feed, Tab oder Andere, dann muß ein 'Backslash' gefolgt von einem x und dem zweistelligen hexadezimalen Code eingegeben werden. Spezialzeichen sind **\r** für Return, **\n** für Line Feed (neue Zeile) und **\t** für tab:

```
"Hier endet die Eingabe mit Return und Line Feed\r\n"
"So geht es\x0D\x0A"
"Und dies hat einen tab\t."
"Der String enthält ein Sonderzeichen\x03."."
```

Achten Sie auf unbeabsichtigte Sonderzeichen, wenn Sie einen DOS Pfad- oder Dateinamen eingeben. Benutzen Sie Großbuchstaben, um dieses Problem zu umgehen:

```
"c:\testpt\bad"          wird zu "c:\x09estpt\bad"

"C:\TESTPT\OK"           bleibt
```

Sie können diese Spezialzeichen auch als eigenständige Zeichen, außerhalb eines Strings angeben:

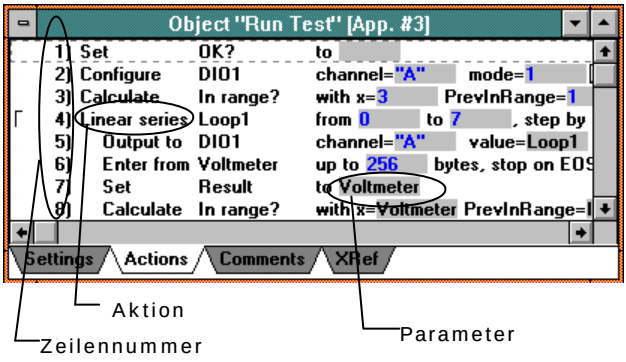
```
CR
LF
CRLF
TAB
```

Konstanten editieren

Zum Editieren einer Konstanten als Parameter, ohne alles neu einzugeben, wählen Sie ihn mit der Maus an und drücken ENTER. Dieses Feld wird dann für die Änderungen hell hinterlegt. Mit den Pfeil-, Pos1- oder Ende-Tasten können Sie gezielt an den erforderlichen Stellen Ihre Änderungen vornehmen.

Ziehen und Ablegen und Auswählen

Es gibt verschiedene Abschnitte der Action Liste, die durch Spaltenaufteilung gekennzeichnet sind. Unterschiede im Verhalten gibt es auch zwischen Mausbedienung und Tastaturbedienung.



Reihenfolge ändern

Ziehen der **Zeilennummer** einer Befehlszeile und Ablegen an einem anderen Punkt innerhalb der 'Action-List', bewegt diese Zeile an eine andere Position. Die Ausführungsreihenfolge wird dadurch verändert..

Zeilen auswählen

Das Klicken oder Ziehen beim **Action Name** direkt nach der Zeilennummer ermöglicht es, diese zu löschen, zu verändern, zu kopieren oder zu speichern.

Parameter auswählen

Anwählen eines schattierten Bereiches (**Parameter)** bestimmt die Referenz dieses Objektes. Diese kann ebenfalls gelöscht, verändert, kopiert oder gespeichert werden. Die Eingabe über die Tastatur ergibt eine Konstante als Parameter.

+ Beachten Sie, daß die Eingabe eines Parameters per Tastatur nicht dasselbe wie das Ziehen eines Objektes dorthin ist. Der Objektname bildet eine Verknüpfung zu den entsprechenden Daten innerhalb dieses Objektes. Auch die Eingabe des Objektnamen ist nur ein einfacher String und hat weiter keine Beziehung zu diesem Objekt.

Kommentare

Kommentare zu einem Objekt können durch das View/Comments Menükommando oder durch Klicken auf den Comments Tabulator am Objekt Detail Fenster angezeigt werden.

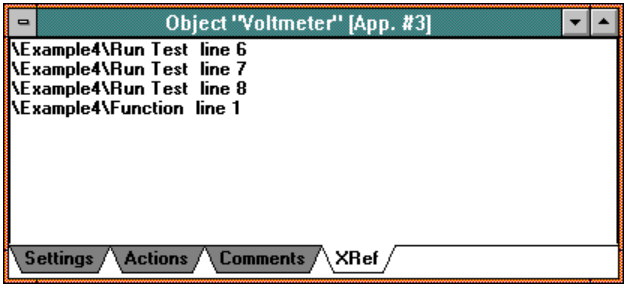
Es wird pro Objekt ein Kommentar gespeichert. Um diesen Kommentar einzugeben oder zu verändern, lassen Sie sich das Comments Fenster anzeigen und geben Sie dort den Text ein. Dieser Text kann maximal 32768 Zeichen beinhalten.

Kommentare können verwendet werden, um Ihre Anwendung zu dokumentieren oder die Funktion und Arbeitsweise eines Objektes zu beschreiben.

Die Kommentare können, zusammen mit dem Rest der Anwendungsbeschreibung (Listing), ausgedruckt werden.

Cross-Referenz (XRef)

Das Cross Referenz Fenster zeigt alle Actionslistenzeilen an, in denen das Objekt verwendet wurde:



Dies kann hilfreich beim Lesen der Anwendung oder bei der Fehlerbeseitigung sein.

+ Durch Doppelklick auf eine Zeile im XRef Fenster können Sie zu der entsprechenden Zeile in einer Action Liste springen.

Panel Fenster

Im Panel Fenster sieht der Anwender seine komplette Anwendung (User Interface).

TestPoint kann mehrere Panel Windows einer oder mehrerer Applikationen zur selben Zeit öffnen. Sind Panel Fenster durch andere Fenster verdeckt, kann über das Window Menükommando auf diese zugegriffen werden.

Ein Unter-Panel Fenster zur Anzeige der Auswahl der Objekte im Object Window einer Anwendung kann über das Menükommando View/Panel ausgewählt werden. Panel Fenster können durch Doppelklick der oberen linken Ecke des Windows geschlossen werden. Das Schließen des Haupt-Panel Window einer Applikation ist das gleiche wie die Wahl des File/Close Menükommandos.

Panel Windows können außerdem durch Aktionen in der Action List erscheinen oder verschwinden. Das Verdecken eines Panel Window ist jedoch nicht das gleiche, wie es zu schließen.

Edit und Run Modus

Die Verwendung der Maus im Panel Fenster wird durch die Einstellungen im Modus Menü in TestPoint bestimmt. Im Editier-Modus können die Objekte im Panel mit der Maus gelöscht, verändert, kopiert, verschoben oder gespeichert werden. Außerdem kann mit der Maus ein komplettes lauffähiges Programm gestartet werden, wenn das Menükommando Mode=Run angeklickt wird und das Ereignis zum Abarbeiten einer Action Liste gegeben ist. Dies kann z. B. durch Klicken auf einen Schalter (Pushbutton) erfolgen.

Die Objekte im Panel Fenster können angewählt und bewegt werden. Sie können aus dem Panel Fenster in die Action Liste und auch umgekehrt gezogen werden.

Auswählen von Objekten

Ist der Editier-Modus eingestellt, so kann ein Objekt durch Anklicken mit der Maus im Arbeitsfenster angewählt werden. Ein editiertes Objekt ist durch eine dicke Umrißlinie gekennzeichnet. Anschließend kann, wie oben erwähnt, dieses Objekt vergrößert, verkleinert, gelöscht, kopiert, eingefügt oder verschoben werden.

Um mehrere Objekte zu selektieren, muß die Strg-Taste gedrückt gehalten werden und mit der linken Maustaste die entsprechenden Objekte angeklickt werden.

Um die Auswahl der Objekte zu widerrufen, muß nur in das Fenster geklickt werden.

Ausrichten am Raster

Der einfachste Weg, Objekte auf dem Panel Fenster gleichmäßig auszurichten ist, das Menükommando Edit/Snap to grid auszuwählen.

Der Rasterabstand kann durch das Menükommando Edit/Grid settings eingestellt werden.

Wenn „Snap to Grid“ aktiv ist, werden die Objekte im Panel Fenster immer auf das nächste Rastergitter plaziert. Die Größe des Objektes wird ebenfalls an ein Vielfaches dieses Rasters angepaßt.

Ausrichten von Objekten

Da das Ausrichten der Objekte auf dem Panel mit der Maus schwierig und zeitaufwendig ist, wurde eigens dafür im Hauptmenü unter der Option ‘Edit’ die Funktion ‘**Align’** entwickelt.

Zuerst müssen die auszurichtenden Objekte mittels der Umschalttaste und der Maus selektiert werden. Anschließend kann mittels des ‘Align’ Menükommandos unter folgenden Möglichkeiten gewählt werden.

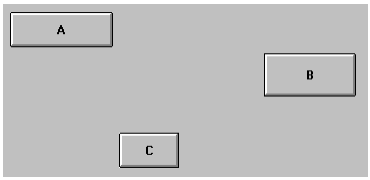
Left	Richtet alle selektierten Objekte nach dem am weitesten **links** stehenden Objekt aus.
Right	Richtet alle selektierten Objekte nach dem am weitesten **rechts** stehenden Objekt aus.
Top	Richtet alle selektierten Objekte nach dem am weitesten **oben** stehenden Objekt aus.
Bottom	Richtet alle selektierten Objekte nach dem am weitesten **unten** stehenden Objekt aus.
Center (H)	Richtet alle selektierten Objekte auf einer **Senkrechten** in der **Mitte** des Arbeitsfensters aus.
Center (V)	Richtet alle selektierten Objekte auf einer **Waagerechten** in der **Mitte** des Arbeitsfensters aus.
Evenly spaced (H)	Richtet alle selektierten Objekten mit einem **gleichbleibenden horizontalem Abstand** aus.
Evenly spaced (V)	Richtet alle selektierten Objekten mit einem **gleichbleibenden vertikalem Abstand** aus.
Size...	Verändert die **Größe** aller selektierter Objekte auf die Größe des schmalsten, breitesten, höchsten oder flachsten.

Die Strg-Taste und die Cursor Tasten können als Kurztastenkombination für die Ausrichtung benutzt werden.

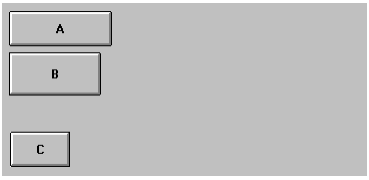
Die Alt-Taste und die Cursor Tasten können als Kurztastenkombination für die Größe benutzt werden.

Die aufgeführten Beispiele zeigen Effekte der Align-Funktion.

Original:

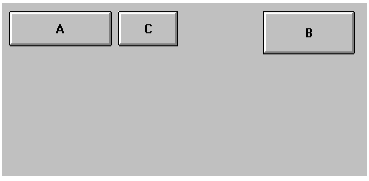


Left:

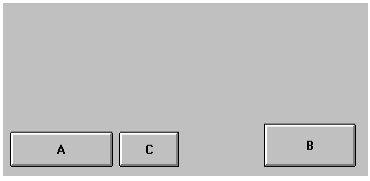


Right:

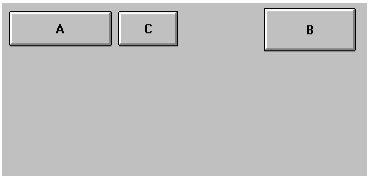
Top:



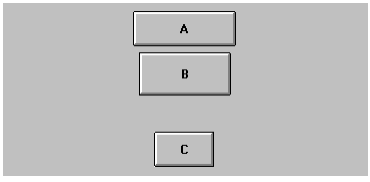
Bottom:



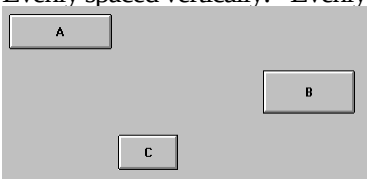
Center Vertically:



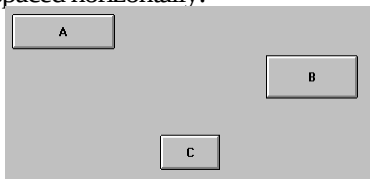
Center Horizontally:



Evenly spaced vertically:



Evenly spaced horizontally:



File	Das File Menü enthält Kommandos, die mit dem Zugriff auf Dateien zusammenhängen.
New	Neubeginn einer leeren, ganz neuen Applikation. Es wird ein neues Panel geöffnet.
Open	Öffnen eines bereits existierenden Programmes
Close	Schließen der aktuellen Applikation. Sie werden gewarnt, wenn die Applikation geändert wurde.
Save	Speichert die aktuelle Anwendung ab.
Save As	Speichert die aktuelle Applikation unter einem spezifischen Programmnamen ab. Das ist sinnvoll, um eine bereits existierende Anwendung unter einem neuen Namen abzuspeichern.

Save with password	Speichert die aktuelle Anwendung mit einem Paßwort-schutz ab. Das Paßwort wird automatisch in der TESTPT.INI Datei mit abgespeichert, sodaß Sie nicht jedesmal beim Laden das Paßwort eingeben müssen. Wenn Sie jedoch die Anwendung weitergeben, ist diese ohne Paßwort nicht zu öffnen.
Save All	Alle offenen Applikationen werden sofort gespeichert.
Set startup bitmap	Spezifiziert ein Bild, das während des Ladens einer Runtime-Version (außerhalb der TestPoint Entwicklungsumgebung) dargestellt wird. Nachdem Sie das Bild ausgewählt haben, müssen Sie die Anwendung abspeichern.

Print	Ausdruck der aktuellen Applikation. Das Dialog Fenster gibt Ihnen eine Auswahl zum Ausdrucken Ihrer Applikation (all), oder nur der aktuellen Objects List (Es werden keine Inhalte anderer Fenster gedruckt). Außerdem stehen zur Auswahl beim Drucken: Fenster (Grafikausdruck), Objekte (Symbole im Fenster), Einstellungen, Action Listen , 'Data Reference' Einzelheiten (Art und Format), Kommentare, und der Inhalt der User-Defined Objekte.
Print Setup	Drucker Optionen auswählen.
Print Object	Druckt das aktuell ausgewählte Objekt. Dieses Menükommando kann verwendet werden, um Grafiken, Tabellen oder auch nur Texte auszudrucken.
Print Panel	Druckt den augenblicklichen Zustand des Panel Fensters.
Exit	Verlassen der TestPoint Entwicklungsumgebung. Es erscheint ein Hinweis, offene und modifizierte Programme zu sichern.

Edit	Das Edit Menü enthält Kommandos zum Verändern der Elemente Ihrer Anwendungen.
Cut	Schneidet den markierten Bereich aus und kopiert ihn in die Windows Zwischenablage
Copy	Erstellt eine Kopie und speichert sie in der Windows Zwischenablage. Es gilt dabei das aktive, hell markierte Fenster als Ausgangspunkt.
Paste	Liest den Inhalt aus der Windows Zwischenablage und fügt ihn im aktiven Fenster ein.
Paste Link	Erstellt einen DDE "hot-link" zwischen der aktuellen Zwischenablage und TestPoint. Wenn beispielsweise eine Tabelle in der Zwischenablage steht, wird, sobald sich die Tabelle ändert, das TestPoint Element, das zur Anzeige benutzt, wird aktualisiert.
Cancel Link	Hebt alle Verbindungen des ausgewählten Objekt auf.
Delete	Löscht alle ausgewählten Funktionen, **ohne** sie in die Zwischenablage zu kopieren.
Group	Faßt ausgewählte Objekte in Gruppen zusammen. Diese Gruppe wird zusammen bewegt (auf dem Panel verschoben) und kann in der Action List individuell oder als Gruppe (indiziert) verwendet werden.

Ungroup	Splittet eine ausgewählte Gruppe in einzelne Objekte. Das Symbol für die Gruppe im Objects Fenster wird ersetzt durch die Objekte, die in der Gruppe vorhanden sind.
Disable action lines	Deaktiviert eine oder mehrere markierte Zeilen in Action Listen. Diese werden bei der Ausführung übersprungen. Deaktivierte Zeilen werden durch einen grauen Text und ein vorangestelltes „#“ Zeichen dargestellt. Sie müssen eine oder mehrere Zeilen in der Action List auswählen, bevor dieses Menükommando verfügbar ist.
Enable action lines	Re-Aktivieren von Zeilen in Action Listen. Siehe oben.
Send to Front	Steuert die Sichtbarkeit vom Anwender erstellter Objekte im Arbeitsfenster. Dieses Kommando bringt das aktuelle Objekt in den Vordergrund, auch wenn es dann andere Funktionen verdeckt.
Send to Back	Dieses Kommando schiebt das aktuelle Objekt in den Hintergrund. Diese Menüoption hat auch auf die TAB Funktion einen Einfluß. Zur Auswahl einer TAB Reihe wählen Sie bitte die erste Funktion aus, bringen Sie anschließend in den Hintergrund und wiederholen sie diesen Vorgang für jede Funktion.

Snap to Grid	Einschalten der Raster Funktion für das Plazieren und die Größe von Objekten. Ein Haken erscheint, wenn Sie diese Funktion aktivieren.
Grid settings	Verändert den x- und y-Abstand des Rasters. Siehe oben.
Align	Besteht aus einem Unter-Menü mit den Kommandos, um die Objekte im Arbeitsfenster auszurichten. Siehe auch den Abschnitt zum Arbeitsfenster.

View	Das View Menü wählt aus, was in der TestPoint Entwicklungsumgebung sichtbar ist.
Settings	Öffnet ein Settings Fenster des angewählten Objektes.
Action List	Zeigt die Action Liste des ausgewählten Objektes.
Panel	Öffnen bzw. Umschalten zum Panel Fenster mit dem ausgewählten Objekt.
Object List	Umschalten der Objects Fensters der ausgewählten Objekte. Prinzipiell wird ein Panel Fenster mit diesem Befehl ausgewählt.
Comments	Zeigt die Kommentare des ausgewählten Objekts.
XRef	Zeigt das Cross Referenz Fenster des ausgewählten Objekts.
Data	Anzeige der in den Objekten abgelegten Datenwerte. Dabei wird dem Objektfenster eine Spalte angefügt, die die Art und den Inhalt des zum Objekt gehörigen Datenwertes beschreibt. + Die "Data"-Funktion erleichtert die Fehlersuche in Programmen.

Utilities	Verschiedene Kommandos.
Package as User-Defined	Konvertiert die markierten Objekte zu einem benutzerdefinierten Objekt.
Edit User-Defined	Ermöglicht das Editieren eines benutzerdefinierten Objekts.
Lock Unlock User-Defined	Umschalten von ‘Lock’ oder ‘Unlock’ eines ausgewählten benutzerdefinierten Objekts. Diese sind dann im Lock-Modus, wenn Sie an andere TestPoint-Anwender verteilt werden sollen und keine Änderungen zugelassen werden. Dabei muß ein Paßwort von 4 bis 40 Zeichen vorgegeben werden. Im ‘Unlocked’-Modus speichert TestPoint das Paßwort im TESTPT.INI File, so daß sie nicht extra eingegeben werden müssen.
Make Indirect Object	Erzeugt neue „indirekte“ Objekte des aktuell ausgewählten Objektes. Dieses Objekt ist des gleichen Typs wie das ausgewählte Objekt, ist aber „indirekt“ mit einem Pfeil im Symbol und einem „actual object“ Parameter in der Action List. Siehe Kapitel über Gruppen und indirekte Objekte.

Add Runtime Icon	Erstellen eines neuen Symbols im Programm Manager bzw. Win-Explorer für die entsprechende Anwendung.
Make Runtime Disk	Erzeugung einer eigenständig ablaufenden Applikation in Form einer oder mehrerer installationsfähigen Disketten.
Load Stock	Laden eines alternativen Stocks von der Festplatte oder Diskette. Dieser Stock wird zum Standard, wenn Sie TestPoint verlassen, da er dann als TPEDIT.STK beim Start von TestPoint wieder geladen wird.
Save Stock	Speichern des momentanen Stocks auf Festplatte.

Mode

Das Mode Menükommando schaltet zwischen dem EDIT und RUN Modus. Wenn die Ausführung im DEBUG Modus angehalten wurde, befinden sich unter dem Modus Menükommando einige weitere Optionen, die im Kapitel über das Debuggen behandelt werden.

Durch den Schalter wird die Art, wie sich die Maus auf dem Panel verhält, beeinflußt. Im Edit-Modus kann die Maus verwendet werden, um Objekte auf dem Panel zu verschieben oder deren Größe zu verändern.

Im Run-Modus verhält sich die Maus gegenüber den Objekten wie in einer eingenständigen Anwendung (Run-Time-Version) außerhalb des TestPoint Editors. Das Klicken auf einen Schalter (Pushbutton) beispielsweise aktiviert diesen.

Das Schalten vom Edit- auf den Run-Modus initialisiert die Anwendung.

Debug	Das Debug Menü hat Kommandos für das Setzen von Breakpoints (Unterbrechungspunkten) und für das schrittweise Abarbeiten der Action-Listen.
Breakpoint set/reset	Setzt oder löscht einen Breakpoint an der aktuell ausgewählten Zeile der Action-List. Breakpoints werden durch einen roten Punkt in der Action-List angezeigt. Wenn eine Anwendung diesen Breakpoint während der Ausführung erreicht, wartet diese, bis ein Kommando aus dem Mode Menü ausgewählt wurde.
Clear breakpoints	Löscht alle Breakpoints in dieser Anwendung.
View breakpoints	Öffnet ein Fenster, in dem alle Breakpoints der Anwendung mit Objekt-Namen und Action Zeilen Nummer aufgelistet werden. Jeder Breakpoint hat einen Zähler. Somit kann er nach einer bestimmbaren Anzahl von Durchläufen gesetzt werden.
Single step mode	Schaltet den Single-Step-(Einzelschritt)-Modus ein oder aus. Die Menüauswahl wird dementsprechend variiert. Wenn der Single Step Modus eingeschaltet ist, wartet TestPoint nach jeder Ausführung einer Action Zeile. Zusammen mit dem View Data Kommando kann dies sehr nützlich sein, wenn Sie Fehler suchen oder lernen möchten, wie TestPoint arbeitet.

Window	Das Window Menükommando hat Befehle zum Arbeiten mit den Fenster.
Arrange	Wiederherstellen der Standardeinstellungen der Fenster.
Objects	Öffnet das Object Fenster, falls es geschlossen war.
Stock	Öffnet den Stock, falls er geschlossen war.
1, 2, 3...	Schaltet zum entsprechenden Fenster. Diese Auswahl ist abhängig von den geöffneten Fenstern. Es ist oft hilfreich, um an Fenster zu gelangen, die im Augenblick im Hintergrund versteckt sind.

Help	Anzeigen der Hilfe über die Benutzung von TestPoint.
Index	Zeigen der verfügbaren Hilfe-Themen.
Using Help	Anzeigen einer Einführung zum Hilfe System.
About TestPoint	Anzeigen der Herstellerinformation und der Versionsnummer von TestPoint. Ebenfalls wird hier angezeigt, wie der Execution Control Key konfiguriert ist (Seriennummer und aktivierte Funktionen).

Tastatur und Maus

Der TestPoint Editor **erfordert** die Maus aufgrund der Ziehen und Ablegen Funktion, die elemetar für das TestPoint Konzept sind.

Die Anwendungen, die Sie erzeugen, benötigen **nicht** zwingend die Mausbedienung. Wenn Sie beispielsweise ein Produktions-Testsystem erstellen, das nur mit der Tastatur gesteuert wird, ist dies kein Problem.

Wie es Standard bei Windows-Anwendungen ist, wird der aktuelle „Fokus“ durch eine gestrichelte Linie um die Überschrift des Objektes dargestellt.

Benutzen Sie die TAB- und Umschalt-TAB-Taste, um den Fokus von einem Objekt zum nächsten zu bewegen. Die Reihenfolge der Objekte kann durch das Menükommando „Edit/Send to Front“ und „Edit/Send to Back“ eingestellt werden (Die TAB-Reihenfolge ist die gleiche, wie die Vordergrund/Hintergrund Sortierung).

Sie können Schalter mit der Eingabetaste drücken.

Wenn Sie die Funktionstasten F1 bis F10 in Ihrer Anwendung verwenden wollen, sehen Sie sich das Pgrammbeispiel FKEY.TST im „Example“-Verzeichnis Ihrer TestPoint Installation an. Es beinhaltet ein Objekt für die Funktionstasten, Löschen, Kopieren, Ausschneiden und Einfügen.

In TestPoint können verschiedene Dinge gelöscht, kopiert, verändert oder eingefügt werden. Das ‘Edit’ Menü enthält hierfür die entsprechenden Kommandos:

Delete entfernt den entsprechenden Punkt unmittelbar.

Copy erstellt eine Kopie in der Windows Zwischenablage.

Paste Fügt etwas aus der Windows Zwischenablage ein.

Cut ist eine Kombination von **copy** und **delete**.

Kopieren von Objekten

In der Objects List oder im Arbeitsfenster können die Objekte durch Anklicken und Löschen, Kopieren o. ä. ausgewählt werden. Wird ein Objekt in die Objects List eingefügt, wird es vor das aktuelle Objekt eingesetzt. Ist kein aktuelles Objekt angewählt, erscheint es am Ende der Objects List.

Kopieren von Action Zeilen

Befehlszeilen in Action Listen können nach Anklicken des Objektnamens gelöscht, kopiert o.ä. bearbeitet werden. Wird eine Befehlszeile in die Liste eingefügt, wird sie vor die aktuelle Zeile eingefügt. Ist keine Befehlszeile selektiert, erscheint der zuzufügende Befehl am Ende der Action Liste.

Kopieren von Action Parametern

Objektreferenzen in der Action Liste können durch Anklicken und Löschen, Kopieren o.ä. ausgewählt werden. Wird eine ‘Object Referenz’ eingefügt, ersetzt es den aktuellen Parameter. Ist kein Parameter angewählt, kann keine Objektreferenz eingefügt werden.

Kapitel Rückblick:

- **Stock Fenster**: Das Stock-Fenster enthält alle Objekte zum Aufbau Ihrer Applikation. Die Objekte werden vom „stock“ in das Arbeitsfenster oder die Objects List gezogen.
- **Object List Fenster**: Die Objects List zeigt alle Objekte, die in einem Arbeitsfenster enthalten sind. Die Objekte können hier auch in das Befehlsfenster gezogen werden.
- **Object (Detail) Fenster**: Das Objekt Detail Fenster kann Einstellungen, Action Listen, Kommentare und Cross-Referenzen anzeigen. Das View-Menü oder die rechte Maustaste sind einfache Wege, um diese Fenster für ein ausgewähltes Objekt zu sehen.
- **Kopieren/Einfügen**: Die Objekte in der ‘Object List’ oder im Arbeitsfenster und die Befehlszeilen mit den Referenzparametern können mit den genannten Menükommandos verschoben werden. Damit können Teile einer Anwendung leicht kopiert werden.
- **Fenster Bedienung**: Alle Fenster im TestPoint Editor können durch normale Windows Techniken, durch Anwahl der Ecken oder der Kopfzeile, mit der Maus verändert, verschoben oder auch ‘minimiert’ werden. Mit dem Window Arrange Menü Kommando wird das aktuelle Fenster wieder in die Grundeinstellung gebracht.

Kapitel 4. Kontrollfluß

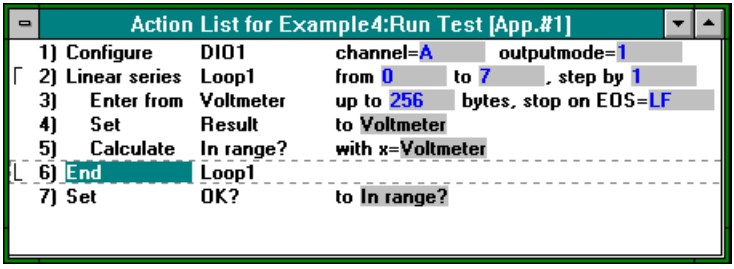
Was wird in diesem Kapitel behandelt?

- Ablaufsteuerung in TestPoint.
- Das Loop Objekt (Schleifen).
- Das Conditional Objekt (Bedingungen).
- Das Case Objekt (Fallunterscheidungen)

Die TestPoint Action-Listen beschreiben den Programmablauf. Sollen einzelne Aktionen wiederholt oder nach Erfüllen einer Bedingung, z. B. abhängig von bestimmten Datenwerten, gestartet werden, wird das Schleifen (Loop-), Bedingungs- (Conditional) oder Case (Fallunterscheidung) Objekt verwendet.

Diese Objekte brauchen mehr als eine Befehlszeile, wenn Sie in eine Action Liste übernommen werden. Die erste Zeile enthält die Laufanweisung oder Bedingung, die folgenden Zeilen die in Abhängigkeit davon abzuarbeitenden Befehle.

Zum Beispiel in dieser Action List:



Mit ‘Linear series’ von 0 bis 7 wird diese Schleife 8 mal ausgeführt. Der Schleifeninhalt besteht aus den Zeilen 3 bis 5. Diese Zeilen werden automatisch eingerückt.

+ Um die Schleifengrenzen für diese Objekte zu verändern, wählen Sie einfach eine andere Zahl für die obere oder untere Grenze.

Es können die Start- und auch die Endzeile verschoben werden. Sind aber mehrere oder verschachtelte Schleifen und Bedingungen vorhanden, lassen sich die Grenzen nicht verschieben, wenn Schleifen sich überschneiden. Eine Schleife kann eine andere enthalten, aber nicht von innen nach außen verschoben werden.

Zum Löschen einer Schleife oder einer Bedingung, braucht nur die entsprechende Zeile gelöscht zu werden. Die anderen Zeilen (z. B. ‘END LOOP’, ‘END IF’) werden dann automatisch gelöscht.

Schleifen

Das Loop Objekt wird zum wiederholten Abarbeiten von Befehlsfolgen verwendet. Es hat verschiedene Ablaufformen wie 'Linear', 'Geometric', und 'Decade' (1,2,5,10,20,50,...), aber auch die Bedingungen "Do while" und "Repeat until".

Linear series

Die einfachste Schleife ist die 'Linear series', die von einem festen Start- bis zum vorgegebenen Endwert mit der entsprechenden Schrittweite abläuft.

Beispiel : Diese Schleife wird 8 mal wiederholt:

```
1) Linear series  Loop1          from 1 to 8, step by 1
2)    Output to   DIO1           channel=A  value=Loop1
3) End  Loop1
```

Wird die Schleife ausgeführt, erhöht sich der Schleifenzähler mit jedem Durchlauf um 1. Also 1,2,3,4,5,6,7 und 8. Innerhalb dieser Schleife kann dieser Wert verwendet werden. Im Beispiel gibt die Zeile 2 diesen Schleifenwert am Ausgang einer digitalen I/O-Karte aus.

Geometric series

Die Schleife 'Geometric series' multipliziert bei jedem Durchlauf den Schleifenzähler mit einer Konstanten. Diese Schleife wird 4 mal wiederholt. Also 2, 4, 8 und 16:

1) Geometric series Loop1 from 2 to 16, multiplying by 2
2) Set Display1 to Loop1
3) End Loop1

+ Vermeiden Sie Schleifen mit dezimalen Schrittweiten (z. B. step by 0,1).

Decade series

Die Schleife 'Decade series' verwendet Werte wie 1, 2, 5 und den Exponenten von 10. Damit lassen sich logarithmische Achsen in einer Grafik erzeugen, wie sie z. B. bei Kurven für Frequenzgänge gebraucht werden. Dieses Beispiel durchläuft mehrere Frequenzen eines Funktionsgenerators, liest ein Voltmeter aus, sammelt die Meßwerte in einem Speicher (Container) und zeichnet dann aus den Werten die Meßkurve. Die Kurve ist im X-Y Modus mit einer logarithmischen X-Achse und Frequenzen von 10, 20, 50 usw. dargestellt.

1) Decade series	Loop1	from 10^1 to 10^4, in 1,2,5 sequence
2) Output to	FnGen	with "FREQ" , Loop1, term.=LF.
3) Enter from	Meter	up to 256 bytes, stop on EOS=LF
4) Append to	Container	from Loop1 , Meter
5) End	Loop1	
6) Draw graph	Curve	with Container

Do while und Repeat until

Mit "Do" läuft eine Schleife, so lange eine Bedingung wahr ist. Es gibt einen Parameter für den Ablauf, der einen Bezug zu Daten anderer Objekte haben muß, sonst würde dies eine Endlosschleife bedeuten. Die ‘Repeat’ Funktion ist ähnlich. Die Schleife läuft hier einmal und überprüft, ob eine Bedingung wahr geworden ist. Ist es das nicht, wird die Schleife jedesmal wiederholt. Zum Beispiel:

1) Enter from EOS=LF	Meter	up to 256 bytes, stop on
2) Store in	Container	from Meter
3) Repeat	Loop1 (non-zero)	until Math1 is true
4) Enter from EOS=LF	Meter	up to 256 bytes, stop on
5) Calculate	Math1	with x=Meter y=Container
6) Store in	Container	from Meter
7) End	Loop1	

Diese Action Liste liest ein Voltmeter so lange aus, bis die Differenz zweier aufeinanderfolgender Meßwerte geringer als 0,1 ist. Das Container Objekt speichert den jeweils letzten Wert ab und das Mathematik Objekt berechnet mit ‘abs(x-y) < 0.1’ das Ergebnis.

Anhalten einer Endlos-Schleife

Sollten Sie einmal versehentlich eine Endlosschleife erstellt haben, können Sie dies mit den Tasten Ctrl (Strg) Break (Pause) abbrechen.

Befinden Sie sich im TestPoint Editor, können Sie zum Stoppen aller Action Listen auch den Modus=‘Edit’ auswählen.

Wann Sie keine Schleife verwenden sollten

Das Mathematik Objekt von TestPoint kann in den meisten Fällen Vektoren und Felder für die Berechnung verwenden. Dies erübrigt in vielen Anwendungen die Verwendung einer Schleife.

Do while switch=on

Sie können die Schleife auch in Abhängigkeit einer Schalterstellung laufen lassen. Verwenden Sie dazu die folgende Action Liste:

1) Do loop	Loop1	while Switch1 is true
2) Enter from	Meter	up to 256 bytes, stop on
EOS=LF		
3) Set	Display1	to Meter
4) End	Loop1	

Hintergrund Schleifen

Wollen Sie einen sicheren Ablauf innerhalb Ihrer laufenden Applikation, sollten Sie dieses Beispiel verwenden. Dabei muß die Anfangsstellung des Schalters 'On' sein und damit seine Action Liste gestartet werden. Wollen Sie die Möglichkeit, diese ‘Background Loop’ anzuhalten, nicht nutzen, können Sie diesen Schalter auch als 'Invisible' definieren.

Loops und Multitasking

TestPoint unterstützt auch das **Multitasking.** Dabei werden mehr als ein Programm und mehr als eine Action Liste zur gleichen Zeit ausgeführt. Es wird immer nur ein Befehl einer Action Liste zu einem Zeitpunkt ausgeführt. Es ist möglich, mehrere Action Listen verschiedener Objekte quasiparallel, durch Sprünge zwischen den Action Listen untereinander, abzuarbeiten.

Der Multitasking Modus in TestPoint schaltet nach dem Starten zwischen einzelnen Tasks nur am Ende einer Schleife um.

Eine sequentielle Action Liste ohne Schleifen läuft von Beginn an bis zum Ende durch, ohne die Kontrolle an andere Programme oder Action Listen abzugeben. Diese sequentiellen Action Listen haben durch Ihre feste Länge eine endliche, maximale Ausführungszeit.

Eine Action Liste mit einer Schleife dagegen hat eine veränderliche Ausführungszeit. Nach dem Durchlaufen einer Schleife erlaubt TestPoint, andere Action Listen und Windows Programme zu starten.

Deshalb ist es durchaus sinnvoll, Applikationen mit dieser Konfiguration zu erstellen:

Switch1:

1) Do loop	Loop1	while Switch1 is true
2) Enter from	Meter	up to 256 bytes
3) Set	Display1	to Meter
4) End	Loop1	

Switch2:

1) Do loop	Loop2	while Switch2 is true
2) Input from	DIO1	channel="A"
3) End	Loop2	

Pushbutton1:

1) Sample	A/D1	once, channel(s)=0
2) Set	Display2	to A/D1

Wird Switch1 ausgeschaltet, beginnt die erste Action Liste und läuft solange dieser eingeschaltet bleibt. TestPoint reagiert aufgrund des Multitaskings auf Switch2 oder Pushbutton1 und erlaubt den Start dieser anderen Action Listen, obwohl die Action Liste des Switch1 aktiv ist.

Eine komplette Beschreibung des Multitasking in TestPoint erfolgt in einem anderen Kapitel.

Bedingungen

Das Conditional-Objekt bestimmt die Ausführung von Befehlszeilen in Abhängigkeit des Ergebnisses einer berechneten mathematischen Formel.

Die Formel, die gelöst werden soll, entspricht den Einstellungen eines Objekts und wird wie eine Formel im Mathematik-Objekt eingegeben. Das folgende Beispiel liefert als Ergebnis ‘true’, wenn die Variable ‘x’ zwischen den Werte 3 und 3.5 liegt:

```
(x >= 3) and (x <= 3.5)
```

Es gibt für dieses Objekt im TestPoint folgende zwei Möglichkeiten: Die If/Then und die If/Then/Else Anweisung. Der Unterschied besteht darin, daß die If/Then/Else Struktur in Abhängigkeit der gegebenen Bedingung zwei Befehlsfolgen starten kann. Eine davon, wenn die Bedingung zutrifft, und die andere, wenn sie nicht zutrifft.

In diesem Beispiel wird das Conditional-Objekt ‘In range’ genannt. Erfüllt der gelesene Wert vom Voltmeter die vorgegebene Bedingung, wird ein Action Objekt für den weiteren Ablauf gestartet (Zeile 3). Ist die Bedingung nicht erfüllt, wird ein Fenster ‘PromptWindow’ mit einem Hinweis für den Anwender geöffnet (Zeilen 5,6).

```
1) Enter from Meter up to 256 bytes, stop on EOS=LF
2) If/Then/Else In range with x=Meter
3) Execute Action1
4) Else if not In range
5) Show PromptWindow
6) Set Prompt to "Warning: cable is disconnected"
7) End If In range
```

Wollen Sie nur den Wert der Bedingung anzeigen lassen, können Sie das Indicator-Objekt und das Math-Objekt verwenden. Der mathematische Teil berechnet die erforderliche Bedingung und das Indicator Object zeigt in Abhängigkeit, ob es 0 oder 1 ist, das Ergebnis farblich kodiert an. Es ist keine If/Then/Else-Anweisung erforderlich.

Mehrfachnutzungen

Je universeller Sie die Angabe für das Conditional-Objekt bestimmen, desto einfacher wird es, dieses an mehreren Stellen in Ihrer Anwendung zu nutzen. Außerdem müssen sie weniger Objekte in Ihrer Applikation verwenden.

Werden verschieden Aufgaben bei einem Meßwert kleiner als 3, kleiner, größer oder gleich 5 erforderlich, können zwei oder drei Conditional Objects definiert werden. Aber nur ein Objekt wird für die Variablen im Objekt ‘expression’ setting wirklich gebraucht:

```
value < limit
```

Mit diesem Ausdruck kann das gleiche Conditional Objekt wie in dieser Action Liste verwendet werden:

```
1) If/Then/Else     Below          with value=Meter, limit=3
2)   Output to      DIO1           channel="A"  value=1
3) Else if not      Below
4)    If/Then/Else  Below          with value=Meter, limit=5
5)       Calculate  Math1          with x=Meter
6)       Output to  Control        with Math1
7)    Else if not   Below
8)       Set        Display1       to "Too high"
9)       Output to  DIO1           channel="A"  value=0
10)   End If        Below
11) End If          Below
```

Die Zeile 2 wird gestartet, wenn der Meßwert unter 3 bleibt. Die Zeile 5 und 6 werden gestartet, wenn der Meßwert zwischen 3 und 5 bleibt, und die Zeilen 8 und 9, wenn der Wert vom Gerät größer oder gleich 5 wird.

Datenwerte

Sollen nicht verschiedene Abläufe in Abhängigkeit eines Wertes gestartet werden, sondern verschiedene Werte für eine bestimmte Aktion, z. B. die Ausgabe für ein Meßgerät, bereitgestellt werden, kann das Math Objekt mit der Formel **if()** verwendet werden. Wollen Sie z. B. den Wert 2 ausgeben, wenn der Eingangswert kleiner 5 ist, und den Wert 8, wenn der Eingangswert größer als 5 ist, brauchen anstatt vieler Befehlszeilen in einem Conditional Objekt nur ein Mathematik Object mit der Formel:

```
if( input<5, 2, 8)
```

innerhalb der Action Liste:

```
1) Enter from     Meter      up to 256 bytes
2) Calculate      Output     with input=Meter
3 )Output to      D/A1       channel=0, value=Output
```

Fallunterscheidungen

Das Case-Objekt wird anstelle des Conditional-Objektes verwendet, wenn mehr als zwei Möglichkeiten zur Verfügung stehen müssen. Ein Case Objekt beginnt, wie ein Condition Objekt, mit der Lösung einer mathematischen Formel, mit einer beliebigen Anzahl von Argumenten. Dem Ergebnis entsprechend, wird eine Zeile oder eine Gruppe von Zeilen ausgeführt, wenn der zugehörige Fall eintritt.

1) Select	Type-of-Part	partnumber=Data-Entry1
2) When	Type-of-Part	is 100
3) Execute	Action1	
4) When	Type-of-Part	is Data-Entry2
5) Execute	Action2	
6)When/Range	Type-of-Part	from 300 to 499
7) Execute	Action3	
8) Set Display1		to Math1
9) Default	Type-of-Part	
10) Execute	Error-action	
11) End	Type-of-Part	

Bei der erstmaligen Verwendung des Case Objektes in einer Action List müssen Sie die **Select** Aktion auswählen, welche die Select und End Zeilen in die Action Liste einfügt:

1) Select	Case1	x=____
2) End	Case1	

Dabei sind die Parameter der Select Zeile abhängig von der mathematischen Funktion, die in den Einstellungen des Case Objektes angegeben wurde.

Danach können Sie eine beliebige Anzahl von Fällen einfügen, indem Sie das Case Objekt zwischen die Select und End Zeilen ziehen.

When

Wählen Sie die When Aktion, um einen Fall zu erzeugen, der dann ausgeführt wird, wenn das Ergebnis des mathematischen Ausdruckes gleich dem gegebenen Wert ist. Der Vergleichswert kann eine Konstante oder der Wert eines anderen Objektes sein.

When/Range

Wählen Sie die When/Range Aktion, um einen Fall zu erzeugen, der dann ausgeführt wird, wenn sich das Ergebnis des mathematischen Ausdruckes in einem gegebenen numerischen Bereich bewegt.

Default

Wählen Sie die Default Aktion, um einen Fall zu erzeugen, der dann ausgeführt wird, wenn keiner der anderen beschriebenen Fälle zutrifft.

Reihenfolge der Ausführung der Fallunterscheidungszeilen

Nach der Ausführung der Select Aktion werden die Fälle in der Reihenfolge ausgeführt, in der Sie in der Action List stehen, bis hin zu dem ersten Fall, der zutrifft.

Die ausgeführten Zeilen gehören zu dem ausgewählten Fall (When, When/Range, oder Default Aktion) bis zum nächstren Fall oder der End Aktion.

Ausdrücke

Der mathematische Ausdruck, der vom Case Objekt während der Select Aktion gelöst wird, kann beliebige verfügbare mathematische Funktionen verwenden. Der am meisten gebrauchte Ausdruck ist:

x

Dieser erlaubt die direkte Fallunterscheidung, basierend auf einem numerischen Wert des Parameters, der mittels der Select Aktion übergeben wurde.

Trotzdem ist selbstverständlich die Verwendung einer beliebigen Anzahl von Parametern möglich. Zum Beispiel kann dieser Ausdruck:

avg (x/y)

verwendet werden, um zwei Vektoren zu dividieren und deren Mittelwert für die Fallunterscheidung zu verwenden.

Fallunterscheidung mit Zeichenketten

Die When und Default Aktionen können sowohl für numerische, als auch für Zeichenketten verwendet werden. Die When/Range Aktion ist ausschließlich für numerische Werte geeignet. Um eine Fallunterscheidung mit einer Zeichenkette durchzuführen, kann jede zeichenkettenbasierende mathematische Funktion verwendet werden.

```
1) Select       Case1       string=Data-Entry1
2) When         Case1       is "abc"
3)   Execute    Action1
4) When         Case1       is "def"
5)   Execute    Action2
6) End  Case1
```

Zeichenkettenvergleiche werden verwendet, wenn der mathematische Wert und der Vergleichswert beides Zeichenketten, keine Zahlen sind. Um sicherzustellen, daß der mathematische Ausdruck nur Zeichenketten zurückgibt, auch wenn Zahlen enthalten sind, fügen Sie einfach einen Null-String hinten an, wie hier dargestellt:

string & ""

Verschachtelte Fallunterscheidungen

Sie können Fallunterscheidungs Aktionen verschachteln, sogar mit dem selben Objekt, wenn dessen mathematischer Ausdruck dazu geeignet ist. Beispiel:

```
1)Select        Case1       x=Data-Entry1
2)When          Case1       is 10
3)  Execute     Action1
4)When          Case1       is 20
5)  Select      Case1       x=Slider1
6)  When        Case1       is 34
7)    Execute   Action2
8)  Default     Case1
9)    Execute   Action3
10) End         Case1
11)Default      Case1
12) Execute     Action4
13)End          Case1
```

Wenn in diesem Beispiel der Wert des DataEntry1 Objektes 20 ist, dann trifft der Fall in Zeile 4 zu. In Zeile 5 beginnt eine neue Fallunterscheidung unter Verwendung des Slider1 Objektes.

- **Schleifen**: Das Schleifen-Objekt erlaubt das Wiederholen einer Gruppe von Befehlszeilen. Eine Schleife kann eine numerische Folge generieren oder in Abhängigkeit vom Datenwert eines anderen Objekts durchlaufen werden.
- **Bedingungen**: Das Conditional-Objekt bestimmt das Starten einer Gruppe von Befehlszeilen durch das Abprüfen einer Bedingung.
- **Fallunterscheidungen**: Das Case Objekt erlaubt die Ausführung einer Action Zeile oder einer Gruppe von Action Zeilen, in Abhängigkeit einer berechneten mathematischen Formel.
- **Schleifen und Multitasking**: TestPoint wartet auf bestimmte Ereignisse zum Starten von Befehlsabläufen. Am Ende dieser Schleifen ermöglicht TestPoint den Start anderer Action Listen.

Kapitel 5. Datentypen und Ein-/Ausgabe Formatierung

Was in diesem Kapitel behandet wird:

- Datentypen in TestPoint.
- Wie die automatische Abwicklung der Datentyen und die Formatierung arbeitet.
- Listen mit Daten und Auswahl einzelner Elemente von Listen.
- Spezifizierung der Ein-/Ausgabeformatierung.

Datentypen

Alle TestPoint Objekte beinhalten Datenwerte. Die von TestPoint dafür verwendeten Datentypen sind:

No Data - 'No data' is t ein Ausnahmetyp für nicht initialisierte Werte.
Viele mathematische Funktionen erzeugen eine Fehlermeldung, wenn ihnen ein 'no data' Argument übergeben wird. Die meisten Anzeige Objekte behandeln "no data" als eine leere Zeichenfolge oder als Wert 0.

Number - Zahlen werden in TestPoint als doppelte Gleitkomma Werte (double float) gespeichert.

String - Zeichenketten, Strings können in TestPoint eine beliebige Reihe von Zeichen bis zu einer Länge von 64K sein. Es dürfen auch nicht druckbare Zeichen Verwendung finden. Damit kann ein String eine beliebige Gruppe von Bytes enthalten.

Vector - Ein Vektor kann Zahlen oder Strings (nicht beides gleichzeitig) halten. Die Länge ist beliebig, bis zu 4x10^9 Elemente (oder die Grenze ist das verfügbare RAM, welches kleiner sein wird).

Array - Ein Array ist ähnlich wie ein Vektor, nur zweidimensional.

List - Eine Liste kann aus einer beliebigen Zahl von Elementen verschiedener Datentypen bestehen. Beispielsweise kann eine Liste aus einer Zahl, einem Vektor, einem String und einem weiteren Vektor bestehen.

Diese Datentypen erlauben es, eine wirklich große Vielfalt von Informationen zu verkörpern. Wenn auch ein TestPoint Objekt nur einen Datensatz enthalten kann, so erlaubt es der List Datentyp dem Objekt, mehrere Elemente zu kombinieren. Einige Aktionen von TestPoint Objekten resultieren in einem festen Datentyp oder die Datentypen können durch die Aktionsparameter bestimmt werden. Hardware- und Dateioperationen können jedoch in einen beliebigen Datentyp (any) resultieren.

NoData

NoData ist ein spezieller Datentyp für nicht initialisierte oder Null Daten.

Beispiel: Wenn Sie mit dem File Objekt über ein Dateiende hinauslesen, ist das Ergebnis **nodata**.

Typische Anwendung und Erzeugung

Startwert für Container und einige andere Objekte.

Leerer Parameter wie:

```
Set      Display     to _____
```

Mathematische Funktion für diesen Typ:

type(x) ***wenn x** nodata **ist**, **wird 0 zurückgegeben.***

Number

3.14159265

Zahlen werden in TestPoint als doppelte Gleitkomma Werte (double float) gespeichert. Dies erlaubt die exakte Darstellung sowohl von großen Integer Werten, als auch Gleitkommazahlen mit großem oder kleinem Betrag.

Grenzen und Speicherverbrauch

Zahlen haben einen Bereich von 2E-308 bis 2E308, sowohl positiv als auch negativ. Sie benötigen jeweils 8 Byte Speicher.

Typische Anwendung und Erzeugung

Konstanten in Action Zeilen, Dateneingabefelder (Data-Entry) die auf numeric eingestellt sind, Eingabewerte von Dateien oder externen Geräten, Mathematische Berechnung.

Mathematische Funktionen für diesen Typ

Alle mathematischen Funktionen.

type(x) wenn x eine Zahl ist, wird 1 zurückgegeben.

String

Abcdef0123*%xyz

Ein TestPoint String (Zeichenkette) besteht aus mindestens einem Zeichen oder Byte. Dabei ist es unerheblich, ob diese Zeichen druckbar sind oder nicht. Selbst NULL Bytes (0) können beinhaltet sein.

Quelle für die Strings können Data-Entry Objekte oder Eingabe Objekte (RS-2332, GPIB, File,...) sein, sowie mathematische Ausdrücke, die String Funktionen verwenden.

Der String Datentyp wird ebenfalls für nicht konvertierte A/D Daten, als Reihe uninterpretierter Bytes verwendet.

Grenzen und Speicherverbrauch

Strings können beliebige Längen zwischen 0 und 65534 Zeichen haben. Dementsprechend ist der Speicherverbrauch (2 Byte pro Zeichen) zuzüglich einiger Bytes für die Längen- und Typeninformation.

Typische Anwendung und Erzeugung

Konstanten in Action Zeilen, Dateneingabefelder die nicht auf numerisch eingestellt sind, Eingaben von Geräten, mathematische Berechnungen mit Strting Funktionen.

Mathematische Funktionen für diesen Typ

substr(str,start,end)
length(str)
str1 & str2
instr(str,otherstr)
type(x) wenn x ein String ist, wird 2 zurückgegeben.

Vector

[0]	**34.5**
[1]	**-573.25**
[2]	**3.14159**
	...

Vektoren (Vectors) können erzeugt werden von mathematischen Funktionen, File Objekten beim Einlesen von Datendateien oder A/D Objekten beim Einlesen von Spannungswerten eines Eingangskanals.

Grenzen und Speicherverbrauch

Vektoren können bis zu 2.097.152 Elemente enthalten. Ein numerischer Vektor benötigt 8 Byte pro Element. Ein String Vektor dagegen 6 Byte pro Element zuzüglich der String Länge.

Typische Anwendung und Erzeugung

A/D Werte (mehr als 1 Wert von einem Kanal), Eingaben aus Dateien oder von Geräten, mathematische Berechnungen.

Mathematische Funktionen für diesen Typ

v[i]
index(v,i)
subarray(v,start,end)
appendVector(v1,v2)
type(x) wenn x ein Vektor ist, wird 3 zurückgegeben.
Die meisten mathematischen Funktionen können direkt auf Vektoren angewendet werden.

Array

	[0]	[1]	[2]	...
[0]	**34**	**82**	**23**	**...**
[1]	**72**	**27**	**92**	**...**
...	**...**	**...**	**...**	**...**

Arrays (Felder) können erzeugt werden durch mathematische Funktionen oder durch Einlesen von Datendateien mit dem File Objekt.

Grenzen und Speicherverbrauch

Arrays können bis zu 2.097.152 Elements beinhalten. Ein numerisches Array verwendet 8 Bytes pro Element. Ein String Array dagegen verwendet 6 Bytes pro Element zuzüglich der String Länge.

Typische Anwendung und Erzeugung

Eingaben von Dateien oder Geräten, wenn ein Array Format vorgegeben wurde, Grid Objekt, wenn es auf Array Datentyp eingestellt ist, mathematische Berechnungen.

Mathematische Funktionen für diesen Typ

a[i,j]
subarray(a,startrow,endrow,startcol,endcol)
type(x) wenn x ein Array ist, wird 4 zurückgegeben.

Listen

Dies ist eine Liste bestehend aus: Zahl, String und Zahl:

3.14 , ABCDEF , -42E3

Ein weiteres Beispiel. Dies ist eine Liste aus 3 Vektoren:

23		**32**		**12**
32	**,**	**36**	**,**	**91**
…		**…**		**…**

Grenzen und Speicherverbrauch

Eine Liste kann bis zu 32767 Positionen irgendeines anderen Datentyps (Nodata, Number, String, Vector, oder Array) beinhalten. Eine Liste benötigt etwa 24 Bytes Speicher pro Position zuzüglich der Größe der Position selbst.

Typische Anwendung und Erzeugung

Eingaben von Dateien oder Geräten und mathematische Berechnungen.

Listen werden am häufigsten von A/D Eingaben in TestPoint verwendet. Wenn eine A/D-Karte Daten mehrerer Kanäle an TestPoint zurückgibt, ist dies eine Liste. Listen aus Vektoren werden vom Graph Objekt automatisch als verschiedene Linien gezeichnet.

Listen werden in TestPoint wesentlich häufiger verwendet als Arrays, weil ein Array unflexibler ist. Auf der anderen Seite ist die Liste eine natürliche Repräsentation verschiedener unabhängiger Daten, wie z. B. verschiedene A/D Kanäle.

Mathematische Funktionen für diesen Typ

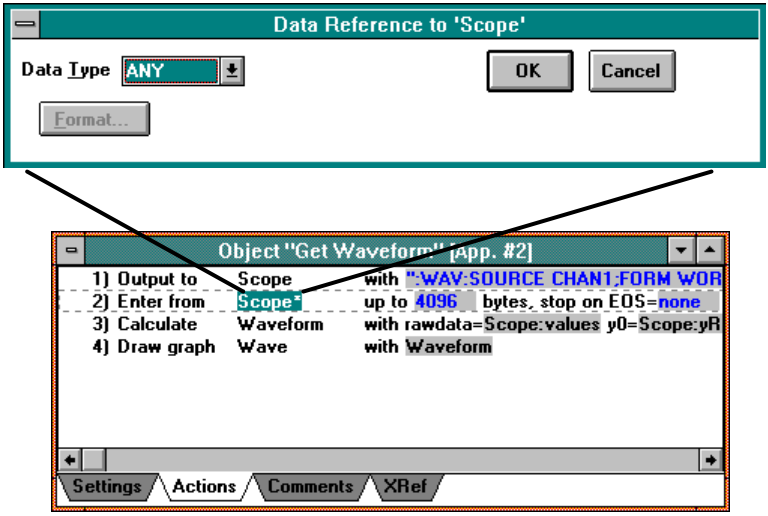
list(x,y,z)
select(list,item_number)

sublist(list,start,end)

type(x) wenn x eine Liste ist, wird 5 zurückgegeben.

Spezifizierung von Datentypen

Datentypen können in den Action Listen spezifiziert werden. Durch einen Doppelklick auf das zu formatierende Element wird ein Fenster geöffnet, welches die Möglichkeit zur Definition des Datentypes bietet.



Zur Formatierung der **Ausgaben** doppelklicken Sie auf die Parameter im rechten Teil der Ausgabe Action Zeile. Für die Formatierung der **Eingaben** klicken Sie doppelt auf die Objektnamen in der Action, wie „Scope“ in der Zeile 2 des oben aufgeführten Beispiels. Die Voreinstellung für jede neue Objektreferenz, wenn das Objekt aus der Objects List oder vom Arbeitsfenster in die Action Liste gezogen wird, ist die Datentypvereinbarung ‘ANY’. In den meisten Fällen ist dies ausreichend. TestPoint behandelt beispielsweise die Umwandlung von Daten und Strings in Zahlen automatisch.

Nur für Ein-/Ausgabe Operationen (z. B. Ausgabe an ein GPIB Gerät, Einlesen einer Datei, etc.) müssen gelegentlich Datentypen festgelegt werden.

Wann Sie Formatierungen verwenden sollten

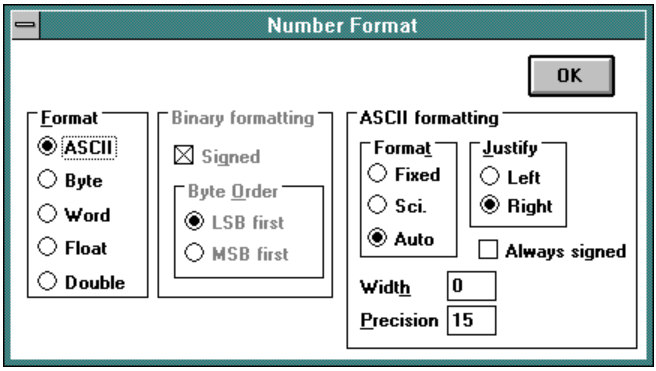
Es gibt zwei Fälle, in denen Datentypen in Action Listen Parametern spezifiziert werden müssen: Zur Listenelement Auswahl und zur Ein-/Ausgabeformatierung. In anderen Fällen, wie z. B. wenn ein Action Listen Parameter an eine mathematische Funktion übergeben wird, hat die Formatierung keinen Effekt.

> **Es ist nicht möglich, einen Datentyp zu deklarieren oder Daten mit dieser Methode in einen bestimmten Typ zu zwingen.**

Wenn in einer Objekt Referenz eine Formatierung an einer nicht erlaubten Stelle versucht wird, wie z. B. ein Parameter für eine mathematische Operation, wird der Formatschalter inaktiv sein.

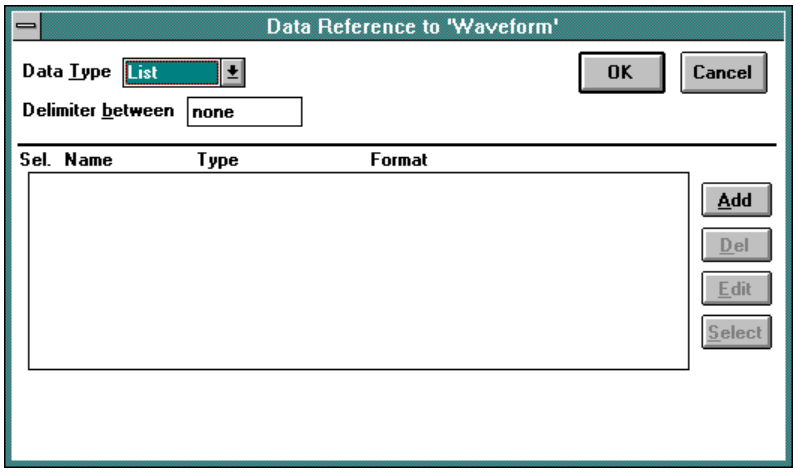
Auswahl des Formates

Nachdem ein Typ ausgewählt wurde, verwenden Sie den Formatschalter, um ein Fenster mit den Formatierungsinformationen zu bekommen. Beachten Sie dabei, daß die Formatierung **nur** mit Ein-/Aus-gabeoperationen zusammenarbeitet, mit denen Daten von einem externen Gerät eingelesen oder dorthin geschrieben werden oder mit Dateioperationen. Dies ist möglich mit folgenden Objekten: File, GPIB, RS-232, DDE und Report.



Das obere Bild zeigt das Format Fenster für numerische Daten. Eine Vielzahl von ASCII und Binär Formaten werden unterstützt. Ähnliche Optionen sind für jeden Daten-Typ verfügbar.

Wenn Sie den **list** Datentyp wählen, erhalten Sie folgende Dialogbox:



Die Schalter an der rechten Seite des Fensters erlauben das Verändern der Liste. Der Add Schalter fügt eine neue Position vor der aktuellen Position (oder am Ende der Liste, wenn keine Position ausgewählt ist) ein. Der Del Schalter löscht den ausgewählten Eintrag. Der Edit Schalter öffnet ein Datentyp Fenster für die ausgewählte Position und erlaubt Modifikationen. Beachten Sie, daß Sie für jede Position einen Namen vergeben können, der in der Action List zur Erklärung mit eingeblendet wird.

Wiederverwenden einer Formatierung

Ist ein Datentyp für eine Objekt Referenz einmal eingegeben, so kann diese Referenz in andere Action Zeilen gezogen werden. Die Information über den Datentyp und das Format sind mit dem Objektnamen verknüpft und müssen nicht nochmals definiert werden. Gleiches gilt auch beim Kopieren und Einfügen einer Objektreferenz.

Wie die Ein-/Ausgabeformatierung arbeitet

Ausgabe Formatierung

Ausgabe Actions in TestPoint, wie diese:

1)Output to File1 with A/D1, A/D2, term.=CRLF

nemen einen oder mehrere Datenwerte und senden sie an Ausgabegeräte, in diesem Fall eine Datei. Die Ausgabe Formatierung bezieht sich auf Aktionen, bei denen die Daten zu einer zeichenbasierenden Schnittstelle (Datei, GPIB-Schnittstelle, RS-232-Schnittstelle) gesendet werden. Sie ist auch beim Senden von Daten via DDE zu einem anderen Programm gültig, da die Daten im Textformat gesendet werden. Die Ausgabe Formatierung kann nicht bei Schnittstellen wie z. B. D/A-Ausgabe angewendet werden, da hier kein Text verwendet werden kann.

Jeder Ausgabe Parameter wird abwechselnd formatiert. Sogar wenn Sie keinen Parameter doppelgeklickt und somit ein Format bestimmt haben, wird ein Ausgabe Format angewendet. Sie können ein anderes auswählen, indem Sie einen Parameter in der Action Zeile doppelklicken, einen Datentyp und anschließend ein Format aussuchen.

Die meisten Ausgabe Aktionen beinhalten auch Abschlußparameter, die das einfache Hinzufügen von carriage return, Zeileneinzug, Tabulatoren oder anderer Separatoren ermöglichen. Sie können auch Codes benutzen wie z. B. TAB in den Ausgabe Parametern, wie z. B.:

1) Output to File1 with A/D1, TAB, A/D2, term.=CRLF

Eingabe Formatierung

Eingabe Actions in TestPoint, wie diese:

1) Input from	File1	up to 256 bytes, stopping at CRLF
2) Enter from	GPIB1	up to 256 bytes, stop on EOS=LF or EOI

lesen Zeichen von einem externen Gerät, wie der Festplatte oder einem Meßgerät und konvertieren sie in einen TestPoint Datenwert.

Das Einlesen geschieht in zwei Stufen: Lesen der Daten und Formatieren.

Der Umfang der gelesenen Daten hängt vollständig von den Parametern in der Action Zeile und der Art des Gerätes ab. Beispielsweise werden in Zeile 2 des oben angeführten Beispiels 256 Bytes vom GPIB Gerät eingelesen, wenn nicht vorher ein CRLF übertragen oder die EOI Leitung gesetzt wird. In Zeile 1 werden maximal 256 Bytes übertragen, wenn nicht vorher ein CRLF gelesen wird.

Der Umfang der Daten, die von einem Gerät eingelesen werden, wird nicht von der Formatierung beeinflußt.

Erst wenn die Daten eingelesen wurden, wird die Formatierung durchgeführt.

Wenn beispielsweise das Format ASCII Zahlen spezifiziert, dann werden die eingelesenen Daten nach dem Anfang einer Zahl durchsucht und die komplette Zahl konvertiert.

Wird ein Vektor oder ein komplexes Datenformat spezifiziert, wird jedes Element in der entsprechenden Reihenfolge formatiert und die

Begrenzungszeichen kontrollieren, wo ein Element endet und das nächste beginnt.

Die Formatierung kann alle oder nur einige Zeichen der eingelesenen Daten verwenden. Wenn die Daten beispielsweise sind:

```
DCV +5.3348E+02, Channel 1, Normal
```

und der Datentyp eine Zahl im ASCII Format ist, wird nur der numerische Teil verwendet. Die restlichen Zeichen werden ignoriert.

Fehler in Eingabeformaten

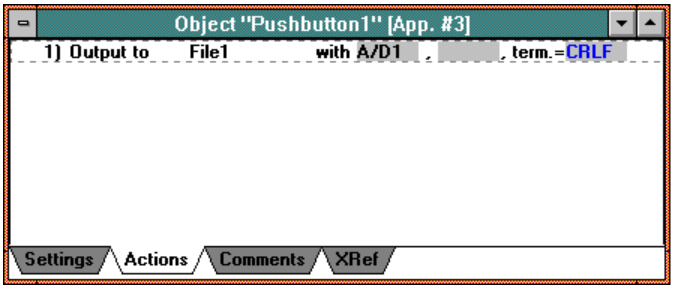
Sollten Sie Schwierigkeiten mit der Eingabeformatierung haben und die Daten, die Sie empfangen, nicht die erwarteten sein, ist es sinnvoll, in erster Näherung den Datentyp String, Delimiter=none einzugeben. Dabei werden alle Zeichen eingelesen und abgelegt. Sie können angesehen werden, indem Sie das Menükommando View/Data wählen oder die Daten auf die Festplatte schreiben und sich die Datei ansehen.

Wenn das Datenformat einmal exakt bekannt ist, ist es üblicherweise wesentlich einfacher, eine entsprechende Formatierung anzuwenden.

Bedenken Sie, daß Sie, falls Sie die eingelesenen Daten nicht in der Form splitten können, wie Sie es gern hätten, immer das Math Objekt mit Funktionen wie Substr(), usw. verwenden können.

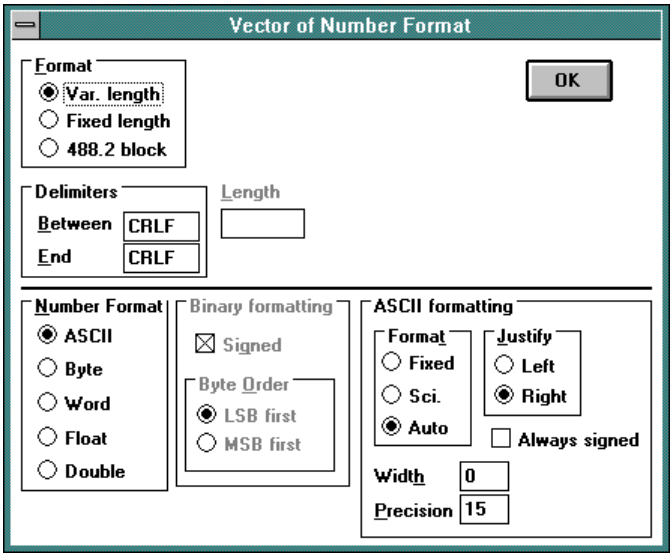
Beispiel: Numerische Formatierung

Häufig, wenn man die Daten eingelesen hat, will man diese für die Ausgabe in eine Datei oder durch DDE formatieren, um die Anzahl der Nachkommastellen einzustellen usw. Ein Doppelklick auf den Ausgabeparameter, wie A/D in der Action Liste:



bringt die Datentyp Dialogbox auf den Bildschirm.

Die Wahl **Vector of Number** und Anklicken von **Format** bringt den folgenden Dialog auf den Bildschirm:



Die Eingabe der Begrenzungszeichen (Delimiter) stellt die Zeichen zwischen den einzelnen Elementen und am Ende des Vektors ein. Die Voreinstellung ist CRLF und erzeugt eine Zeile pro Wert.

Der Zahlen Formatteil der Dialog Box stellt eine Auswahl von Details zur Verfügung, die jede Zahl des Vektors definiert. Im ASCII Format haben Sie beispielsweise die Optionen, den Dezimalpunkt fest einzustellen, wissenschaftliche Darstellung oder automatische Einstellung zu wählen.

\+ In der TESTPT.INI (2.0a) kann gewählt werden, welcher Dezimalseparator für Ausgaben in Dateien verwendet wird.

Beispiel: String I/O Formatierung

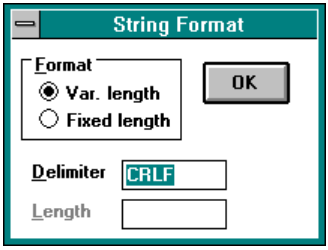
In TestPoint können Strings eine beliebige Reihe von Zeichen sein. Wenn die Daten von einer Datei oder einem Gerät gelesen werden, so definiert die Befehlszeile, wieviel Daten eingelesen werden sollen und wann das Einlesen beendet wird:

1) Enter from	File1	up to 32768 bytes, stop at CRLF
2) Enter from	File1	up to 32768 bytes, stop at ___
3) Enter from	GPIB1	up to 256 bytes, stop at EOS=LF or EOI

In Zeile 1 wird das Einlesen beendet, wenn ein Zeilenumbruch und ein Zeilenvorschub (Carriage Return, Line Feed, kurz CRLF) auftritt (oder nach 32K Byte). Zeile 2 liest 32K Byte oder bis zum Ende der Datei. Zeile 3 liest 256 Byte von einem GPIB Gerät, beendet dies jedoch vorher, an einem Line Feed oder bei einem EOI GPIB Signal.

In allen oberen drei Fällen, wenn ein String der Wert ist, werden **alle** Zeichen eingelesen. Nach Zeile 1 beispielsweise, enthält das File Objekt einen String mit der Endung CRLF, das ein 'Display' Objekt wie folgt anzeigt:

Diese zusätzlichen Zeichen kann man durch die Definition einer String Formatierung abschneiden, die den String an diesem Zeichen trennt:



Zur Ausgabe stellt die Actions Zeile normalerweise die Möglichkeit zur Verfügung, Abschlußzeichen an die Daten anzufügen:

```
1) Output to        File1          with Data-entry1,
   term.=CRLF
```

Im folgenden Beispiel mit zwei Strings kann es nötig sein, die Abschlußzeichen einzeln in dem Datenformat Fenster zu definieren:

```
1) Output to        File1          with Data-entry1,
                    Data-entry2,
                    term.=CRLF
```

Beispiel: Daten Dateien mit Kommentar-zeilen am Anfang

Manche Anwendungen verlangen, daß Meßdateien mit gemischtem Format gelesen werden, wie etwa Dateien mit einigen Kommentarzeilen gefolgt von Meßwerten. Oft sind diese Dateien durch andere Programme oder als Protokoll zur besseren Lesbarkeit erzeugt worden.

Wenn man den genauen Aufbau einer solchen Datei kennt, ist der einfachste Weg, die Kommentarzeilen und die Meßwerte in zwei getrennten 'Input from'-Aktionen einzulesen. Zum Beispiel :

```
1) Input from     File1          up to 2 lines,
                  stop at __
2) Set            Display1       to File1
3) Input from     File1          up to 32768
                  bytes, stop at __
```

Zeile 1 liest den zwei Zeilen langen Dateikopf ein, der in Zeile 2 zur Anzeige gebracht wird. Zeile 3 liest den Rest der Datei.

Hat der Dateikopf keine feste Länge, aber ein bekanntes Abschluß-zeichen, so kann etwas ähnliches wie folgt Verwendung finden:

```
1) Input from     File1          up to 30 lines,
                  stop at "*"
2) Input from     File1          up to 1 line, stop at __
3) Input from     File1          up to 32768 bytes,
   stop at __
```

Zeile 1 liest ein, bis ein Stern "*" gefunden wird und Zeile 2 liest den Rest der Zeile, die noch den Stern enthält. Zeile 3 liest dann den Rest der Datei ein.

Beispiel einer Daten Formatierung: Keithley 2001 Multimeter

Das Keithley Multimeter Typ K2001 stellt seine Meßwerte in verschiedenen Formaten zur Verfügung. Das voreingestellte Format besteht aus mehreren Parametern, die aus Meßwert, Kanalnummer, Einheiten, Zeitangabe, etc zusammengesetzt sind.

Hier zum Beispiel ein typischer String, den das K2001 sendet, wenn eine Spannungsmessung angefordert wird:

```
+3.482e1 VDC,00,1,34.57 sec,N
```

Wenn dieser String mit der automatischen Formatierung von TestPoint bearbeitet wird, so ist das Ergebnis ein Vektor mit vier Zahlen.

In diesem Beispiel soll nur ein Wert gelesen werden, der Spannungswert. Der Rest der Information kann ignoriert werden. Um dies zu tun, geht man einfach zur Zeile 2 in der folgenden Aktion Liste:

1) Output to	Meter term=LF	from ":MEAS:VOLT:DC?",
2) Enter from EOS=LF	Meter	up to 256 bytes, stop on

Man doppelklickt auf die ‘Meter’-Referenz in Zeile 2 und wählt den Datentyp ‘Number’ mit der Formatierung ‘ASCII’ als Voreinstellung. Nun liest TestPoint die erste Zahl des ankommenden Strings, überspringt alle nicht numerischen Präfixe und ignoriert den Rest der Information.

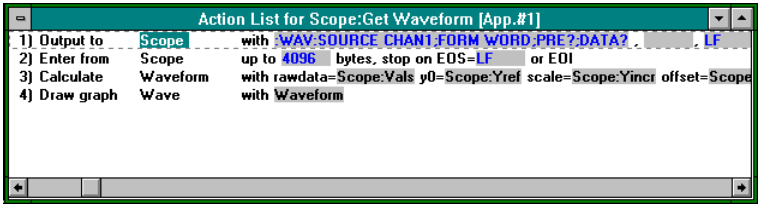
Beispiel einer Daten Formatierung: Digitales Oszilloskop

Für das Lesen einer Kurvenform von der Serie digitaler Oszilloskope HP545x ist es notwendig, zusätzlich zu den Meßdaten (Zahlenvektor) eine Skalierungsinformation einzulesen. Die Meßwerte werden als Rohdaten zur Verfügung gestellt, müssen jedoch vor der Verwendung skaliert und symmetriert werden..

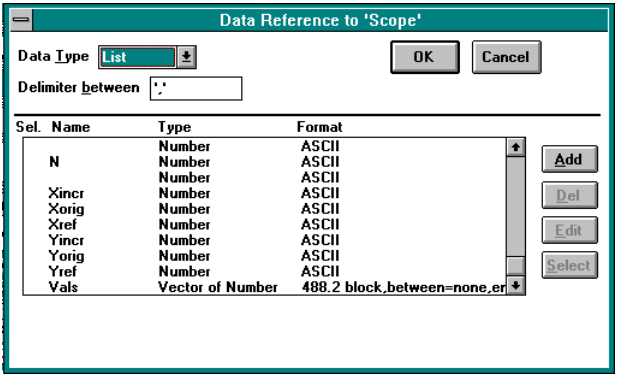
Dieses Beispiel ist bereitgestellt als HPSCOPE.TST.

Das Beispiel verwendet die folgenden Objekte: Einen Druckschalter ‘Get Wave’, ein GPIB Objekt ‘Scope’, ein Mathematik Objekt ‘Waveform’ und ein Grafik Objekt.

Die Action Liste für ‘Get Wave’

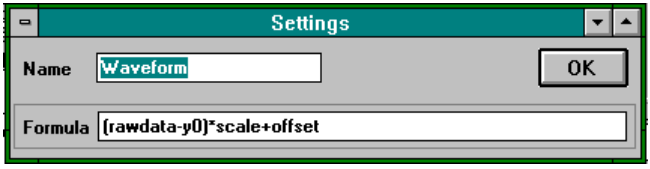


Nach dem Erstellen der Befehlszeile ‘Enter from’ muß als nächster Schritt in dieser Anwendung der Datentyp und das Format für die vom Oszilloskop kommenden Daten festgelegt werden. Für dieses Oszilloskop besteht die Information aus 10 ASCII Zahlen, getrennt durch Kommata, gefolgt von einem Kopf "#800001000" und dann 500 zwei Byte lange Zahlen (wobei die Bytes jeder Zahl in der umgekehrten Reihenfolge geordnet sind). Dieses ist eine ziemlich komplexe Kombination von Daten. In TestPoint wird daraus eine Liste erzeugt :



Die Erstellung derartiger Datenlisten wurde ausführlich im vorangengangenen Kapitel behandelt.

Die nächste Aktionszeile enthält ein Mathematikobjekt, welches die Kurvenform entsprechend der Spannungsvorgaben berechnet. Die Gleichung dazu lautet :



Die Zuordnung der Variablen für diese Berechnung erfolgte durch Ziehen der Objektreferenz ‘Scope’ in das Parameterfeld des

Mathematikobjektes und Selektieren der benötigten Daten aus der Liste.

Die Auswahl von Listen Elementen

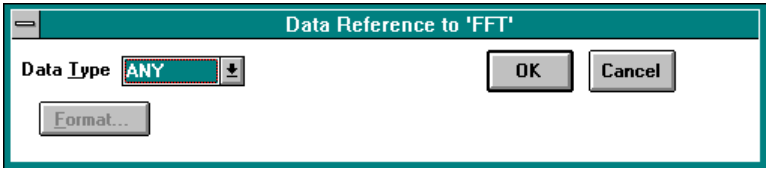
Angenommen, ein Objekt enthält einen '**List'** Datenwert und es soll zur Verwendung in einer Aktion ein Element aus der Liste ausgewählt werden.

+ Wichtiger Hinweis:
Die beste Methode zur Auswahl eines einzelnen Listeneintrags ist die Verwendung des Math Objektes mit der Funktion select(). Die unten beschriebene Auswahlmethode erfordert keinerlei mathematische Funktionen. Erforderlich ist es jedoch, das Datenformat von vornherein genau zu kennen.

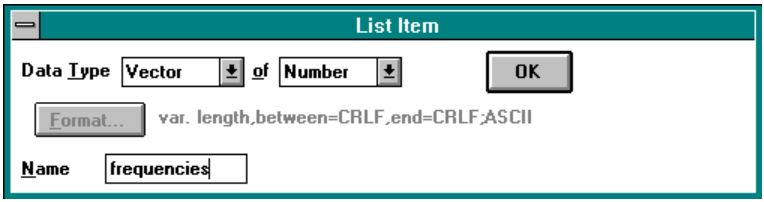
Beispiel: Die Funktion "fft" des Math Objektes gibt eine Liste mit drei Vektoren zurück: Frequenz, Amplitude und Phase. Nachdem die Anzeige stattfand, möchte man zur weiteren Betrachtung nur die Amplitude verwenden. Dazu die Action Liste:

1) Calculate	FFT	with waveform=File1
2) Draw graph	Spectrum	with FFT
3) Calculate	Filtered	with input=FFT:magnitude

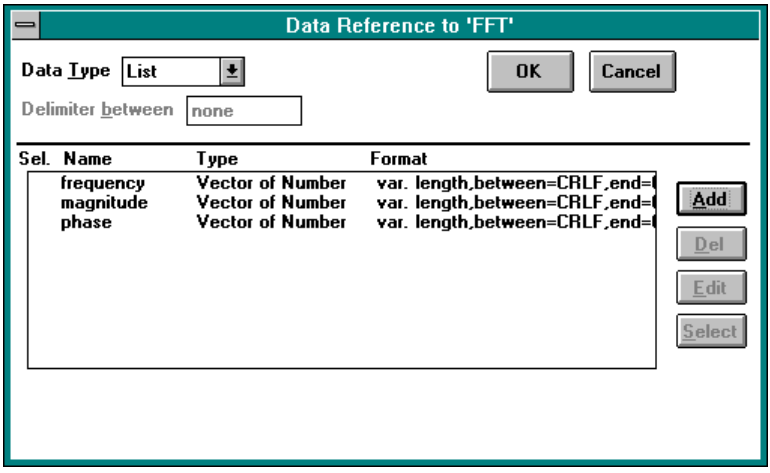
Zeile 3 ordnet die Amplitude aus der Liste des FFT Objekts zu. Dieser Parameter wurde durch folgende Schritte erzeugt: Zuerst durch einen Doppelklick auf 'FFT' in Zeile 1, es erscheint das Datentyp Fenster:



Aus dem ‘drop down’-Menü wird der Datentyp ‘List’ ausgewählt. Mit der ‘Add’ Option wird ein neues Listenelement hinzugefügt, welches als ein Zahlenvektor definiert wird und den Namen ‘frequencies’ erhält.



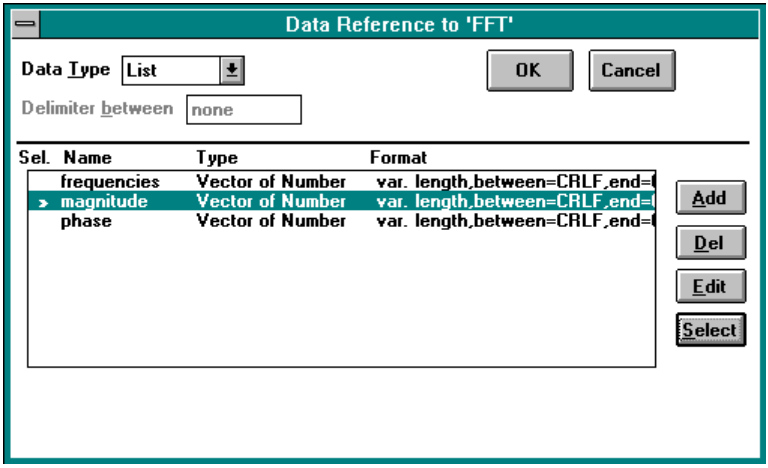
Zwei weitere Listen Elemente werden hinzugefügt, eines für die Amplitude und eines für die Phase. Abschließend bestätigt man die Einstellungen mit ‘OK’. Damit ist der Ergebnisdatentyp für die mathematische Operation beschrieben.



Nun wird die ‘FFT’ Referenz von Zeile 1 zu Zeile 3 in das Parameterfeld für die zweite Berechnung gezogen. Zu diesem Zeitpunkt zeigt sich die Zeile 3 folgendermaßen:

3) Calculate Filtered with input=FFT

Einen Doppelklick auf den ‘FFT’ Parameter und es öffnet sich wieder das Datentypen Fenster. Dann wird das Element ‘magnitude’ mit der Maus angewählt und der ‘Select’-Schalter angeklickt:



Nach Schließen des Datentypfensters sieht die Zeile 3 so aus:

3) Calculate Filtered with input=FFT:magnitude

Die letzte Darstellung kennzeichnet ein spezielles Element der Datenliste des FFT-Ergebnisses, die Amplitude. Die Berechnung der Filterung erfolgt nur mit dieser selektierten Wertefolge, nicht für die gesamten drei Datensätze der FFT.

Kapitel Rückblick:

- **Datentypen**: Die Daten der Objekte in TestPoint können verschiedene Formate besitzen : Zahlen, Felder, Strings, Vektoren oder Listen. Die Objekte und Aktionen in TestPoint sind sehr flexibel in der Handhabung der Datentypen, so daß normalerweise darauf nicht geachtet werden muß.
- **Daten Referenz**: Daten Typen und Formatierungen werden durch einen Doppelklick auf den Objektnamen in der Aktionszeile spezifiziert, wodurch ein weiteres Fenster -das Datenreferenz-Fenster- geöffnet wird. Wird ein Objektname von einer Aktionszeile in eine andere gezogen, so werden alle Datentyp- und Formatierungsinformationen beibehalten. Das gilt auch für das Kopieren zwischen Action Listen.
- **Listen Auswahl**: Benötigt man aus einer Ergebnisliste einer Aktion nur ein einzelnes Element, so kann man dieses im Datenreferenz-Fenster selektieren und in die folgenden Aktionszeilen übernehmen.
- **Formatierung**: Eine Formatierung kann für Einlese- und Ausgabe-aktionen verwendet werden. Zahlen können als ASCII oder binär formatiert sein. Für Listen, Vektoren, Felder und Strings lassen sich zusätzlich verschiedene Trennzeichen einstellen..

Kapitel 6. Organisieren großer Anwendungen

Was in diesem Kapitel behandelt wird:

- Wie Anwendungen übersichtlich gehalten werden.
- Werkzeuge zur Organisation großer Anwendungen.

Vielfachnutzung von Conditional Objekten

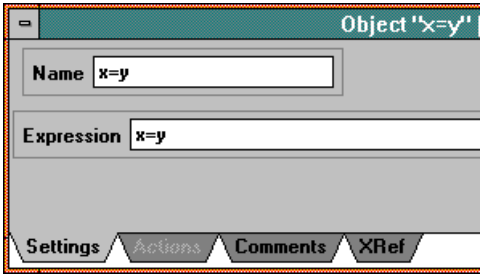
Die häufigste Verwendung des Conditional Objektes (If/Then/Else) ist die Überprüfung auf Gleichheit. Im einfachsten Falle auf Null oder Eins:

```
1) If/Then              EqualToZero      with x=Data-Entry1
          ...do some actions here
n) End If               EqualToZero
```

Es gibt keinen Grund, warum mehrere Conditional Objekte verwendet werden sollten, wenn diese sich nur im Vergleich unterscheiden.

Sollte an einer Stelle Ihrer Anwendung der Vergleich auf Null, an anderer Stelle der Vergleich auf Eins erforderlich sein, können Sie trotzdem ein einziges Conditional Objekt für alle Fälle verwenden.

Verwenden Sie einfach den Ausdruck „x=y“ im Conditional Objekt und verwenden Sie dies gleich als Namen:



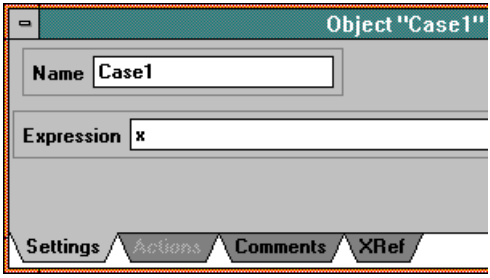
Das Objekt kann, auch verschachtelt, mehrfach verwendet werden:

```
1) If/Then              x=y              with x=Data-Entry1, y=0
    ... do some actions here
n)  If/Then             x=y              with x=Data-Entry2, y=1
       ... do more actions here
```

Die gleiche Vereinfachung kann auch bei komplexeren Ausdrücken verwendet werden. **Ersetzen Sie in Conditional Objekten die Konstanten durch Variablen, wenn immer es möglich ist.**

Vielfachnutzung von Fallunterscheidungen

Die gebräuchlichste Fallunterscheidung mit dem Case Objekt, ist die einfache Überprüfung auf einen bestimmten Wert, unter Verwendung der Variable x:



1) Select	Case1	with x=Data-Entry1
2) When	Case1	is 0
3) Execute	Action1	
4)When	Case1	is 2
... etc.		

Das gleiche Case Objekt kann an jeder Stelle wiederverwendet werden, an der eine Fallunterscheidung auf einzelne Werte erforderlich ist. Sogar Verschachtelungen oder die Anwendung in verschiedenen Action Listen ist möglich.

Vermeidung wiederholter Aktionen

Bei der Zusammenstellung von Action Listen ist es nicht unüblich, daß beim Auftreten wiederholter Aktionen die Copy/Paste Funktion verwendet wird, um die Aktionen zu duplizieren.

```
1) Output to        RS232       from "Read A"
2) Enter from       RS232       up to 256 bytes
3) If/Then          x>y         with x=RS232, y=2
4)   Set            Alarm       to 1
5)End If            x>y
6) Output to        RS232       from "Read B"
7)Enter from        RS232       up to 256 bytes
8) If/Then          x>y         with x=RS232, y=2
9)   Set            Alarm       to 1
10) End If          x>y
...
... usw. Viele, viele Wiederholungen.
...
50) End If          x>y
```

Wenn Aktionen sich mit kleinen Variationen wiederholen, gibt es eine große Wahrscheinlichkeit, daß die Action Liste vereinfacht werden kann.

Schleifen mit einfachen Berechnungen

Im obenstehenden Beispiel werden an ein Gerät die Zeichenketten „Read A“, „Read B“, usw. übertragen und jeder Wert wird auf eine Überschreitung eines Grenzwertes überprüft. **Wenn sich dies alles zwischen zwei Wiederholungen ändert, verwenden Sie einfach eine Schleife:**

```
1) Linear series    Loop1          from 1 to 10, step by 1
2)   Calculate        Command        with x=Loop1
3)   Output to        RS232          from Command
4)   Enter from       RS232          up to 256 bytes
5)   If/Then          x>y            with x=RS232, y=2
6)     Set           Alarm          to 1
7)   End If          x>y
8) End                Loop1
```

Dabei werden die konstanten Zeichenketten ersetzt durch einen einfachen mathematischen Ausdruck, der auf dem Schleifenzähler basiert, der automatisch mitgeführt wird. Die Formel für das Math Objekt „Command“ ist in diesem Beispiel:

```
"Read "  &  chr( x-1 + asc("A"))
```

Schleifen mit Vergleichs-Tabelle

Wenn die Konstanten mehr in willkürlicher Art und Weise variieren, wird der Schleifenzähler die Aufgabe nicht erfüllen:

```
1)Output to          GPIB          with "command1"
... Mehr Aktionen
6)Output to          GPIB          with "another string"
... wiederholen der gleichen Aktionen
11)Output to         GPIB          with "xyz"
... und noch einmal
... usw.
```

Für eine willkürliche Reihe von konstanten Werten verwenden Sie eine Vergleichs-Tabelle.

Die Vergleichsliste kann auf zwei verschiedene Arten erzeugt werden: Aus einer Datei und dem File Objekt oder als Konstanten-Tabelle in den Settings eines User-defined Objektes (siehe Beispiel CONSTANT.TST).

Eine Tabelle aus einer Datei

Um eine Datei zu verwenden, erzeugen Sie einfach eine Textdatei, die Ihre Konstanten enthält. Eine Konstante pro Zeile, wie hier:

```
Kommando1
Ein anderes Wort
xyz
```

Bringen Sie nun die sich wiederholenden Aktionen in eine Schleife und verwenden Sie das **File Objekt**, um die Befehle zu lesen, wie hier:

```
1) Repeat loop     Loop1          until File1 is true
2)   Enter from    File1          up to 1 line
3)   Output to     GPIB           from File1
4)   Execute       RestOfActions
5)   Test EOF      File1
```

6) End Loop1

Eine Tabelle in einem Objekt gespeichert

Sie können einen konstanten Vektor oder eine Liste in einem Objekt in Ihrer Anwendung speichern. Dies erfordert keine externe Datei (obwohl die Datei-Methode Änderungen in der Tabelle einfach macht!).

Die Datei EXAMPLES\CONSTANT.TST, die mit TestPoint ausgeliefert wird, verwendet ein solches Objekt, welches einen Vektor aus Zahlen, Zeichenketten, usw. beinhalten kann. Sie können es verwenden, indem Sie es mit der Copy/Paste Funktion in Ihre Anwendung kopieren. Dann können Sie Action Listen wie diese erzeugen:

1) Linear series	Loop1	from 1 to 10, step by 1
2) Calculate	Command	with table=Constant,
	i=Loop1	
3) Output to	GPIB	from Command
4) Execute	RestOfActions	
5)End	Loop1	

Dabei ist die Formel für das Math Objekt "Command":

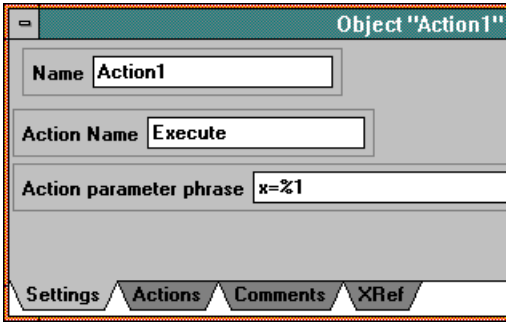
index(table,i)

Action Objekte mit Parametern

Manchmal unterscheiden sich die Zeilen in mehr als nur den Konstanten. Sie können z. B. Daten von verschiedene Objekten verwenden:

```
1)If/Then                Condition1      with x=Data-Entry1
2)   Output to           File1           with Data-Entry1
... lots of action lines here...
11If/Then                Condition1      with x=Data-Entry2
12)  Output to           File1           with Data-Entry2
   Wiederholung
21)If/Then               Condition1      with x=Data-Entry3
22)  Output to           File1           with Data-Entry3
   nochmal alle Zeilen wiederholen...
```

Diese Art von Action List kann durch Verwendung eines Action  Objektes vereinfacht werden, das sich wie eine Unterroutine mit Parametern verhält. Verwenden Sie in Ihrer Anwendung ein neues Action Objekt und definieren Sie die „Action parameter phrase“ durch Platzhalter wie „%1“, „%2“, usw. für die Parameter, die sich verändern werden:



Übernehmen Sie dann alle sich wiederholenden Action Zeilen in die Action Liste des Action Objektes. Verwenden Sie die Copy/Cut Funktion, um die Original Zeilen auszuschneiden und Edit/Paste, um die ausgeschnittenen Zeilen in der Action Liste des Action Objektes einzufügen. Übernehmen Sie das Action Objekt in Ihre Original Action Liste, um die neue Unterroutine aufzurufen:

Original Action Liste:

1)Execute	Action1	x=Data-Entry1
2)Execute	Action1	x=Data-Entry2
3)Execute	Action1	x=Data-Entry3

...usw. ...

Liste für das Action Objekt:

1)If/Then	Condition1	with x="%1"
2) Output to	File1	"%1"

... weitere sich wiederholende Aktionen...

Gruppen mit Indizierung

Wenn sich wiederholende Action Zeilen sich nicht nur in den Parametern, sondern auch in den Objekten unterscheiden, enstehen Action Listen wie diese:

```
1)Output to          RS232          from "Read A"
2)Enter from         RS232          up to 256 bytes
3)Set                DisplayA       to RS232
... usw..
11)Output to         RS232          from "Read B"
12)Enter from        RS232          up to 256 bytes
13)Set               DisplayB       to RS232
                     ... usw..
21)RS232             from "Read C"
22)Enter from        RS232          up to 256 bytes
23)                  Set            DisplayC      to RS232
... usw..
```

Dabei sind die Zeilen 3, 13, 23, usw. entstanden durch unterschiedliche Objekte (DisplayA, DisplayB, usw.). Hier können Objekt Gruppen verwendet werden, um die Anwendung zu vereinfachen. Objekt Gruppen werden in **Kapitel 17** ausführlich erläutert. Dies läßt Sie ähnliche Objekte kombinieren und über eine Indexnummer zugreifen, wie in Zeile 5 dieser Action Liste:

```
1)Linear series      Loop1          from 1 to 10, step by 1
2)  Calculate        Command        with x=Loop1
3)  Output to        RS232          from Command
4)  Enter from       RS232          up to 256 bytes
5)  Set              Display-group  to RS232  [index=Loop1]
6)End                Loop1
```

Handhabung komplexer Anwendungen

Einige Anwendungen sind einfach von Natur aus groß, deshalb kann man sie nur in gewissem Maße vereinfachen. Wie kann man aber nun diese große TestPoint Anwendung handhaben, um sie verständlich und übersichtlich zu halten?

Werkzeuge zur Organisation von Object Listen

Wenn die Object Listen lang werden, können sie durch drei Werkzeuge organisiert werden: **Gruppen**, **Panels** und **User-defined Objekte**.

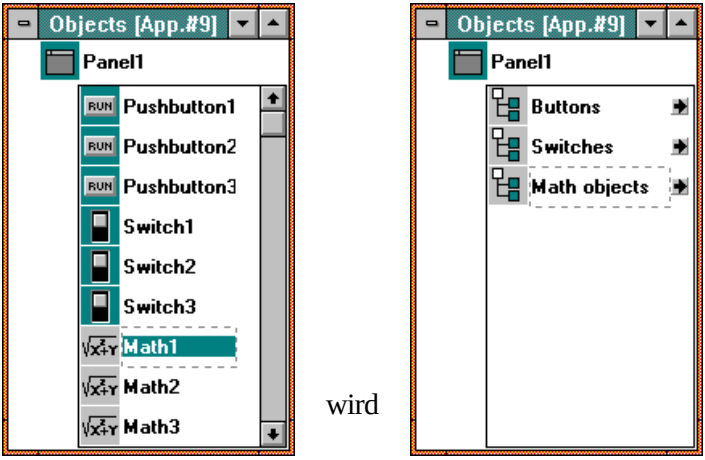
Gruppen fassen Objekte zusammen und vereinfachen Action Listen.

Panels sind Anzeigefenster von Anwendungen. Objekte, die in einem Panel plaziert werden, erscheinen zusammen auf dem Bildschirm. Außerdem haben Panels eigene Object Listen. Dies hilft also dabei, die Object Listen kurz zu halten.

User-defined Objekte sind neue Objekte, die durch Kombination anderer Objekte entstanden sind. Sie sind zwar etwas schwieriger zu benutzen, erlauben aber eine sehr gute Programmstrukturierung. Die Objekte innerhalb eine User-defined Objektes bilden eigene Object Listen und halten somit ebenfalls die Haupt Object Liste kürzer.

Gruppen

Objekte werden gruppiert, indem sie selektiert werden und das Edit/Group Menükommando aufgerufen wird. Gruppen können zum einen für die erweiterte Programmierung, wie in Kapitel 17 beschrieben, verwendet werden, zum anderen einfach dazu, aus langen Action Listen kürzere zu machen:



Um einzelne Objekte zu sehen, klicken Sie auf den Pfeil auf der rechten Seite des Group Objektes. Sie gelangen dann in die Object Liste des Group Objektes.

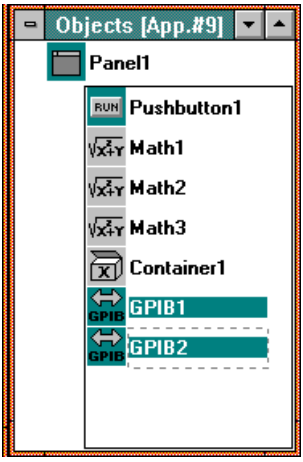
Objekte können auf jede Art gruppiert werden, in der sie für den Anwender besser handzuhaben sind: nach Objekt Typ (wie oben gezeigt) oder nach funktionellen Gruppen der Anwendung.

Beachten Sie, daß Objekte in Gruppen sich zusammen bewegen, wenn ein Objekt dieser Gruppe auf dem Panel Fenster verschoben wird. Sie können dies vermeiden, indem Sie die Umschalt-(Shift-)Taste bei der Verschiebung gedrückt halten.

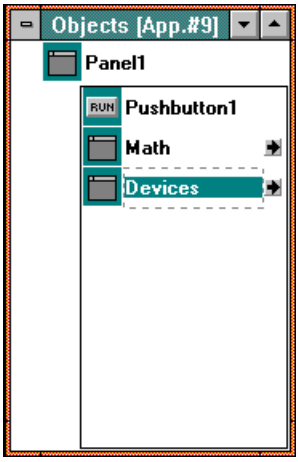
Panels

Die Verwendung neuer Panel Objekte  in Ihrer Anwendung läßt Sie mehrere Fenster gleichzeitig auf dem Bildschirm darstellen. Dadurch können dem Endanwender übersichtlicher Bildschirme zur Verfügung gestellt werden.

Durch Objekte in verschiedenen Fenstern kann ebenfalls die Object Liste in kürzere und einfachere Listen gesplittet werden:



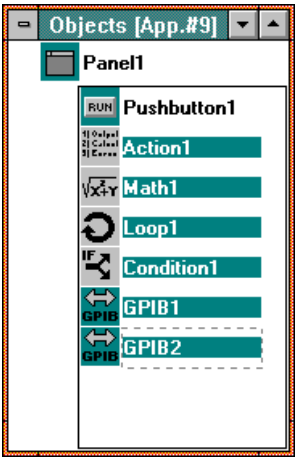
wird



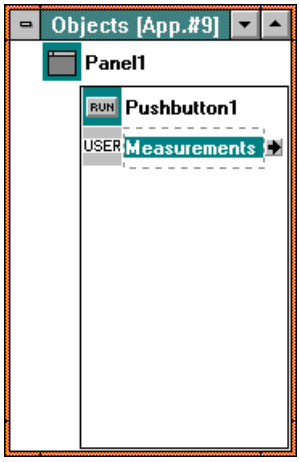
Die einzige Einschränkung ist, daß Objekte, die zusammen auf dem Bildschirm dargestellt werden sollen, im gleichen Panel sein müssen.

User-defined Objekte

User-defined Objekte werden in Kapitel 22 ausführlich beschieben. Sie werden erzeugt durch das Selektieren bestehender Objekte und die Benutzung des Menükommandos Utilities (Package as User-defined):



wird



User-defined Objekte sind sehr leistungsfähig - sie können komplett neue Objektkategorien schaffen, die alle Möglichkeiten der vorhandenen Objekte von TestPoint nutzen können. Trotzdem sind sie etwas komplexer zu erzeugen und korrekt zu benutzen. Möglicherweise ist der beste Weg, sie dann zu verwenden, wenn Sie mit TestPoint vertraut sind.

Kapitel 7. A/D und D/A

Was in diesem Kapitel behandelt wird:

- Analoge Eingabe und Ausgabe mit TestPoint.
- Die Funktionsweise der ereignis-gesteuerten A/D Aufnahme.
- Linienschreiber in Echtzeit.
- Disk logging of data. (Speicherung in Datei).
- Nachbearbeitung der Daten.

Einstellungen

Bevor eine Einsteckkarte mit TestPoint zusammen verwendet werden kann, muß beides korrekt konfiguriert sein, die Software und die Hardware. Die Konfiguration ist anhängig vom Hersteller und vom Typ der Karte, die Sie verwenden. Alle erforderlichen Softwaretreiber und Hinweise für viele verschiedene Karten sind im TestPoint inbegriffen.

Um eine Anleitung für einen bestimmten Kartentyp zu bekommen, doppelklicken Sie auf das **ReadMe - A/D Setup** Symbol in der TestPoint Programmgruppe.

Folgen Sie bitte diesen Anweisungen zur Installation Ihrer Karte.

Die typischen Schritte bei der Konfiguration:

Hardware Einstellungen

Als erstes müssen die Schalter und Jumper der Karte entsprechend der Herstellerangaben für die Karte eingestellt werden.

Die geeigneten Einstellungen sind u. U. abhängig von weiteren Karten, die Sie in Ihrem PC eingebaut haben. Sie müssen Konflikte zwischen den Einstellungen der A/D-Karte und anderen Karten bei der Auswahl der **I/O Adresse**, des **IRQ (Interrupt) Levels** und des **DMA Kanals** vermeiden.

In den meisten PC´s sind folgende IRQ Levels **nicht** verfügbar: **0, 1, 4, 6**. In einigen Fällen wird der IRQ 3 ebenfalls (für COM2) verwendet.

In fast allen PC´s sind die DMA Kanäle **0 und 2 nicht** verfügbar.

Treiber Software Einstellungen

Für die meisten A/D und D/A Karten liefert der Hersteller Windows Treiber. TestPoint verwendet diesen Treiber normalerweise. Somit ist der nächste Schritt, die Treiber Software des Herstellers entsprechend dessen Angaben einzustellen (wie in der ReadMe - A/D Datei beschrieben).

Verwenden Sie bitte NICHT die Angaben des Herstellers, wenn es für TestPoint besondere Angaben für die Einstellungen gibt. Die Herstellerangaben treffen in diesem Fall für die Benutzung von Basic oder Pascal zu.

Für Einsteckkarten von Keithley/Metrabyte starten Sie das Installationsprogramm, welches eine CFG-Datei erzeugt, die die I/O Adresse usw. beinhaltet. Karten von Data Translation werden über das Windows Control Panel eingestellt.

Die Kartenhersteller liefern ein Diagnoseprogramm, das die korrekte Installation überprüft. In diesem Fall können Sie sicher sein, daß die Einstellungen in Ordnung sind, bevor Sie TestPoint starten.

TestPoint Einstellungen

Im letzten Schritt wird TestPoint mitgeteilt, welche Karte Sie installiert haben.

Bei einigen Herstellern geht dies automatisch. Die Voreinstellung von TestPoint sucht automatisch heraus, welche Karten installiert sind. Das geht mit Karten von Keithley/Metrabyte, Data Translation oder NI.

Bei anderen Herstellern, welche die automatische Einstellung nicht unterstützen, muß die TESTPT.INI Datei geändert werden. Der Abschnitt für die A/D Karten beinhaltet Informationen über Hersteller und Einstellungsdaten. Komplette Hinweise finden Sie in der jeweils entsprechenden Readme Datei (wenn der Treiber mit TestPoint geliefert wird) oder in der Dokumentation des Herstellers.

+ Eine ausführliche Installationsanweisung für Ihre A/D bzw. D/A Karte finden Sie in der Datei "Readme-A/D Setup" in der TestPoint-Programmgruppe.

A/D

Das A/D Objekt in TestPoint wird zum Messen von analogen Eingangsgrößen mit A/D Einsteckkarten benützt. Die analogen Eingänge können als Einzelmessungen oder als zeitlich getaktete Reihe für einen oder mehrere Kanäle erfolgen.

Jedes A/D Objekt steht für eine einzelne A/D Einsteckkarte. Werden mehr als eine benutzt, benötigt man entsprechend viele A/D Objekte. Bis zu vier A/D Einsteckkarten werden gleichzeitig unterstützt. Mit dem A/D Objekt können alle Meßkanäle der A/D Meßkarte abgefragt werden.

Das A/D Objekt erlaubt Einstellungen (Settings) für die Kartennummer (Board Number) und den Demo Modus. Die Board Nummer verweist auf die Hardware Konfiguration in der TESTPT.INI Datei. Genauere An-gaben lassen sich im Kapitel Hardware Konfiguration ersehen. Der Demo Mode wird zum Testen verwendet, wenn keine A/D Einsteckkarte verfügbar ist.

Messung eines Datenpunktes

Einzelmessung werden mit "Sample A/D" vorgenommen:

1)Sample A/D	A/D1	once, channel(s)=0
2)Set	Display1	to A/D1

Die Sample A/D Aktion nimmt sofort einen einzelnen Wert des vorgegebenen Kanals auf, welcher dann als Datenwert in diesem AD-Objekt verfügbar ist. Zeile 2 zeigt diesen Wert im Display1 an.

Kanallisten

A/D Aktionen benutzen eine Kanal-Liste als Parameter. Die Liste kann einen oder mehrere Kanäle umfassen. Die Kanalnummern beginnen bei 0. Für die Abfrage mehrerer Kanäle werden die Kanalnummern mit Komma voneinander getrennt.. In der nachfolgenden Aktion (Zeile 1) wird beispielsweise jeweils ein Wert von Kanal 0,1 und 3 aufgenommen:

```
1)Sample A/D      A/D1          once, channel(s)="0,1,3"
2)Output to       File1         with A/D1, term.=CRLF
```

Eine aufeinanderfolgende Reihe von Kanalnummern kann angegeben werden, indem Start- und Stopkanal, getrennt von 2 Punkten angegeben werden:

```
1)Sample A/D      A/D1          once, channel(s)="0..2"
```

Die Kanal-Liste kann zusätzlich verschiedene Verstärkungsfaktoren für jeden Kanal festlegen. Diese werden hinter den Kanalnummern in Klammern angegeben.
In dieser Aktionszeile wird Kanal 1 mit Verstärkung 10, Kanal 2 mit den gegenwärtigen Voreinstellungen und Kanal 3 mit 100facher Verstärkung aufgenommen:

```
1)Sample A/D      A/D1          once, channel(s)=
                  "1(10),2,3(100)"
```

Einige A/D-Meßkarten beschränken sich auf fortlaufende Kanalnummern oder erlauben keine unterschiedlichen Verstärkungsfaktoren pro Kanal.
Bei anderen Karten wird die Abtastrate bei zeitgsteuerter Datenaufnahme durch nicht fortlaufende Kanalnummern und Veränderung der voreingestellten Verstärkung beschränkt.
Die unterschiedlichen Eigenschaften der einzelnen A/D-Karten sind in einer *.DOC Datei beschrieben.
Eine Kanal-Liste kann auch die Werte 0..10 enthalten, dies bedeutet, daß die Kanäle 0-10 verwendet werden.

Zeitgesteuerte Erfassung

Das Aufnehmen mehrerer Meßpunkte mit einer bestimmten Abtastrate geschieht mit der ‘Acquire A/D’ Aktion.

Diese Action Liste nimmt 100 Meßwerte auf und zeigt das Ergebnis in einer Grafik an:

1)Acquire A/D	A/D1 rate=1000 Hz, channel(s)=2	# samples=100,
2)Draw graph	Graph1	with A/D1

Die Aktion ‘Acquire A/D’ erlaubt die Anzahl der aufzunehmenden Meßpunkte, die Abtastrate in Hertz und die Kanalliste anzugeben. In dem Eingabefeld ‘rate= Hz’ kann eine Zahl, ein Objekt oder das Wort ‘external’ stehen, womit zur Datenaufnahme der externe Takteingang der A/D-Karte genutzt wird.

Hintergrunderfassung und A/D-Ereignisse

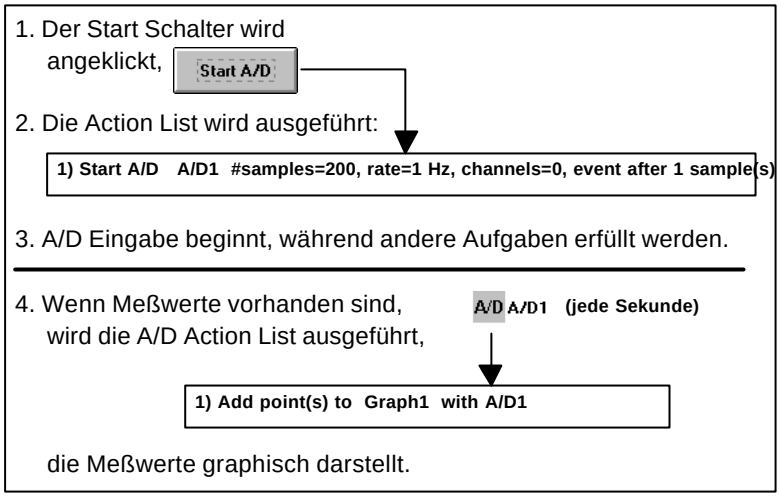
Die ‘Acquire A/D’ Aktion wartet, während die Meßpunkte aufgenommen werden. Eine langsame Abtastrate oder eine große Anzahl von Meßpukten nimmt unter Umständen längere Zeit in Anspruch. Währenddessen kann im Computer nichts anderes ausgeführt werden.

Soll in der Zeit einer A/D Messung eine andere Aufgabe ausgeführt werden, verwendet man die ‘Start A/D’ Aktion.

```
1)Start A/D     A/D1            # samples=100,
                rate=1000 Hz,
                channel(s)=2, event after       "all"
samples
```

Diese Aktion startet die Datenaufnahme, geht jedoch zur nächsten Befehlszeile, ohne auf das Ende der Erfassung zu warten. Ist ein Meßpunkt oder eine bestimmte Anzahl von Meßpunkten aufgenommen, löst das ein Ereignis (EVENT) aus und die ‘action list’ des A/D Objekts wird ausgeführt.

Hier zeigt das Diagramm die Folge der Aktionen:


1. Der Start Schalter wird
angeklickt,
Start A/D
2. Die Action List wird ausgeführt:
1) Start A/D A/D1 #samples=200, rate=1 Hz, channels=0, event after 1 sample(s)
3. A/D Eingabe beginnt, während andere Aufgaben erfüllt werden.
4. Wenn Meßwerte vorhanden sind,
wird die A/D Action List ausgeführt,
A/D A/D1 (jede Sekunde)
1) Add point(s) to Graph1 with A/D1
die Meßwerte graphisch darstellt.

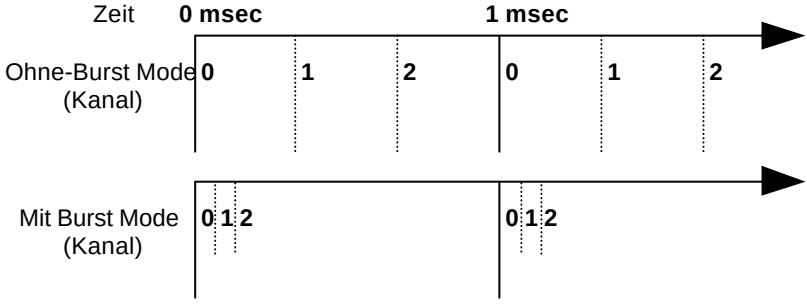
Der ‘event after’ Parameter in der Aktionszeile bestimmt, wann die "events" bereitgestellt werden, um die ‘action list’ des A/D Objekts zu starten. Der Gebrauch einer Zahl oder des Wortes "all" sind möglich. "Event after all" startet die Action Liste, nachdem alle Meßwerte aufgenommen wurden

+ Es ist nicht empfehlenswert, die Daten des A/D Objekts in der gleichen Action Liste, die diese Datenaufnahme startet, zu verwenden. Die ‘start A/D’ Aktion initialisiert lediglich die Datenaufnahme, wartet aber nicht auf das Ende der Erfassung.

‘Events after 1’ eignet sich für niedrige Abtastraten (bis zu ein paar Hertz), wenn nach jeder Messung die Werte sofort bearbeitet und angezeigt werden sollen .’Events after all’ ist für sehr schnelle Datenaufnahme sinnvoll, oder wenn die Bearbeitung nach Verfügbarkeit aller Daten erfolgen soll, z. B.Kurvenglättung, Frequenzanalyse etc.

Erfassungsrate, burst modus, und Grenzen

Die Abtastrate ist die Frequenz, mit der jeder Kanal abgetastet wird. Ein Beispiel : Ist der Wert im Parameterfeld ‘rate’ 1000 und enthält die Kanalliste drei Kanäle, so wird jeder Kanal 1000 mal in der Sekunde abgefragt. Die A/D Karte muß daher insgesamt 3000 Werte in der Sekunde aufnehmen und wandeln, zusätzlich zu der Zeit, die benötigt wird, zwischen den Kanälen zu schalten. Die Aufnahme mehrerer Kanäle kann auf zwei Arten erfolgen: ‘burst mode’ oder ‘non-burst mode’. Mit ‘burst mode’ wird zwischen den Kanälen so schnell wie möglich umgeschaltet, gleichgültig welche Abtastrate eingestellt ist. Ohne ‘burst mode’ werden die Kanäle in gleichmäßigen Zeitabständen aufgenommen. Folgende Abbildung zeigt die zeitliche Abfolge einer Aufnahme von drei Kanälen, mit 1000 Hz abgetastet mit und ohne ‘burst mode’.



Einige A/D Karten haben keinen ‘burst mode’, oder ermöglichen ‘burst mode’ nur mit besonderen Datentransferraten, die möglicherweise das Maximum der Abtastrate beschränken.

Zu jeder A/D-Wandlerkarte wird die Summenabtastrate (maximale Abtastrate) angegeben. Werden mehr als ein Kanal benutzt, errechnet sich die erreichbare Abtastrate pro Kanal, indem man die maximale Abtastrate durch die Anzahl der Kanäle teilt.

A/D Daten: Listen und Vektoren

Sobald eine Serie A/D Wandlungen aufgenommen oder mehrere Kanäle benutzt werden, ist es zur Weiterverarbeitung notwendig, ebenso viele Spannungspunkte der Anwendung zu übermitteln. Das A/D Objekt stellt bei Verwendung mehrerer A/D Kanäle eine Zahlenliste zur Verfügung. Ist die Einstellung des Parameters für "sampling event" größer als 1, bekommt man einen Wertevektor. Beides kombiniert ergibt eine Liste mit Vektoren.

Das ist einfacher, als es klingt. Eine Werteliste ist wie eine Spalte mit Zahlen, und ein Vektor ist wie eine Reihe. Die A/D Werte stehen also in Form einer Tabelle zur Verfügung, die abhängig von der Anzahl der Kanäle und der Abtastungen aus mehreren Spalten und Reihen besteht.

Diese Aktion:	ergibt diesen Datentyp:
Sample channels=0 **number** example: **4.582**	
Acquire #samples=10,channels=0	**vector of number (dim. 10)**

Beispiel:

3.47
3.58
3.43
3.23
2.14
2.21
1.21
0.23
-0.23
1.23

Sample channels=0,1 **list of: number, number**

Beispiel: **3.27 , -0.32**

Acquire #samples=10, channels=0,1 **list of: vector, vector**

Beispiel:

3.43	**.**	**3.43**
3.47	**,**	**3.47**
3.58	**,**	**3.58**
3.23	**,**	**3.23**
2.14	**,**	**2.14**
2.21	**,**	**2.21**
1.21	**,**	**1.21**
0.23	**,**	**0.23**
-0.23	**,**	**-0.23**
1.23	**,**	**1.23**

Verwenden andere Objekte diese Daten, so geschieht die Abarbeitung dieser Listen und Vektoren automatisch. Zum Beispiel: Die nachfolgende Action Liste schreibt unter Benutzung des 'File object' Werte in eine Datei. Kanäle sind in Spalten angeordnet und die einzelnen Meßwerte ergeben Reihen, so daß diese Datei gleich ausgedruckt oder in eine Tabellenkalkulation eingewiesen werden kann:

1)Output to File1 from A/D1

Wenn Sie die Daten graphisch darstellen wollen, behandelt das Graph Objekt jede Position der Liste (jeden Vektor) als einzelne Linie. Somit können die einzelnen Kanäle korrekt dargestellt werden:

1)Draw Graph Graph1 with A/D1

Im folgendem Beispiel wird ein ‘Math object’ zur Skalierung und Addieren eines Offsets benutzt.

```
1)Calculate     Scaled          with rawdata=A/D1,
                scale=2.5,
                offset=-1.8
```

Stoppen der Erfassung

Zum Beenden einer Hintergrundmessung, wird die ‘Stop A/D’ Aktion verwendet.

```
1)Stop A/D      A/D1
```

Verstärkung

Viele A/D Wandlerkarten verfügen über Software einstellbare Verstärkungsfaktoren. Höhere Verstärkungen werden für kleine Eingangsignale verwendet, weil der Bereich und die Auflösung des A/D Wandlerbausteins auf der Karte fest ist. TestPoint erlaubt es, den Verstärkungsfaktor mit einer Aktionszeile einzustellen, bevor die Messung gestartet wird:

1)Set A/D gain	A/D1	to 10
2)Sample A/D	A/D1	once, channel(s)=2

Die Verstärkungen können, wie schon beschrieben, auch in der Kanal Liste angegeben werden. Ist eine Verstärkung für eine A/D Karte nicht zulässig, erscheint eine Fehlermeldung. Die Möglichkeiten jeder A/D Karte, die von TestPoint benutzbar ist, werden in einer Datei mit der Endung *.DOC im TestPoint Verzeichnis beschrieben.

TestPoint verrechnet automatisch die Verstärkung in die zurückgegebenen Spannungswerte. Ist zum Beispiel die Eingangsspannung 0,01 Volt und die Verstärkung 100 (dadurch wandelt die A/D Karte ein 1 Volt Signal), erhält man im TestPoint den Wert 0,01 Volt (der Originalwert)!

Triggerung

Solange nichts anderes eingestellt ist, beginnt eine Messung sofort nachdem die ‘Start’ oder ‘Acquire Aktion’ ausgeführt wird. Häufig ist es von Vorteil, die Messung zu starten, nachdem eine Triggerbedingung erfüllt ist. In TestPoint können verschiedene Triggerarten gewählt werden. Das folgende Beispiel benutzt den digitalen Trigger, welcher einen externen digitalen Hardware Eingang auf der A/D Wandlerkarte verwendet.

```
1)Set A/D trigger digital    A/D1
2)Start A/D          A/D1              # samples=100,
                     rate=1000 Hz,
                     channel(s)=0, event after 1
   samples
```

Dieses Beispiel arbeitet mit dem Analog Trigger, welcher einen A/D Kanal solange abtastet, bis dieser die angegebenen Einstellungen erfüllt:

```
1)Set A/D trigger analog              A/D1        channel=3, level=2.5,
                     polarity=+,
                     mode=edge, hysteresis=0.1
2)Acquire A/D        A/D1             #samples=100,
                     rate=1000 Hz
                     channel(s)=0
```

Auf eine Flanke zu triggern (Edge-triggered), ist die einfachste und zuverlässigste Art, da der Übergang von einem unteren zu einem oberen Level oder umgekehrt betrachtet wird.

Pre-trigger

Einige Karten sind in der Lage, ein Hardware Trigger Signal zu erkennen, während sie analoge Daten empfangen. In diesem Fall nehmen sie sowohl vor als auch nach dem Trigger Daten auf, was auch Pre-Triggering genannt wird. Die Karten der DAS-1800 Serie von Keithley/Metrabyte haben z. B. diese Fähigkeit.

Wenn Sie pre-triggern, brauchen Sie eine bestimmte Anzahl von Meßwerten ("continuous" ist nicht erlaubt). Sie wählen wie oben beschrieben eine Triggerart aus. Dann wählen Sie aus allen Meßwerten die Anzahl aus, die Sie vor dem Triggern bekommen möchten, indem Sie die "other A/D commands" Aktion nutzen, wie z. B.

1)Set A/D trigger digital	A/D1	
2)Other A/D command	A/D1	command word="pretrigger", value=100
3)Acquire A/D	A/D1	#samples=1000, rate=10000 Hz, channel(s)=0

Diese Aktions Liste wird 1000 Meßwerte aufnehmen, wenn der digitale Trigger kommt. 100 dieser Meßwerte werden vor dem Trigger erfaßt, die restlichen 900 danach.

Weitere A/D Befehle

Einige A/D Karten haben außer den Grundfunktionen, die von den TestPoint Actions Acquire, Start, Stop, Set trigger, Sample, und Set gain zur Verfügung gestellt werden, noch andere Funktionen. Die "Other A/D command" Aktion ermöglicht den Zugriff auf diese Funktionen.

"Other A/D command" verlangt nach 2 Parametern. Der erste ist ein Befehlswort. Das gültige Befehlswort ist von der jeweiligen Karte abhängig, die Sie benutzen und ist im Readme File zu finden. Der zweite Parameter ist ein Wort und kann entweder eine Zahl oder ein String sein.

Ein Beispiel hierfür ist das im vorhergehenden Abschnitt beschriebene Pre-Triggering.

Ein anderer Befehl, der mit einigen Karten verfügbar ist, erlaubt die sogenannte simultaneous-sample-and-hold Funktion:

1)Other A/D command	A/D1	command word="ssh", value=1

Raw binary data mode

Das A/D Objekt hat ein Setting für "raw mode", das es Ihnen ermöglicht, analoge Eingangsdaten zu empfangen, ohne sie in Volt umzurechnen. Diese Funktion ist vor allem dann nützlich, wenn die Geschwindigkeit der Datenaufnahme entscheidend ist, da der Umrechnungsprozeß in Volt einige Zeit in Anspruch nehmen kann.

Wenn Sie den Raw Mode einsetzen, ist der Datenwert des A/D Objektes anstatt des üblichen Vektors oder einer Zahl ein **String** Datentyp, dessen Bytes die Binärdaten der A/D Karte sind.

Sie können diesen String nicht direkt für mathematische Berechnungen oder Graphen verwenden. Sie **können** ihn jedoch in einer Datei speichern, indem Sie ein File Objekt benutzen. Das erlaubt Ihnen, mehr Daten zusammenzutragen, als in den Speicher passen, und sie in einer Datei zu speichern. Währenddessen wird die maximal mögliche Geschwindigkeit aufrechterhalten.

Pushbutton action list:		
1)Start A/D	A/D1	#samples=1E6, rate=10000, channel(s)="0,1", samples/event=10000
A/D action list:		
1)Output to	File1	with A/D1, term.=none

Um die Daten des Raw Data Mode zu nutzen, müssen sie diese in Volt umrechnen, indem sie die "Convert to Volts" Aktion des A/D Objektes auswählen. Übergeben Sie dieser Aktion die String Daten zusammen mit der Original Kanalliste, die bei der Datenerfassung verwendet wurde. Diese Funktion wird oft verwendet, indem die gemessenen Daten von einem File geladen werden, nachdem der Hochgeschwindigkeits Disk-logging-Teil Ihrer Applikation beendet ist.

1)Input from	File1	up to 32768 bytes, stop on __
2)Convert to volts	A/D1	raw data=File1, channel(s)="0,1"
3)Draw graph	Graph1	with A/D1

Nachdem die "Convert to Volts" Aktion ausgeführt ist, ist der Datenwert des A/D Objektes der normale Datenwert von Zahlen und Vektoren oder Listen von Vektoren.

+ Da der String in TestPoint maximal 65534 Bytes lang sein kann, können Sie "Samples/event" nicht höher als 32767 in der "Start A/D" Aktion einstellen.

D/A

Das D/A Objekt ermöglicht die Wandlung von Digital zu Analog. Jedes D/A Objekt repräsentiert eine D/A Wandler Karte. Für solche, die sowohl A/D als auch D/A Wandler auf einer Karte haben, kann ein A/D und ein D/A Objekt mit der gleichen Kartennummer (board number) verwendet werden.

Das A/D Objekt unterstützt eine willkürliche Anzahl von Kanälen, die abhängig von der Hardware ist.

Einzelne Ausgabewerte

Zur Ausgabe eines einzelnen Spannungswertes verwendet man die 'Output' Aktion:

```
1)Output D/A        D/A1              once, channel=0, value=1.5
```

Zeitgesteuerte Ausgabe

Zur Ausgabe einer Reihe von Spannungswerten auf einem oder mehreren D/A Kanälen mit einer vorgegebenen Taktrate, wird die Start D/A Aktion verwendet

```
1)Start D/A          D/A1              rate=100 Hz,
   channel(s)="0,1",
                     values=File1,
                     mode="continuous"
```

Der ‘values’ Parameter in dieser Aktionszeile muß ein Objekt sein, das einen Zahlenvektor beinhaltet. Die Anzahl der Elemente in diesem Vektor bestimmt die Zahl der verschiedenen Werte, die von dem D/A Kanal ausgegeben werden.

Der ‘mode’ Parameter kann ‘continuous’ oder ‘once’ sein. Mit der Einstellung ‘continuous’ werden die Ausgabenwerte wiederholt verwendet, um eine periodische Kurve am analogen Ausgabekanal zu erzeugen..

Ist ‘continuous’ ausgewählt, so kann die Ausgabe durch die ‘Stop D/A’ Aktion beendet werden.

Hinweis: Viele Einsteckkarten, die A/D und D/A Wandler enthalten, können nicht gleichzeitig getaktete Ausgabe und Eingabe durch-führen. Bei diesen Karten stoppt die Start D/A-Aktion eine gerade laufende A/D Messung.

Linienschreiber

Eine häufige Anwendung analoger Messungen ist das Aufnehmen eines oder mehrerer Kanäle und das Anzeigen der Meßwerte in einem sich ständig erneuernden, bewegenden graphischen Diagramm.

TestPoint macht dies durch die Hintergrund A/D Messung, Ereignissteuerung bei Ankunft der Meßwerte, ein Graph Objekt mit Linienschreiber Modus und eine 'Add point(s)' Aktion einfach.

Nachfolgend eine typische Action Liste eines Schalters, der eine A/D Messung startet:

1)Start A/D A/D1 # samples=continuous, rate=1 Hz,
channel(s)=0, event after 1 sample(s)

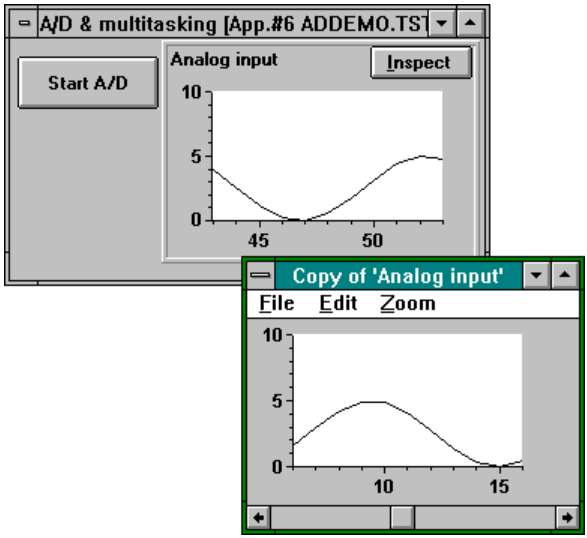
Nachfolgend die Action Liste des A/D Objekts, die den Linienschreiber erneuert.

1)Add point(s) to Graph from A/D1, max # points=500

In den Settings des Graph Objekts muß "strip chart" eingestellt sein. Der Parameter „max # points" kontrolliert die Menge der bereits vorhandenen Informationen, die im Graph für spätere Betrachtungen gespeichert wird.

Hinweis: Die Einstellung auf weniger als 4096 Punkte für die Vorgeschichte erhöht die Leistung des Linienschreibers.

Zu jeder Zeit, während der Linienschreiber in Betrieb ist, kann der ‘Inspect’-Mode benützt werden, um eine Kopie des Graph Objects zu erstellen und vorher gemessene Daten zu betrachten. Der Linienschreiber wird dadurch nicht gestoppt!



Vorsicht, die maximale Linienschreibergeschwindigkeit hängt von der Anzahl der Kanäle, der Computergeschwindigkeit und der Graphikkarte ab! Die obere Grenze ist normalerweise einige Hertz, da der Bildschirmaufbau nicht schnell genug erfolgen kann. **Der beste Weg, den Durchsatz zu erhöhen, ist es, die Anzahl der Samples/event zu erhöhen!** Durch das Anfügen mehrer Meßpunkte an die Grafik erreichen Sie wesentlich höhere Gesamtraten. Stellen Sie den Samples/event Parameter auf einen Bruchteil der Meßpunkte ein, so daß nur wenige Male pro Sekunde die Grafik erneuert werden muß.

Das ist alles!

Siehe auch ADDEMO.TST-Beispielprogramm!

Leistung / Geschwindigkeit

Häufig gestellte Fragen über die Leistungsfähigkeit von TestPoint mit A/D-Karten sind diese:

> Kann man 32 Kanäle mit 1 kHz erfassen und speichern? Können Daten von verschiedenen Quellen als Linienschreiber dargestellt werden? Wie steht es mit 4 Kanälen bei 100 kHz, Mittelwertbildung, FFT und Speicherung auf Festplatte?

Kann TestPoint das?

Der erste Teil der Problematik ist die Definition der Fragestellung. Obwohl alle diese Fragen typisch sind, ist die Informationsbasis nicht ausreichend, um eine Antwort darauf geben zu können.

Man muß wissen:

- Wie viele Kanäle sollen eingelesen werden?
- Wie hoch ist die Abtastrate?
- Wie viele Werte sollen eingelesen werden?
- Wie hoch ist die maximale Geschwindigkeit der Karte?

Mit den Antworten auf diese Fragen können Sie eine Go/No-go Arithmetik anwenden. Daten von einem einzigen Kanal einzulesen ist schneller, als Daten von vielen Kanälen einzulesen; und Daten von aufeinanderfolgenden Kanälen zu erfassen ist wesentlich schneller, als Daten von nicht aufeinanderfolgenden Kanälen zu erfassen. Ist die Anzahl der Kanäle mal der Abtastrate kleiner als die maximale Abtastrate der Karte, ist der erste Test bestanden. Wenn nicht, ist eine schnellere A/D-Karte erforderlich. Andernfalls geht TestPoint nicht in die Gleichung ein.

Die nächste Frage ist „Wie viele Daten müssen bei maximaler Abtastrate bearbeitet werden?“. Dies ist in der Regel der größte Stolperstein, weil es hier viele Mißverständnisse gibt. **Meistens ist es nicht notwendig, die Daten während der Erfassung zu analysieren. Es genügt, erst zu messen und anschließend zu analysieren.** Es ist immer möglich, das Problem in zwei Teile aufzuteilen: Erfassung und Analyse. Wenn diese Entscheidung getroffen wurde, ist es eine einfache Rechnung. **Multiplizieren Sie die Anzahl der Meßwerte mit der Anzahl der Kanäle und dann mit 10. Wenn das Ergebnis kleiner ist als die Anzahl der Bytes, die im Speicher zur Verfügung stehen, dann können Sie zuerst messen und dann analysieren.** An dieser Stelle nochmals der Hinweis: Die Geschwindigkeit von TestPoint geht nicht in diese Gleichung ein, sondern nur die maximale Abtastrate der Karte.

Was aber, wenn nicht genügend Speicher vorhanden ist, um die Meßwerte zu speichern? Der einfachste (und wahrscheinlich preiswerteste) Weg ist es, für mehr Arbeitsspeicher zu sorgen. Wenn es zu viele Werte sind, um dies zu einer praktikablen Alternative zu machen, wird es notwendig sein, die Daten auf Festplatte zu speichern. Die Rate, mit der dies möglich ist, hängt stark von der Fragmentierung der Festplatte, den Zugriffs- und Schreibzeiten und den Einstellungen im Windows-Setup ab. Ein typischer gemessener Wert auf einem 486er Computer mit Rohdaten ist 60 kHz (60.000 Werte pro Sekunde). Die besten Übertragungsraten sind mit Rohdaten (unskaliert) zu erreichen (siehe Beispiel ADRAW.TST).

Wenn die Meßaufgabe eine kontinuierliche Abtastung erfordert und die Rate zu hoch ist, um die Daten direkt auf die Festplatte zu schreiben, überdenken Sie die Problemdefinition. Ist es tatsächlich erforderlich, alle Daten abzuspeichern? Wenn nur ein Ergebnis benötigt wird, könnte die Vektormathematik, die TestPoint bietet, eine Alternative sein. Beispielsweise kann ein Mittelwert über tausende Werte mit nur einer Gleichung gebildet werden. Errechnen Sie den Mittelwert und speichern Sie das Ergebnis im Arbeitsspeicher, nicht auf der Festplatte.

Wenn Sie schnelle Datenflüsse auf dem Monitor so, wie sie auftreten, betrachten, sollten Sie nicht einzelne Meßpunkte anzeigen, sondern nur jeden n-ten Punkt. Kontinuierliche Datenerfassung und Analyse hat oft den geringsten Durchssatz zur Folge. Als Daumenregel dazu kann gelten, daß man die Daten in den größmöglichen Paketen verarbeiten sollte. Generieren Sie nicht zu jedem Meßwert ein Ereignis.

Was tun, wenn die Datenraten weit über der Leistungsfähigkeit des PC´s liegt? Wenn ein PC die Aufgabe nicht lösen kann, können es vielleicht zwei. Eine weitere Möglichkeit ist eine externe Datenerfassung oder ein Oszilloskop. Ist die Erfassung einmal beendet, kann TestPoint die gesamte Analyse, Präsentation und Berichtserstellung erledigen, ohne daß für viel Geld zusätzliche Optionen gekauft werden müssen.

Speicherung auf Platte

In einigen Anwendungen müssen Meßwerte von A/D-Karten in einer Datei abgespeichert werden, um sie später zu verarbeiten. Häufig erfaßt man zuerst die Daten und speichert sie dann mit Hilfe des File Objektes auf der Festplatte. Dies funktioniert, solange man eine bekannte Anzahl von Meßwerten hat, die in den Arbeitsspeicher des Computers passen.

Wenn eine kontinuierliche Abspeicherung bis zu einer Abbruchbedingung benötigt wird, kann das „Disk Logging“ verwendet werden. Erreicht wird dies mit Hilfe eines File Objektes. In den Voreinstellungen des File Objektes sollte die Option ’Update disk on each output’ aktiv sein. Die Action Liste lautet folgendermaßen :

"Start" Schalter Action Liste:

1)Start A/D	A/D1	#samples=1000, rate=2 Hz, channels=0, event after 1 sample(s)

"A/D1" Objekt:

1)Output to	File1	with A/D1, term.=CRLF

Damit wird ein A/D Event nach jedem Meßpunkt ausgelöst, zwei Ereignisse pro Sekunde, und ein Datenpunkt wird nach jedem Ereignis in eine Datei geschrieben.

Schnelle Speicherung auf Platte

Der ‘event after’ Parameter wird groß gewählt, um bei schnellen Abtastraten die Werte auf Festplatte sichern zu können. Es ist empfehlenswert, diesen so zu wählen, daß nur alle ein bis zwei Sekunden ein Festplattenzugriff erfolgt.

Dazu ein Beispiel:

1)Start A/D	A/D1	#samples=10000, rate=1000 Hz, channels=0, event after	1000 sample(s)

und

1)Output to	File1	with A/D1, term.=none

Dies schreibt Gruppen von 1000 Werten auf die Festplatte. Die „Terminator=none“ Einstellung wird verwendet, weil der Vektor mit 1000 Werten im ASCII Format bereits ein CRLF am Ende verwendet.

Noch größere Datenflußmengen erreicht man durch die Verwendung eines binären Datei Formats. Dadurch muß eine geringere Informationsmenge gespeichert werden. Eine ‘float binär’ Zahl einfacher Präzision benötigt lediglich 4 Byte, was geringer ist, als bei einer ASCII-Text-Datei. Binäres Speichern erreicht man durch eine einfache Änderung in der Action Liste des A/D Objekts:

1)Output to	File1	with A/D1, term.=none

Einfach auf den Objekt Namen, in diesem Fall A/D1, einen Doppelklick ausführen und den Datentyp und das Format entsprechend ändern. Das Format ist ‘float’. Ist die Rate zu hoch gewählt, und die Ereignisse treffen schneller ein, als die Daten auf Festplatte geschrieben werden können, so tritt ein ‘overrun’ Fehler auf. Die maximale Abtastrate hängt stark von der jeweiligen Systemkonfiguration ab. Als Beispiel: Mit

einem 486DX2 Prozessor mit 66MHz Taktfrequenz können ca. 20kHz erreicht werden.

Sehr hohe Geschwindigkeiten: Raw Data Speicherung

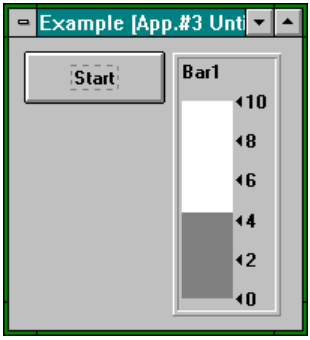
Für die beste Speicherungsleistung auf Festplatte schalten Sie die „Raw data“ Option im Settings Fenster des A/D Objektes ein. **In diesem Modus ist die Anzahl der „Samples/Event“ auf 32768 in der „Sample A/D“ Aktion limitiert.**

Der Raw Data Modus kann um ein Vielfaches schneller sein als der reguläre Modus.

Wenn Sie später diese Daten aus der Datei wieder einlesen, ist es erforderlich, das Eingabeformat als „String“ ohne Endezeichen zu definieren und es in gleich langen Stücken einzulesen maximal 65534 Bytes. Anschließend kann die „Convert to Volts“ Aktion die Raw Data in Spannungswerte konvertieren.

Balken Grafik Analog Anzeige

In TestPoint ist ein "Bar" Objekt (Balkengrafik) enthalten, welches oft mit analogen Eingängen verwendet wird:



"Start" Schalter:

1) Start A/D	A/D1	#samples=continuous, rate=2 Hz, channel(s)=0, event after 1 sample(s)

"A/D1" Objekt:

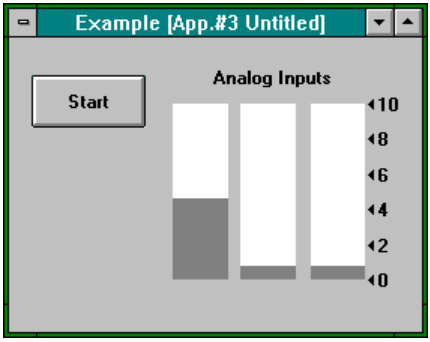
1) Set	Bar1	to A/D1

Trennung der A/D Kanäle und Messungen

Eine Balkengrafik zeigt einen einzelnen Wert an. Um mehrere Kanäle von A/D-Karten darstellen zu können, werden mehrere Balkengrafiken für die verschiedenen Kanäle verwendet.

Die Kanäle werden durch den TestPoint **list**-Typ dargestellt. Um einzelne Positionen auszuwählen und darzustellen, kann man **Groups** und die mathematische Funktion **Select()** verwenden

Positionieren Sie drei Bar Indicator Objekte (Balken Grafiken) auf Ihrem Panel. Markieren Sie diese und wählen Sie das Edit/Group Menükommando. Nun können Sie das Group Objekt in die Action Liste ziehen, und alle drei Balken gleichzeitig verändern. Übergeben Sie die Kanalliste als „index“ Argument:




```
1)Set                Group1          to A/D1  [index="0,1,2"]
```

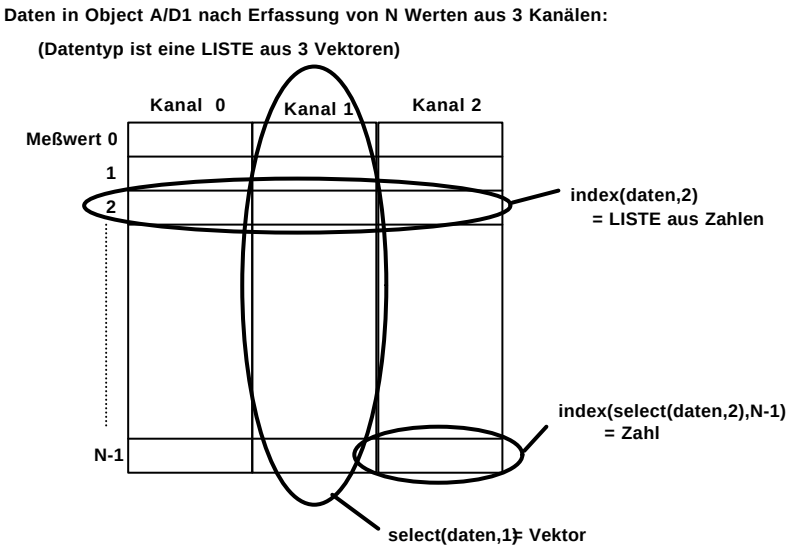
Ein weiteres Beispiel verwendet das Display Objekt und ist als ADGROUP.TST in Ihrem TestPoint Examples Verzeichnis abgelegt. Desweiteren gibt es ein ganzes Kapitel in diesem Buch, das sich mit der Gruppierung von Objekten beschäftigt. Wenn Sie zu Berechnungen Einzelwerte benötigen, verwenden Sie das Math Objekt mit der Formel:

```
select (data, ch).
```

Wenn mehr als ein Meßwert pro A/D Ereignis erfaßt wird, sind die Daten jedes Kanals in einem Vektor abgelegt. Da der Bar Indicator nur Einzelwerte darstellen kann, ist es erforderlich, jeden Wert per Indexfunktion aus dem Feld auszulesen, um Einzelwerte zu erhalten. Verwenden Sie diese Formel:

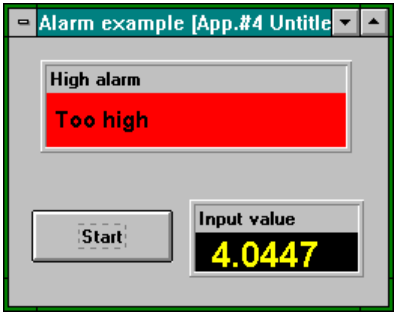
```
index( select(data,ch), 0 )
```

Verwenden Sie häufig mehrere Kanäle mit mehreren Datenpunkten, können Sie Untermengen oder Einzelpunkte auswählen, indem Sie Kombinationen aus den mathematischen Funktionen **select()** oder **index()** verwenden:



Alarmgrenzen überwachen

Eine Grenzwertüberwachung gemessener Werte läßt sich sehr einfach durch ein Math-Objekt und einen oder mehrere Indikator-Objekte bewerkstelligen.



"Start" Schalter:

1)Start A/D	A/D1	#samples=continuous, rate=1 Hz, channel(s)=0, event after 1 sample(s)

"Above limit" Math object:

Formel Settings:		value > limit

"A/D1" Objekt:

1)Set	Input value	to A/D1
2)Calculate	Above limit	with value=A/D1, limit=4
3)Set	High alarm	to Above limit

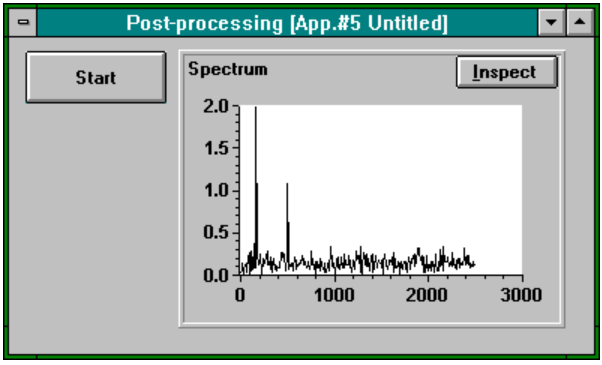
Beachten Sie, daß in diesem Beispiel die voreingestellten Farben und der Text des Indicator Objekts geändert wurden, so daß Rot als Anzeigefarbe benutzt wird, wenn ‘Above limit’ wahr wird.

Werden mehr als ein Kanal abgetastet, so können diese getrennt betrachtet werden. (Siehe vorheriges Beispiel zum Bar Graphen)

Datenerfassung und anschließende Bearbeitung

Soll eine gesamte Datenreihe bearbeitet werden, nachdem die Messung abgeschlossen ist, verwendet man die ‘Acquire A/D’-Aktion.

Ein Beispiel:



Die Action Liste des ‘Start’ Schalters:

1)Acquire A/D	A/D1	# samples=1024, rate=5000 Hz, channel(s)=0
2)Calculate	FFT	with freq=5000, waveform=A/D1
3)Draw graph	Spectrum	with FFT

Das Grafikobjekt muß zuvor als ‘X gegen Y’-Plot definiert sein, weil die FFT Funktion eine Liste zurückgibt, die Frequenz, Amplitude und Phase enthält. Damit nur die Amplitude und keine Kurve für die Phase angezeigt wird, geht man im Graph Objekt zu den ‘Trace settings’, wählt ‘trace 2’ und stellt die Y Achse auf Y2, die ausgeschaltet ist.

Schwierigkeiten

Installationsschwierigkeiten mit A/D-Karten sind nicht außergewöhnlich, weil viele Hardwareeinstellungen (I/O, IRQ, DMA) verändert werden können, die eine Wahrscheinlichkeit der Fehl-Konfigurationen erhöhen.

Wenn Sie Schwierigkeiten haben, von TestPoint aus auf eine A/D-Karte zuzugreifen (weil entweder die Karte nicht gefunden wird, oder ein Fehler bei der Initialisierung der Karte auftritt):

1. Überprüfen Sie auf Hardwarekonflikte.

   Die werkseitigen Einstellungen sind üblicherweise ein guter Ausgangspunkt für die Hardwareeinstellungen. Die I/O Adresse 300 hex ist meist eine gute Wahl bei der Basis Adresse. Der IRQ 7 oder 5 ist meistens OK, obwohl einige Parallelschnittstellen oder Netzwerk-karten damit arbeiten. Als DMA Kanal ist 3, 5, 6 oder 7 in den meisten Fällen OK.

2. Starten Sie das Diagnose Programm des Herstellers.

   Verwenden Sie nach Möglichkeit ein Windows Diagnose Programm anstelle eines unter DOS. Manch-mal existiert ein Harwarekonflikt nur unter Windows, weil die benutzte Hardware unterschiedlich ist.

3. Überprüfen Sie nochmals die Einstellungsschritte.

   Wenn eine Karte als vorhanden erkannt wird, Sie aber trotzdem eine Fehlermeldung von TestPoint bekommen, lesen Sie diese aufmerksam und ziehen Sie ggf. das Hardware Referenzhandbuch zu Rate, um Fehler-nummern zu behandeln.

Wenn die Einstellungen der Karte in Ordnung sind, Sie aber trotzdem keine korrekten Daten bekommen:

Es könnte noch immer ein Hardwarekonflikt vorliegen.

Bei niedrigen Abtastraten werden Interrupts benutzt. Überprüfen Sie den Interrupt Modus damit. Bei höheren Raten wird der DMA Modus verwendet. Versuchen Sie, einen einzelnen Punkt mit „Sample A/D“ zu erfassen. Wenn dies in Ordnung ist, zeitgetaktete Erfassungen nicht funktionieren, liegt wahrscheinlich ein DMA- oder Interruptkonflikt zwischen der Hardware- und der Softwareeinstellung vor.

Spezielle Tips für einige Fälle:

Keithley/Metrabyte: Installieren Sie immer den speziellen DMA Treiber, der für diese Karten entwickelt wurde. Auf einigen Computern funktioniert der „normale“ DMA Treiber, auf anderen gibt es Abstürze, wenn nicht der Keithley Treiber installiert ist. Der Treiber heißt VDMAD.386. Hinweise zur Installation finden Sie in der Datei "ReadMe (A/D Setup)" in der TestPoint Programmgruppe.

Kapitel Rückblick:

- **A/D**: Das A/D Objekt sammelt analoge Daten. Es kann einzelne Daten oder zeitlich getaktete Reihen im Hintergrund aufnehmen.
- **D/A**: Das D/A Objekt steuert analoge Ausgänge. Es kann Einzelwerte ausgeben oder zeitlich getaktete Reihen im Hintergrund.
- **Events**: Die getaktete A/D Aufnahme erzeugt Ereignisse (Events), wenn Daten für die Weiterbearbeitung verfügbar sind. Diese Bearbeitung erfolgt dann in der Action Liste des A/D-Objektes und nicht in der Befehlsfolge, die unmittelbar nach dem 'Start A/D'-Aktion eingeleitet wird.
- **Listen und Vektoren:** Wenn mehr als ein Kanal oder/und mehr als eine Abtastung nach einem A/D Ereignis zurückgegeben werden, so sind diese in Listen (für die Kanäle) und Vektoren (für die Abtastungen) geordnet. Diese Anordnung ist vergleichbar mit den Spalten und Zeilen einer ausgegebenen Datei bzw. der Darstellung mehrerer Kurven in einem Diagramm.

Kapitel 8. GPIB

Was in diesem Kapitel behandelt wird:

- Wie man das TestPoint GPIB Object benutzt.
- Benutzung der GPIB Bibliotheken.
- Grundlegende GPIB Konzepte.

Was ist GPIB?

GPIB steht für „General Purpose Interface Bus“. Es wird ebenfalls HPIB, IEEE-488 oder IEC-625 genannt. GPIB ist ein Standard-Kabel und -Protokoll für die Verbindung von Meßgeräten mit einem Computer.

GPIB ist seit 1975 ein Industrie-Standard (IEEE-488). Es existieren tausende von verschiedenen Geräten, die diese Schnittstelle verwenden.

GPIB stellt eine schnelle Verbindung für bis zu 15 Geräte an einem einzigen Bus zur Verfügung. Die Kabelverbindung wird dabei immer vom Einen zum Nächsten vorgenommen.

Im Jahre 1987 veröffentlichte das IEEE Standard Komitee eine Anpassung des Standards mit dem Namen IEEE-488.2. Diese spezifiziert allgemeine Kommandos und Formate, die zum Datentransfer zu den Geräten verwendet werden. Zusätzlich unterstützt ein Industrie-Konsortium SCPI, das weitere allgemeine Kommandos beinhaltet. Viele der neueren Geräte folgen den neuen Standards.

TestPoint beinhaltet alle Möglichkeiten der GPIB Schnittstellen.

GPIB Geräte Adressen

Jedes Gerät, das an das GPIB-System angeschlossen wird, hat eine Adresse, die eine Zahl zwischen 0 und 30 ist. Diese Adresse darf nur einmal vergeben werden. Meistens wird die Adresse über die Tastatur der Frontplatte der Geräte oder manchmal an deren Rückseite eingestellt.

In TestPoint ist die Geräteadressse, die für das GPIB-Objekt verwendet wird, eine Einstellung (Setting), die vom Benutzer der Anwendung geändert werden kann. Das ist wichtig, weil es dem Anwendungsent-wickler die Möglichkeit gibt, die Anwendung unabhängig von Geräteadressen zu entwickeln.

Einige Geräte haben zusätzlich eine **Sekundäradresse**, die zwischen 0 und 31 liegen kann. Diese Angabe wird in TestPoint als Nachkommastelle zur Primäradresse angegeben.

Das GPIB Object

Das TestPoint GPIB Objekt stellt ein externes GPIB Gerät dar. Wenn in Ihrer Anwendung Zugriff auf mehrere GPIB Geräte benötigt wird, müssen dementsprechend viele GPIB Objekte verwendet werden.

Jedes GPIB Objekt repräsentiert alle Möglichkeiten des angeschlossenen GPIB Gerätes. Es kann Daten an das angeschlossene Gerät senden (Output to), Daten des Gerätes empfangen (Enter from), Statusinformationen erhalten und vieles mehr. Die Daten des GPIB Objektes werden nach jedem Einlesen aktualisiert.

Output

Die ‘Output to’ Aktion sendet Daten an das Gerät. Dieser Befehl akzeptiert jede Anzahl von Parametern, welche nacheinander gesendet werden. Auf diese Art können die an das Gerät gesendeten Daten aus einer großen Anzahl von geräteabhängigen Kommandos und Variablen ausgewählt werden. Einige Beispiele:

```
1)Output to     FnGen       from "Freq " , Frequency , term=LF.
2)Output to     Scope       from ":Chan " , Channel , ":Range " ,
                Range, term=LF.
```

In Zeile 2 werden zwei Strings in Verbindung mit Datenwerten zweier Objekte an das Gerät gesendet. Hat das Objekt ‘Channel’ einen Datenwert von 3 und das Objekt ‘Range’ einen Wert von 4.8, würde der gesamte an das Oszilloskop gesendete String folgendermaßen aussehen:

```
:Chan 3:Range 4.8
```

Die Formatierung der Daten des Parameters in der ‘Output to’-Aktion kann durch einen Doppelklick auf das Parameterfeld und anschließender Auswahl im Datenreferenz-Fenster erreicht werden.

Input

Die ‘Enter from’-Aktion liest die Daten des Gerätes ein. Die maximal einzulesende Anzahl von Bytes und die Abbruchbedingung werden als Parameter spezifiziert. Eine ‘Enter from’ Aktion wird gestoppt, wenn die spezifizierte Anzahl von Bytes eingelesen wurde, das Daten-Endezeichen (end-of-string character) eintrifft oder das GPIB EOI Signal mit einem Byte gelesen wird. Hier ein Beispiel:

```
1)Enter from    DMM     up to 256 bytes,
                        stop on EOS=LF or EOI
```

Diese Action Zeile liest 256 Bytes, beendet aber das Einlesen auch früher, wenn das Daten-Endezeichen (LF) oder das EOI Signal gelesen wird. Das EOI Zeichen wird benutzt, um das Ende einer GPIB Nachricht zu kennzeichnen.

Die Datenformatierung der Eingangsdaten (input data) kann durch einen Doppelklick auf den Namen des GPIB Objektes in der Befehlszeile (z. B. ‘DMM’ in dem Beispiel oben) erreicht werden. Hierzu können Sie in dem Kapitel Datentypen und Formatierung mehr erfahren.

Daten-Endezeichen

Sowohl die ‘Output to’- als auch die ‘Enter from’-Aktionszeilen weisen Parameter für das Daten-Endezeichen (terminating character) auf. Beim ‘Output to’-Befehl wird das Daten-Endezeichen an die zu übertragenden Daten angehängt. Bei der ‘Enter from’-Aktion wird das Einlesen beendet, wenn das Daten-Endezeichen eintrifft, obwohl noch nicht die maximal einzulesende Anzahl von Zeichen gelesen wurden.

Es gibt folgende Daten-Endezeichen (Terminators) :

CR	Carriage Return (ASCII code 13)
LF	Line Feed (ASCII code 10)
CRLF	Beides CR and LF
none	Kein Endezeichen
x	Beliebiges eingegebenes Zeichen

Das übliche, für die meisten GPIB Geräte verwendete, Zeichen ist LF.

GPIB Output

Bei '**Output to'** werden Parameter an das Gerät gesendet. Diese Parameter können ein String, mehrere Konstanten oder Daten eines TestPoint Objekts sein.

Konstanten und Variablen

Häufig sollen Kombinationen von Konstanten und Daten eines Objekts zum Gerät gesendet werden. Dieses Beispiel zeigt die Ausgabe der Frequenz eines Funktionsgenerators:

```
1)Output to     Generator     with "FR", Frequency, "HZ"
                term.=LF
```

Ist **Frequency** ein Dateneingabefeld (Data entry) oder ein Schieberegler (Slider) mit dem Wert 100, sieht die Zeile so aus:

```
FR100HZ
```

Selector-Objekt Werte

Wird ein Selector Objekt zur Auswahl und Steuerung eines GPIB Gerätes verwendet, kann der Selector den Wert für den GPIB String des Gerätes enthalten. Hier ist es der Selector **Function**:

1)Output to Meter with Function, "X"
term.=LF

bei dem der Selector die Einstellungen hat:

labels: DCV,ACV,Ohms
values: F0,F1,F2

Wählt der Anwender die ‘ACV’-Option, steht im Selector der String ‘F1’. Nach Ausführung des Befehls wird ‘F1X’ zum Meßgerät übertragen.

Endezeichen bei der Ausgabe

Die durch die Geräte geforderten Schlußzeichen am Ende der gesendeten Strings sind unterschiedlich. Einige verlangen ein ‘LF’, andere zusätzlich ein ‘CR’. Bei binären Datenübertragungen wird keines der beiden benutzt, weil das letzte Byte das Stringende darstellt.

Das GPIB **EOI** Signal kennzeichnet normalerweise das letzte Byte einer GPIB Information. EOI kann geschickt oder unterdrückt werden. Sie können bei Verwendung mehrerer Befehlszeilen für einen GBIB-String EOI ausschalten und nur am Schluß dieses Strings wieder einschalten.

Hier einige Befehle mit verschiedenen Endezeichen (terminators):

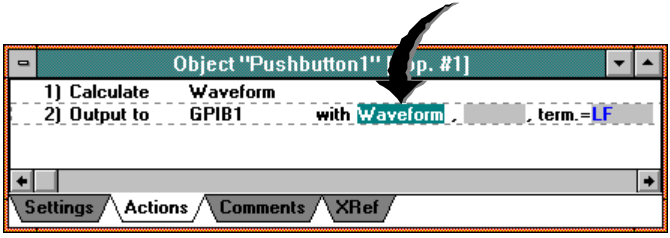
```
1)Output to     Meter          with "F0X", term.=LF,
                send EOI?=1
2)Output to     Supply         with "V", Voltage,
                term.=CRLF, send EOI?=1
3)Output to     Scope          with Waveform,
   term.=none,
                send EOI?=1
4)Output to     Generator      with "WAVE ", term.=none,
                send EOI?=0
5)Output to     Generator      with wave, term.=LF,
                send EOI?=1
```

Formatierung der Ausgabe

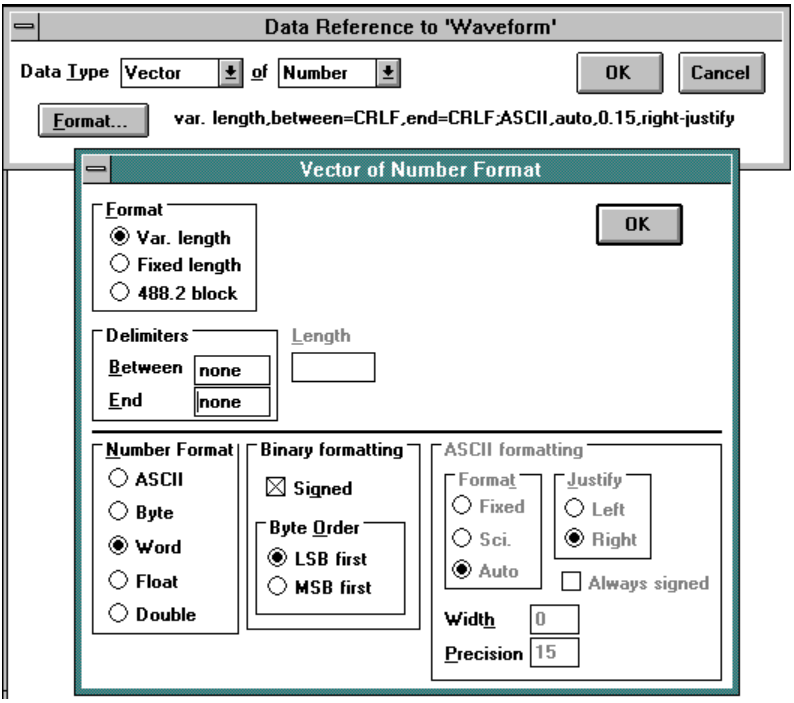
Werden Daten zu einem GPIB Gerät gesendet, müssen sie zu einer Folge von Daten Bytes formatiert werden. Die Strings werden der Reihe der Eingabe nach, Zeichen für Zeichen ausgegeben. Es gibt aber weitere Formatierungsmöglichkeiten.

Ein Zahlenvektor (Vector Of Numbers) kann im ASCII oder Binärformat an einen Arbitrary Kurvenformgenerator geschickt werden. Im Binärformat können die Zahlen aus einem oder zwei Bytes bestehen. Werden zwei Bytes verwendet, können sie ‘Most Significant First’ oder ‘Least Significant First’ sein.

Zur Auswahl des speziellen Datenformates einer Schreibaktion zum Gerät muß der Parameter mit einem Doppelklick angewählt werden:



Danach können sie das erforderliche Format der Daten angeben:



GPIB Einlesen

Bei '**Enter from**' wird ein GPIB Meßwert vom Gerät gelesen. Das Einlesen kann nach einer speziellen Anzahl von Bytes, einem bestimmten Endezeichen, einem Byte, das mit dem EOI Signal versehen ist, oder nach einem Timeout beendet werden.

Beim ersten Einlesen in den Computer erhält das GPIB Objekt den neuen Wert und steht für den weiteren TestPoint Programmablauf zur Verfügung.

Endezeichen beim Einlesen

Viele GPIB Strings enden mit einem Line Feed (LF). Es können auch andere Endezeichen spezifiziert werden.

Werden binäre Daten eingelesen, muß der Terminator auf '**none**' eingestellt werden. Damit wird verhindert, daß bei einem Byte abgebrochen wird, das zufällig die Wertigkeit des Endezeichens besitzt.

Formatierung der eingelesenen Daten

TestPoint formatiert die ankommenden Daten in das entsprechende TestPoint Format. Voreingestellte Formatierungen bestimmen automatisch die Art der ankommenden Daten. Diese Daten beginnen mit Digits als Zahl. Sind mehrere Zahlen vorhanden, wird ein Vektor oder eine Vektorliste (List Of Vectors) gebildet. Andernfalls ergibt sich ein String.

Sie können ein hiervon abweichendes Format mit einem Doppelklick auf den Namen des GPIB Objekts in der '**Enter from**' Befehlszeile auswählen.

Extrahieren von Zahlen

Einige GPIB Geräte senden einen numerischen Meßwert mit vielen Zusatzzeichen vor und nach dem Meßwert. Zum Beispiel kommt vom KEITHLEY Modell 2001 der Wert mit Präfix:

```
NDCV+3.45600E+01
```

Bei einer automatischen Formatierung dieses Wertes würde TestPoint einen String definieren. Spätere Berechnungen mit diesem Format würden wegen der nicht numerischen Zeichen eine ‘0’ zur Folge haben.

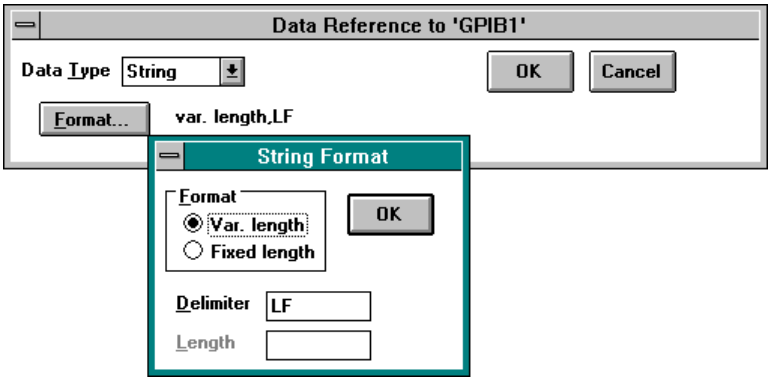
Zum Filtern der reinen Zahl wählen Sie mit einem Doppelklick auf dem Objektnamen in der ‘Enter from’ Aktion **„Number“** aus. Jetzt unterdrückt TestPoint die nicht numerischen Zeichen.

Unterdrückung von LF

Erhalten Sie einen String aus einem Lesebefehl, werden einfach alle vorhandenen Zeichen eingelesen. Damit steht auch ein Endezeichen, z. B. Line Feed (LF) darin. Das Ergebnis sieht dann so aus:

Das Box Zeichen bedeutet, daß es sich um ein nicht druckbares Zeichen, wie z. B. Line Feed handelt.

Zur Unterdrückung dieser zusätzlichen Zeichen wählen Sie mit dem Doppelklick beim Objekt Namen in der Befehlszeile das Datenformat ‘String’aus und definieren das entsprechende Endezeichen (Delimiter):



GPIB Operationen

Zusätzlich zu den ‘Output to’ und ‘Enter from’ -Funktionen hat das GPIB Objekt mehrere spezielle andere Funktionen:

Serial poll

Der Serial Poll liest das Statusbyte des angeschlossenen Gerätes ein. Der Inhalt dieses Statusbytes setzt sich aus einer Vielzahl von Geräteinformationen zusammen. TestPoint aktualisiert beim Serial Poll die Daten des GPIB Objekts.

Parallel poll

Bei einem Parallel Poll wird ein Byte eingelesen, wobei jedes einzelne Bit ein oder mehrere Geräte repräsentiert. Dieses Bit gibt an, ob eines der Geräte einen Service Request vom Computer benötigt.

Trigger

Eine Trigger Operation sendet ein spezielles Kommando an das Gerät, wobei dieses Kommando die Messung startet. Die Art des Triggers muß dem Gerät mitgeteilt werden. Die meisten Geräte weisen unterschiedliche Tiggerarten auf, über die das GPIB Objekt die Messung starten kann.

Clear

Eine ‘Clear’ Operation setzt den Status des Gerätes in seinen ursprünglichen Zustand zurück. Normalerweise ist dieses keine Reset-Funktion, wobei die exakte Reaktion auf dieses Kommando allerdings geräteabhängig sein kann.

Local/Remote

Die Meßgeräte können in den ‘Local’ oder ’Remote’ Modus geschaltet werden. Im ‘Remote’ Modus kontrolliert der Computer das Gerät. Im ‘Local’ Modus kann das Gerät dann wieder über die Frontplatte bedient werden.

Weitere Kommandos

Spezielle Low-Level GPIB Bus Kommandos können mit ‘Transmit GPIB commands’ gesendet werden. Dabei wird der Kommandostring als Parameter gesendet. Dieser String enthält einen oder mehrere der folgenden Ausdrücke, die durch ein Leerzeichen voneinander getrennt sind:

listen 1	Sendet ein GPIB Listen Kommando, bei dem die Ziffer die Adresse des Gerätes angibt. Es können eine oder mehrere Adressen in einem Listenbefehl angegeben werden.
talk 5	Sendet ein GPIB Talk Kommando, bei dem nur eine Adresse erlaubt ist.
sec 8	Sendet ein GPIB Sekundär Adresse. Dieses Kommando kann entweder dem Talk oder Listen Kommando folgen und definiert zusätzlich zur primären Adresse eine sekundäre Adresse.
unt	Untalk schaltet bei allen Geräten den Talk Modus aus.
unl	Unlisten schaltet bei allen Geräten den Listen Modus aus.
mta	My Talk Address setzt den Computer in den Talk Modus.

mla	My Listen Address schaltet den Computer in den Listen Modus.
data 'xyz'	Sendet einige Zeichen als Daten, die in Hochkommata eingeschlossen sind. Sollen Steuerzeichen im ASCII Code gesendet werden, müssen die Hochkommata weggelassen werden (z. B. 13 10). Das Data Kommando kann einen oder mehrere Zeichenketten enthalten.
end	Sendet Line Feed zusammen mit dem EOI Signal.
eoi 10	Sendet das Daten Byte mit dem Kommando zusammen mit dem EOI Signal. In diesem Beispiel wird ein Zeichen im ASCII Code 10 gesendet.
ren	Remote Enable schaltet das Remote Enable Signal beim GPIB Bus ein.
gtl	Go To Local veranlaßt alle gerade als Listener geschalteten Geräte, zur Frontplattenbedienung zurückzukehren.
spe	Serial Poll Enable veranlaßt ein Gerät im Talk Modus zum Senden seines Serial Poll Status Byte.
spd	Serial Poll Disable schaltet den Serial Poll Modus eines Gerätes aus.
ifc	Interface Clear setzt alle Geräte am Bus und das GPIB Interface zurück.
dcl	Device Clear setzt alle Geräte zurück.
sdc	Selected Device Clear setzt die speziellen Geräte zurück, die sich gerade im Listen Modus befinden.

cmd 3	Ein Kommando zum Aussenden eines ASCII Codes im Anschluß an ein GPIB Kommando Bytes.
get	Group Execute Trigger sendet ein GPIB Trigger Byte, das alle aktuellen Listener startet.
llo	Local Lockout schaltet die Frontplattenbedienung aller im Remote Modus befindlichen Geräte aus.

Beispiele

Einige typischen Action Listen für GPIB Geräte:

HP54502 Oszilloskop: Peak-to-Peak Messung

```
1)Output to     Scope     with ":MEASURE:VPP?" ,
   term=LF.
2)Enter from    Scope     up to 256 bytes, stop on
   EOS=LF.
3)Set           Display   to Scope
```

Keithley 2001 Multimeter: DC Volt Einstellung

```
1)Output to     DMM       with ":CONFIGURE:VOLT:DC"
                term=LF.
```

oder mit einem Selector-Objekt ‘Function’ für die Strings VOLT:DC, VOLT:AC je nach Auswahl:

```
1)Output to     DMM       with ":CONFIGURE:" , Function ,
                term=LF.
```

Philips 2812 Netzgerät: Trigger Modus Einstellen und Ausführung

In der ersten Aktionszeile wird der Ausgang auf 10 Ausgangsspannungen eingestellt:

```
1)Output to     Supply    with ":List:Seq:Start 1;Stop
   10;:Init" , term=LF.
2)Output to     Supply    with ":List:Source:Bus" ,
   term=LF.
```

und dann über den GPIB Bus für jeden Schritt getriggert:

```
1) Trigger      Supply
```

GPIB Instrument Libraries

TestPoint wird mit einer Vielzahl von vorgefertigten Objekten für verschiedene GPIB Geräte ausgeliefert. Sie können aus der Bibliothek wählen, welche GPIB Geräte Sie installieren wollen.

Benutzung der Geräte Bibliotheken

1. Um die Gerätebibliothek zu öffnen, benutzen Sie das Kommando 'FILE' aus dem Hauptmenü und anschließend die Option 'Open'. In dem Unterverzeichnis **Library** von TestPoint sind alle verfügbaren Geräte aufgeführt. Die Gerätebibliotheken beinhalten lediglich ein Geräte-Objekt und ein Text-Objekt, welches die Funktion beschreibt.

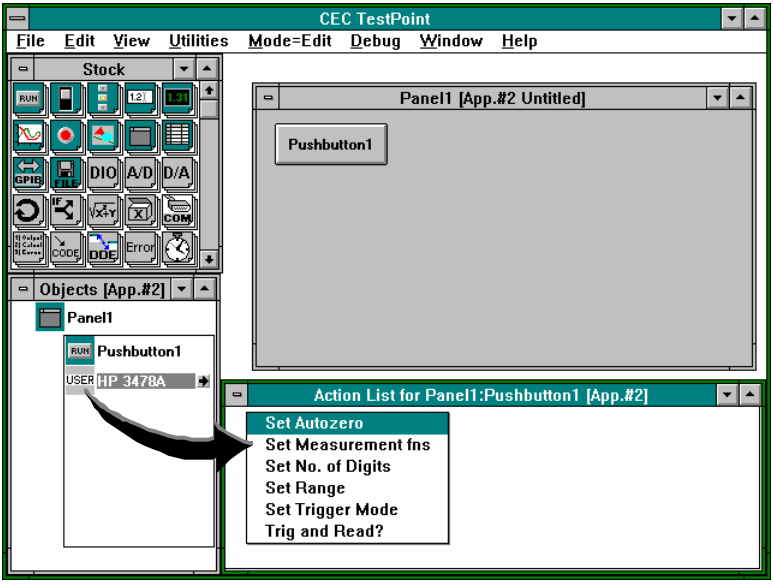
2. Selektieren Sie das Gerätesymbol im Objektfenster und benutzen Sie anschließend aus den 'EDIT' -Menü die Funktion 'Copy', um das Objekt in das Windows Clipboard zu kopieren.

3. Benutzen Sie aus dem Hauptmenü die Option 'FILE' und 'Close', um die Bibliothek zu schließen.

4. Öffnen Sie Ihre bestehende Applikation oder eine neue Anwendung.

5. Benutzen Sie aus dem Hauptmenü das Kommando 'EDIT' und 'Paste', um das kopierte Object in Ihre Anwendung einzufügen.

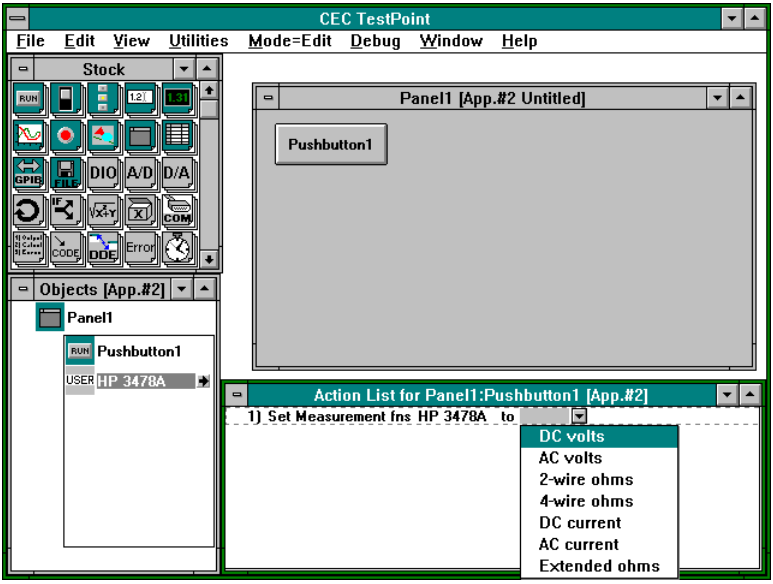
6. Wird das Objekt in eine beliebige Action Liste (so wie die des Pushbutton1 unten) gezogen, öffnet sich ein Menü der verfügbaren Kommandos für das entsprechende Objekt.



Hinweis

Abhängig von den Angaben des Herstellers oder anderen Faktoren, ist es möglich, das nicht alle Kommandos für ein Gerät in der Bibliothek vorhanden sind. Diese Kommandos können mit dem GPIB Objekt implementiert werden.

Wählen Sie ein Kommando, geben Sie die Parameter ein oder benutzen Sie für das ‘drop down’ Menü“ den nebenstehenden Pfeil und wählen Sie den benötigten Parameter aus.



Sollten Sie die Gerätebibliothek benutzen?

Gerätebibliotheken gewährleisten einfach zu benutzende Kommandos für spezielle Geräte, um die Entwicklungszeit für Programme zu reduzieren.

Andererseits werden Ihre erstellten Programme unnötig größer und langsamer, wenn Sie von einem komplexen Gerät nur ein Kommando aus der Bibliothek verwenden. In solchen Fällen ist es sinnvoll, das normale GPIB Objekt mit der ‘Output to’-Aktion zu verwenden. Alternativ können Sie auch ein Objekt aus der Bibliothek verwenden und mit der Option ‘Unlock’ aus der Menüzeile ‘UTILITIES’ den Zugriff auf ein Objekt erlangen. Mit dem Paßwort ‘Library’ können dann überflüssige Kommandos gelöscht werden.

Andere Geräte

Mit der Gerätebibliothek können Sie keine Funktion ausführen, die Sie nicht auch mit dem GPIB Objekt ausführen könnten, sie erspart es Ihnen aber, die geräteabhängigen Kommandos in den Gerätehandbüchern nachzuschlagen und in die Schreib- und Lesefunktionen einzubinden.

Sollte ein Gerät, mit dem Sie arbeiten, nicht in der bei TestPoint mitgelieferten Gerätebibliothek aufzufinden sein, können Sie folgendermaßen vergehen:

a. Benutzen Sie direkt das GPIB Objekt.

b. Rufen Sie bei Ihrem TestPoint-Händler an, und bitten Sie ihn, Ihnen eine Liste der neuesten Bibliotheken zuzusenden.

c. Erstellen Sie sich Ihr eigenes Objekt. Die Objekte in der Bibliothek sind alle sogenannten benutzerdefinierte Objekte. Sie werden aus GPIB- und anderen Objekten zusammengesetzt. Mehr über die Erstellung finden Sie im Kapitel ‘User-defined objects’.

Beachten Sie, daß Gerätebibliotheken keine Hardware Treiber sind. Sie sind mit TestPoint erstellt und verwenden das GPIB-Objekt für alle Gerätekommandos.

- **GPIB Addresses**: Jedes GPIB Gerät hat eine Adresse zwischen 0 und 30. Einige Geräte haben zusätzliche Sekundäradressen. Zur korrekten Funktion müssen alle Geräte am GPIB Bus unterschiedliche Adressen erhalten.
- **Endezeichen**: Informationen vom und zum Meßgerät enden meist mit einem Line Feed (LF) Zeichen. Ab und zu verlangen Geräte nach weiteren Endezeichen, z. B. Carriage Return und Line Feed (CRLF).
- **Ausgabe**: Werden im TestPoint Daten zu einem GPIB Gerät geschickt, können eine beliebige Anzahl von Parametern in einer Befehlszeile eingegeben werden. Werden Werte in einem Parameter abgeschlossen, erscheint ein weiteres freies Parameterfeld für weitere Eingaben.
- **Eingabe**: TestPoint liest Daten von einem Gerät solange ein, bis eine bestimmte Anzahl von Bytes, ein Endezeichen oder ein EOI erfolgt.
- **Daten Formatierung**: Die voreingestellte Formatierung von eingehenden Daten braucht nicht unbedingt verändert zu werden. Soll eine Änderung des Datenformates für Einlese- oder Ausgabeaktionen vorgenommen werden, führen Sie einen Doppelklick auf das Parameterfeld mit den Ausgabedaten (Schreibaktion) oder den Namen des GPIB-Objektes, von dem die Daten eingelesen

werden sollen, aus. So gelangen Sie in das Fenster zum Einstellen des Datenformates.

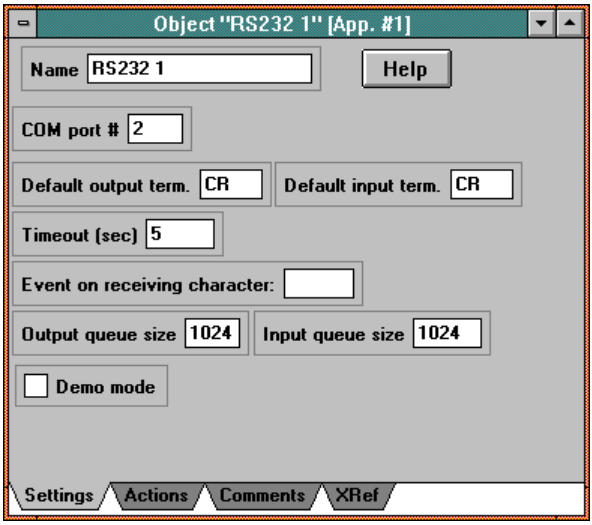
Kapitel 9.
Serielle Kommunikation

Was in diesem Kapitel behandelt wird:

- Wie das RS-232 Objekt verwendet wird.
- Serielle Gerätekonfiguration und Verdrahtungsfragen.
- Die verschiedenen seriellen Standards: RS-232, RS-422, RS-485

Einstellungen

Das RS-232 Objekt besitzt viele Einstellungsmöglichkeiten:



Die wichtigste Einstellung ist die Nummer der COM Schnittstelle. Die meisten PC´s haben mindestens die COM1 und COM2 Schnittstelle, manche sogar noch mehr. Es kann ebenfalls die Timeout-Periodendauer für die Ein- und Ausgaben angegeben werden.

Hardware Kompatibilität

Windows unterstützt normalerweise bis zu 4 serielle Schnittstellen, kann aber auch so konfiguriert werden, daß es 9 unterstützt. TestPoint arbeitet mit allen Schnittstellen zusammen, auf die über den Windows Treiber COMM.DRV zugegriffen werden kann.

Einige Multiple-Port Schnittstellen werden mit Treibern ausgeliefert, die die COMM.DRV Datei ersetzen, so daß sie so aussehen wie normale COM-Ports. Die Überprüfung ist: Wenn die COM Schnittstelle mit

Standard Windows Zubehör, wie z. B. das Terminal Programm arbeitet, sollte es auch ohne Schwierigkeiten mit TestPoint arbeiten.

Hohe Geschwindigkeit

Wenn Sie Geschwindigkeiten über 4800 Baud an der seriellen Schnittstelle verwenden wollen, stellen Sie sicher, daß Ihre serielle Schnittstelle auch die richtige Hardware besitzt. Ältere Hardware verwendet u.U. den 8250 UART Chip, der bei höheren Raten unter Windows nicht zuverlässig läuft. Neuere Koponenten, wie der UART 16550 Chip, unterstützen höhere Raten.

Kommunikations Parameter

Serielle Verbindungen haben eine Reihe von Parametern, die gesetzt werden müssen, um sicherzustellen, daß an beiden Enden die Kommunikation korrekt funktioniert. Die primären Einstellungen sind: Baudrate, Anzahl der Datenbits, Parität und Anzahl der Stop Bits.

Man kann diese Einstellungen bei Bedarf im Windows Control Panel verändern und TestPoint wird auf diese Parameter zugreifen. Der Vorteil dieser Variante ist es, daß, wenn Sie die Anwendung auf einem anderen Computer laufen lassen, Sie nur die Windows Einstellungen, nicht aber Ihre TestPoint Anwendung anpassen müssen.

Wenn Sie die Parameter in Ihrer Anwendung einstellen, sind Sie unabhängig von den Windows Einstellungen. Sie können direkt über die „Set mode“ Aktion zugreifen, bevor Sie die Kommunikation öffnen:

1)Set mode of	RS232 1	baud rate=9600, parity=None, bits=8, stop bits=1
2)Open	RS232 1	

Selbstverständlich können Sie diese Parameter per Data-Entry Objekt vom Benutzer eingeben lassen, damit Sie maximale Flexibilität bei der Entwicklung der Anwendung haben.

Verdrahtung

RS-232 definiert viele elektrische Signale. Einige Geräte benutzen nur wenige, einige verwenden alle davon.

Normalerweise wird ein Standard serielles Kabel ausreichen. Gelegentlich ist es jedoch erforderlich, ein spezielles Kable zu erstellen, weil ein Gerät Signale benötigt, die es ansonsten nicht empfängt.

Das am häufigsten verwendete Verdrahtungsmodell ist das sogenannte „Null Modem“. RS-232 Geräte können so konfiguriert werden, daß sie auf Pin 2 oder 3 eines konventionellen 25-poligen Steckers Daten übertragen. Wenn beide Geräte diese Verbindung verwenden, braucht man ein Null Modem, damit die Übertragung auf dem gleichen Pol stattfinden kann. Dabei sind die Leitungen gekreuzt.

Handshake

Einige Geräte unterstützen es, andere benötigen es: das Hardware oder Software Handshake, um Überläufe zu verhindern, bei denen Daten schneller auftreten, als ein Gerät sie verarbeiten kann.

Hardware Handshakes benutzen die Signale RTS und CTS des RS-232 Kabels. Wenn ein Gerät beschäftigt ist und sein Eingabepuffer voll wird, signalisiert es dies durch diese Leitungen, und die Geräte an der anderen Seite warten. Software Handshakes verwenden spezielle Zeichen, die XON und XOFF genannt werden und dann über die normale Datenleitung gesendet werden, wenn der Empfänger beschäftigt ist. Diese Methode funktioniert, solange keine binären Daten gesendet werden.Um eine Kommunikation korrekt aufzubauen, muß bekannt sein, ob und welche Art von Handshake das verwendete Gerät benutzt.

Das Handshake kann im Windows Control Panel eingestellt werden oder mit der „Set handshaking“ Aktion des RS-232 Objektes, die diese Einstellung überschreibt.

Sollten Sie versuchen, Daten zu senden und keine Daten werden gesendet, versuchen Sie das Handshake auszuschalten.

Sie können hier ebenfalls mit dem Windows Terminal Programm experimentieren. Es stellt Einstellungsmöglichkeiten für alle Kommunikations- und Handshakeparameter zur Verfügung.

Puffergröße

Die Einstellungen „Input Queue Size“- und „Output Queue Size“ des RS-232 Objektes steuern die Anzahl der Zeichen, die gesendet oder empfangen werden können, während der PC andere Aufgaben erledigt, ohne daß eine Fehlermeldung erzeugt wird. Die voreingestellten Werte sind normalerweise in Ordnung, können aber zwischen 256 und 8192 Zeichen liegen.

Software Konflikte

Wenn Sie Schwierigkeiten mit der Kommunikation an der RS-232 haben, entfernen Sie Windows „Shell“ Programme, wie z. B. Dashboard, die den normalen Windows Treiber stören können.

Was sind RS232, RS422 und RS485?

Es gibt einige unterschiedliche serielle Kommunikationsprotokolle, die auf den gleichen Signalen basieren: RS-232, RS-422 und RS-485. Alle davon können mit dem TestPoint RS-232 Objekt benutzt werden.

RS-232 ist das gebräuchlichste serielle Protokoll, das von den meisten seriellen Schnittstellen direkt unterstützt wird und in die meisten PC´s eingebaut ist. Die RS-232 unterstützt die Verbindung von zwei Geräten, wobei Daten in beide Richtungen gesendet werden können.

RS-422 ist das gleiche, mit dem Unterschied, daß bei der Verdrahtung verdrillte Leitungen für jedes Signal verwendet werden, um ein Rauschen bei größeren Distanzen zu vermeiden. Wenn Sie eine RS-422 Karte kaufen, arbeitet diese genauso, wie die RS-232, was die Programmierung von TestPoint betrifft.

RS-485 ist das gleiche wie RS-422, außer daß jedes Gerät die Möglichkeit hat, seinen Transmitter zu aktivieren oder zu deaktivieren, so daß mehr als 2 Geräte verbunden werden können. Bei der RS-485 muß der Transmitter aktiviert werden, bevor Daten gesendet werden und deaktiviert werden, bevor Daten empfangen werden können.

Aktivierung und Deaktivierung des Transmitters der RS-485 erfordert zusätzliche Action Zeilen im TestPoint. Verschiedene Hersteller von RS-485 Adapter Hardware verwenden verschiedene Methoden, um dies zu machen. Einige verwenden ein seperates I/O Hardware Port, das mit dem Port I/O Objekt (siehe nächstes Kapitel) angesprochen wird. Andere binden die Transmitter an eine Handshake Leitung, wie DTR, die man mit dem RS-232 Objekt über die „Set DTR“ Aktion steuern kann.

Serielle Ausgaben

Die „Output to“ Aktion nimmt die gegebenen Parameter und überträgt sie zum Gerät. Diese Parameter können Zeichenketten-, Zahlenkonstanten oder Datenwerte von anderen Objekten sein.

Konstanten und Variablen

Häufig sollen Kombinationen von Konstanten und Daten eines Objekts zum Gerät gesendet werden. Dieses Beispiel zeigt die Ausgabe der Frequenz eines Funktionsgenerators:

```
1)Output to     Generator       with "FR", Frequency, "HZ"
                term.=LF
```

Ist **Frequency** ein Dateneingabefeld (Data entry) oder ein Schieberegler (Slider) mit dem Wert 100, sieht die Zeile so aus:

```
FR100HZ
```

Selector-Objekt Werte

Wird ein Selector Objekt zur Auswahl und Steuerung eines GPIB Gerätes verwendet, kann der Selector den Wert für den GPIB String des Gerätes enthalten. Hier ist es der Selector **Function**:

1)Output to Meter with Function, "X"
term.=LF

bei dem der Selector die Einstellungen hat:

labels: DCV,ACV,Ohms
values: F0,F1,F2

Wählt der Anwender die ‘ACV’-Option, steht im Selector der String ‘F1’. Nach Ausführung des Befehls wird ‘F1X’ zum Meßgerät übertragen.

Warten bis zum Ende der Ausgabe

Die „Output“ Aktion besitzt einen Parameter, der sich „Wait for completition?“ nennt. Wenn dieser Parameter ungleich Null (Wahr) ist, wartet die „Output“ Aktion so lange, bis alle Daten gesendet wurden, bevor die nächste Action Zeile ausgeführt wird. Das ist normalerweise das erwartete Verhalten.

Es gibt jedoch auch Fälle, in denen ein Gerät bekanntermaßen langsam beim Empfangen von Zeichen (gerade bei geringen Baudraten) ist, und man die nächsten Aktionen in TestPoint schon ausführen kann, während an der seriellen Schnittstelle langsam die Daten übertragen werden. In diesem Fall setzt man den Parameter „Wait for completition?“ auf Null.

Endezeichen bei der Ausgabe

Die durch die Geräte geforderten Schlußzeichen am Ende der gesendeten Strings sind unterschiedlich. Einige verlangen ein ‘LF’, andere zusätzlich ein ‘CR’. Bei binären Datenübertragungen wird keines der beiden benutzt, weil das letzte Byte das Stringende darstellt.

Hier einige Befehle mit verschiedenen Endezeichen (terminators):

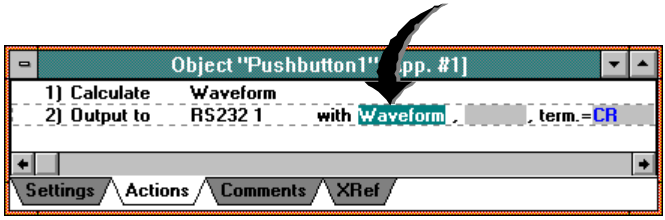
1)Output to	Meter	with "F0X", term.=CR, wait for completion?=1
2)Output to	Supply	with "V", Voltage, term.=CRLF, wait for completion?=1
3)Output to	Scope	with Waveform, term.=none, wait for completion?=1
4)Output to	Generator	with "WAVE ", term.=none, wait for completion?=0
5)Output to	Generator	with wave, term.=LF, wait for completion?=1

Formatierung der Ausgabe

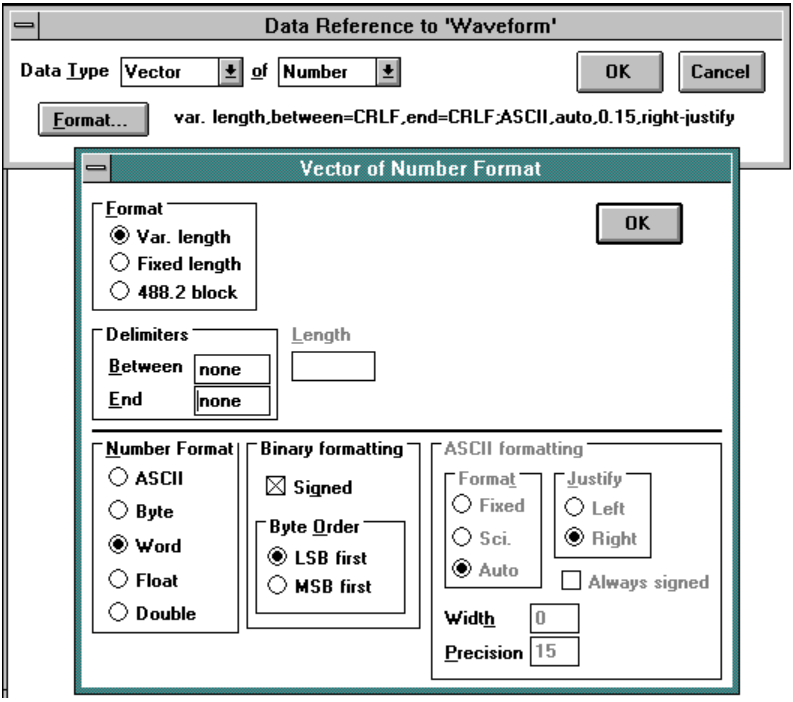
Werden Daten zu einem seriellen Gerät gesendet, müssen sie zu einer Folge von Daten Bytes formatiert werden. Die Strings werden der Reihe der Eingabe nach, Zeichen für Zeichen ausgegeben. Es gibt aber weitere Formatierungsmöglichkeiten.

Ein Zahlenvektor (Vector Of Numbers) kann im ASCII oder Binärformat an einen Arbitrary Kurvenformgenerator geschickt werden. Im Binärformat können die Zahlen aus einem oder zwei Bytes bestehen. Werden zwei Bytes verwendet, können sie ‘Most Significant First’ oder ‘Least Significant First’ sein.

Zur Auswahl des speziellen Datenformates einer Schreibaktion zum seriellen Gerät muß der Parameter mit einem Doppelklick angewählt werden:



Dannach können Sie das erforderliche Format der Daten angeben:



Einlesen der seriellen Schnittstelle

Bei '**Enter from**' wird ein Meßwert vom seriellen Gerät gelesen. Das Einlesen kann nach einer speziellen Anzahl von Bytes, einem bestimmten Endezeichen, einem Byte oder nach einem Timeout beendet werden.

Beim ersten Einlesen in den Computer erhält das RS-232 Objekt den neuen Wert und steht für den weiteren TestPoint Programmablauf zur Verfügung.

Endezeichen beim Einlesen

Viele RS-232 Zeichenketten enden mit einem Line Feed (LF). Es können auch andere Endezeichen spezifiziert werden.

Werden binäre Daten eingelesen, muß der Terminator auf '**none**' eingestellt werden. Damit wird verhindert, daß bei einem Byte abgebrochen wird, das zufällig die Wertigkeit des Endezeichens besitzt.

Formatierung der eingelesenen Daten

TestPoint formatiert die ankommenden Daten in das entsprechende TestPoint Format. Voreingestellte Formatierungen bestimmen automatisch die Art der ankommenden Daten. Diese Daten beginnen mit Digits als Zahl. Sind mehrere Zahlen vorhanden, wird ein Vektor oder eine Vektorliste (List Of Vectors) gebildet. Andernfalls ergibt sich ein String.

Sie können ein hiervon abweichendes Format mit einem Doppelklick auf den Namen des RS-232 Objekts in der '**Enter from**' Befehlszeile auswählen.

Extrahieren von Zahlen - Einige RS-232 Geräte senden einen numerischen Meßwert mit vielen Zusatzzeichen vor und nach dem Meßwert. Zum Beispiel kommt vom KEITHLEY Modell 2000 der Wert mit Präfix:

```
NDCV+3.45600E+01
```

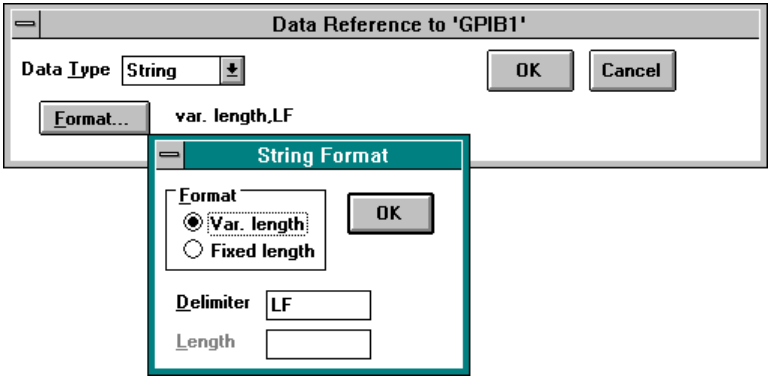
Bei einer automatischen Formatierung dieses Wertes würde TestPoint einen String definieren. Spätere Berechnungen mit diesem Format würden wegen der nicht numerischen Zeichen eine ‘0’ zur Folge haben.

Zum Filtern der reinen Zahl wählen Sie mit einem Doppelklick auf dem Objektnamen in der Enter Aktion **„Number“** aus. Jetzt unterdrückt TestPoint die nicht numerischen Zeichen.

Unterdrückung von LF - Erhalten Sie einen String aus einem Lesebefehl, werden einfach alle vorhandenen Zeichen eingelesen. Damit steht auch ein Endezeichen, z. B. Line Feed (LF) darin. Das Ergebnis sieht dann so aus:

Das Box Zeichen bedeutet, daß es sich um ein nicht druckbares Zeichen, wie z. B. Line Feed handelt.

Zur Unterdrückung dieser zusätzlichen Zeichen wählen Sie mit dem Doppelklick beim Objekt Namen in der Befehlszeile das Datenformat ‘String’aus und definieren das entsprechende Endezeichen (Delimiter):



Sie können CR, LF, oder CRLF als String Delimiter spezifizieren.

Fehler in Eingabeformaten

Sollten Sie Schwierigkeiten mit der Eingabeformatierung haben und die Daten, die Sie empfangen, nicht die erwarteten sein, ist es sinnvoll, in erster Näherung den Datentyp String, Delimiter=none einzugeben. Dabei werden alle Zeichen eingelesen und abgelegt. Sie können angesehen werden, indem Sie das Menükommando View/Data wählen oder die Daten auf die Festplatte schreiben und sich die Datei ansehen.

Wenn das Datenformat einmal exakt bekannt ist, ist es üblicherweise wesentlich einfacher, eine entsprechende Formatierung anzuwenden.

Bedenken Sie, daß Sie, falls Sie die eingelesenen Daten nicht in der Form splitten können, wie Sie es gern hätten, immer das Math Objekt mit Funktionen wie Substr(), usw. verwenden können.

Auf Ereignisse antworten

In vielen Anwendungen will man Ausgaben an ein Gerät an der seriellen Schnittstelle machen und die Antwort zurücklesen.

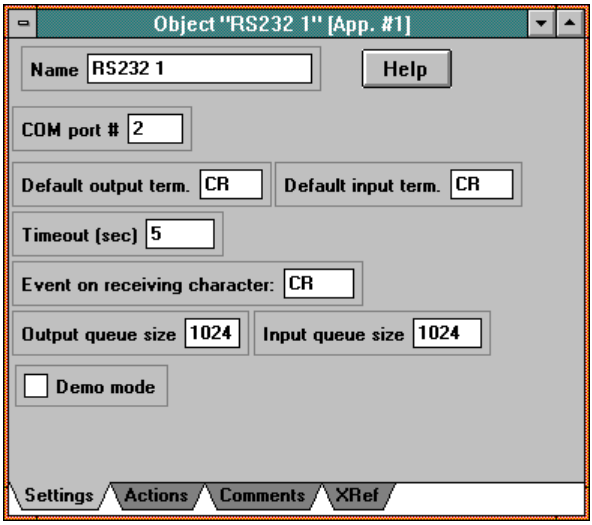
Einige Geräte senden ihre Daten kontinuierlich an den Computer, sobald neue Daten vorhanden sind. Anstatt eine Schleife zu programmieren und darin auf Datenzu warten, kann das RS-232 Objekt so eingestellt werden, daß es automatisch reagiert, wenn Daten vorliegen und der Computer wärend der restlichen Zeit für andere Aktionen zur Verfügung steht.

Um ein Ereignis einzustellen, muß ein „Ereignis-Zeichen“ angegeben werden. Dies ist üblicherweise ein Endezeichen des Gerätes, wie z. B. CR. Es kann jedoch ebenso ein beliebiges anderes Zeichen sein, welches das Gerät einmal pro Meldung mitsendet. Wenn ein Gerät beispielsweise einmal zu Beginn einer Meldung ein Escape (ASCII code 27) überträgt, kann daraufhin ein Ereignis ausgelöst werden.

Um das „Ereignis-Zeichen“ zu setzen, öffnen Sie das Settings Fenster des RS-232 Objektes und geben Sie ein Zeichen in das entsprechende Feld ein.

In der Action Liste verwenden Sie die Open Aktion, wenn Ihre Anwendung startet. Danach wird das RS-232 Objekt seine Action Liste ausführen, wann immer das angegebene Zeichen gesendet wird.

Hier sind typische Settings Fenster und Action Listen:



Pushbutton "Init", eingestellt auf „Execute at initialization“:

1)Open	RS232 1	

RS232 Objekt:

1)Enter from	RS232 1	up to 256 bytes, stop on EOS=CR

Sie können die Spezial Codes CR, LF, CRLF oder TAB als Zeichen zum Auslösen von Ereignissen verwenden.

Um ein nicht druckbares Zeichen zu verwenden, wie ein Escape Zeichen, verwenden Sie beispielsweise folgende Zeile beim Initialisieren der Anwendung:

1)Set RS232 1(Event on receiving character) to "\x1B"

Sie können solche Action Zeilen erzeugen, indem Sie das „Event Character“ Feld aus dem Settings Fenster in die Action Liste ziehen. Die Einstellung „\x1B“ in der Action Zeile verwendet den hexadezimalen Code für diese Zeichen.

Kapitel 10.
Weitere Ein-/Ausgabegeräte

Was in diesem Kapitel behandelt wird:

- Wie man das DIO Objekt benutzt.
- Wie man das Port I/O Objekt benutzt.
- Was für die Steuerung der Hardware zu beachten ist.

Digital I/O

Das TestPoint DIO Objekt  unterstützt jede Hardware Schnittstellen Karte, die den Intel 8255 Digital I/O Chip verwendet. Dies ist der gebräuchlichste Chip für diese Art von Karte.

Digitale Ein-/Ausgabe Karten werden verwendet, um Schalterstellungen zu überprüfen, Relais zu steuern oder externe Schaltkreise mit an/aus Werten zu kontrollieren.

Die Einstellungen, die für TestPoint erforderlich sind, um Digital I/O Karten zu steuern, werden im Anhang dieses Handbuches behandelt. Hauptsächlich besteht die Einstellung aber daraus, die entsprechende I/O-Adresse in der TESTPT.INI Datei und die Anzahl der 8255 Chips auf der Karte anzugeben.

Digitale I/O Ports sind in Gruppen zu je 8 Bit aufgeteilt. Das DIO Objekt stellt die Aus- oder Eingabe numerischer Werte zwischen 0 und 255 für jede dieser Gruppen zur Verfügung. Jeder 8255 Chip besitzt drei dieser 8 Bit Gruppen: A, B und C.

Man definiert das digitale I/O Port in den DIO Action Zeilen durch den Gruppen-Buchstaben. Wenn mehr als ein 8255 Chip verwendet wird (also mehr als 24 digitale I/O Leitungen vorhanden sind), wird ein numerischer Prefix, gefolgt von dem Gruppen-Buchstaben, angegeben, wie im Beispiel unten:

```
1)Output to        DIO1        channel="A", value=130
2)Output to        DIO1        channel="2B", value=4
```

Die „Configure“ Aktion stellt die Auswahl auf Aus- oder Eingabe für jede der 8-Bit Gruppen zur Verfügung. Voreinstellung ist der Eingabe-Modus für alle Gruppen:

```
1)Configure        DIO1        channel="B", mode="output"
```

Konvertierung von und zu Bit-Vektoren

Häufig, wenn man mit digitalen Ein-/Ausgabegeräten arbeitet, ist es nützlich, wenn man zwischen zwei Datenformaten konvertieren kann: Bytes (als Zahlen zwischen 0 und 255) und Bit Mustern (Vektoren, die Nullen und Einsen enthalten).

Die folgenden Formeln erlauben eine einfache Umwandlung zwischen diesen beiden Formaten:

Zahlen zu einem Bit Vektor:

sgn(x and (pow2(ramp(n)))

(konvertiert die Zahl x zu einem Bit-Vektor der Länge n).

Bit Vektor zu Zahlen:

polynomial(2, bitvec)

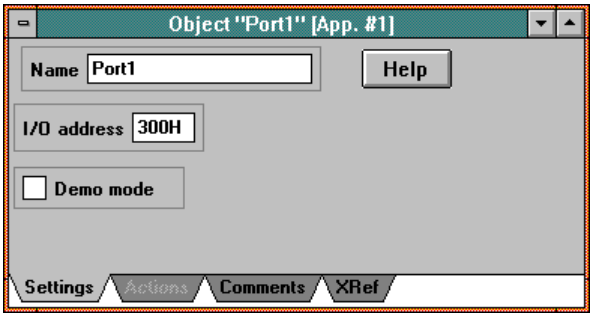
(konvertiert den Bit-Vektor bitvec in eine Zahl).

Port I/O (Direktzugriff auf Hardware)

Mit dem Port I/O Objekt 

können viele verschiedene Typen von Hardware gesteuert werden.

Port I/O gibt den direkten Zugriff auf Hardware Register vieler Einsteck-karten in dem PC frei. Sie stellen die Basis I/O Adresse im Settings Fenster des Objektes ein:



und verwenden dann die Aktionen des Objektes, um Bytes oder Worte von den Ports einzulesen oder auszugeben:

1)Output byte to	Port1	offset=2, value=64
2)Input word from	Port1	offset=4

Der „Offset“ Parameter wird zur Basisadresse addiert, um die Hardware I/O Adresse, die für diese Aktion verwendet wird, zu erhalten.

Counter/Timer Chips und fast alle Arten handelsüblicher oder spezieller Hardware können auf diese Art und Weise gesteuert werden. Voraussetzung dazu ist die Dokumentation der Hardware Register der Karte.

Hinweis: Das Port I/O Objekt kann nicht für den Zugriff auf den System Port unterhalb der Adresse 100 hex verwendet werden.

Erstellung spezifischer I/O Treiber

In den meisten Fällen kann spezifische Hardware mit dem Port I/O Objekt gesteuert werden (siehe vorheriger Abschnitt).

Sollten Sie eine komplexere Hardwarekomponente verwenden, kann es die Mühe Wert sein, einen Treiber in einer konventionellen Programmiersprache, wie C oder Pascal, zu entwickeln und mit Hilfe des Code Objektes in TestPoint einzubinden.

Analyse & Präsentation

Kapitel 11. Mathematik

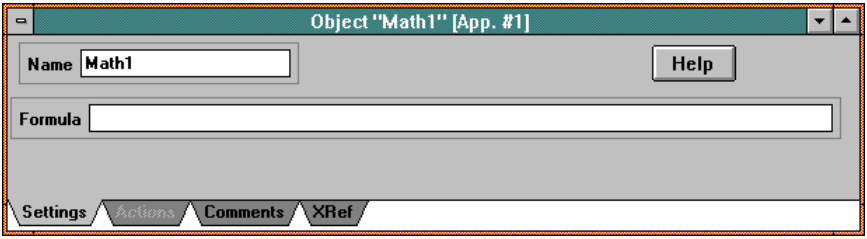
Was in diesem Kapitel behandelt wird:

- Wie das Math Objekt verwendet wird.
- Verwendung der Mathematik in Condition und Case Objekten.
- Anwendung mathematischer Operatoren auf Vektoren, Feldern und Listen.
- Behandlung mathematischer Fehler.

Benutzung der Mathematik

Das Math Objekt kann Formeln aus einer beliebigen Anzahl von Variablen berechnen. Es kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden, ohne notwendigerweise Schleifen zu programmieren.

Wenn ein neues Math Objekt verwendet wird, kann die Formel, die ver-wendet werden soll, im Settings Fenster eingegeben werden:



Wird ein Math Objekt für eine Berechnung verwendet, wird das Ergebnis als Datenwert des Math Objektes gespeichert.Somit arbeitet das Math Objekt mit der Formel **X*2+3** genauso, wie der Ausdruck **Math1=X*2+3** in einer konventionellen Programmiersprache.

Mathematische Funktionen beinhalten sowohl arithmetische, statistische und Zeichenketten-Funktionen, als auch erweiterte Fuktionen, wie digitale Filter und FFT´s. Es sind aber auch Funktionen zur Manipulation verschiedener Datentypen im TestPoint enthalten: Indexverarbeitung von Vektoren oder Feldern und Erzeugung und Auswahl aus Listen.

Eine ausführliche Referenz zu den verfügbaren mathematischen Funktionen im TestPoint enthält das Kapitel „Mathematik Referenz“ in diesem Handbuch, beginnend auf Seite Math-1.

Beispiel

Mit der Formel:

1 / (R*C)

als Formel Einstellung im Settings Fenster kann die Zeitkonstante für ein Widerstands-Kondensator Netzwerk berechnet werden:

1)Calculate Time Constant with R=Resistance, C=Capacitance'

Dabei sind Resistance und Capacitance TestPoint Objekte, wie z. B. Dateneingabe Felder.

Die Previous() Funktion

Oftmals ist eine Funktion zu definieren, die, in Abhängigkeit eines vorhergehenden Wertes des Math Objektes, eine iterative Berechnung ausführt. Dadurch wird die Verwendung eines weiteren Objektes vermieden, in dem das vorige Ergebnis zwischengespeichert wird.

Das Math Objekt unterstützt die **Previous()** Funktion für diesen Zweck. Diese Formel gibt den Wert des Math Objektes vor der erneuten Berechnung zurück.

Initialisierung des Math Objektes

Neben der Berechnung von Formeln kann das Math Objekt auch auf einen Wert gesetzt werden, indem man die „Set“ Aktion verwendet. Dies kann sehr sinnvoll bei der Initialisierung eines Objektes sein, dessen Formel die Previous() Funktion verwendet.

In vielen Anwendungen ist ein Zähler erforderlich, der hochgezählt wird, wenn ein Schalter angeklickt wird, oder wenn ein Durchlauf beendet ist, usw.

Ein Zähler ist ein Beispiel für eine mathematische Berechnung, deren Ergebnis auf dem vorhergehenden Wert der Berechnung basiert. Für solche Fälle ist die „Previous()“ Aktion geeignet. Die „Set“ Aktion initialisiert den Zähler.

Die Formel für den Zähler ist:

previous()+1

Diese Action Zeile wird verwendet, um den Zähler zurückzusetzen oder zu initialisieren:

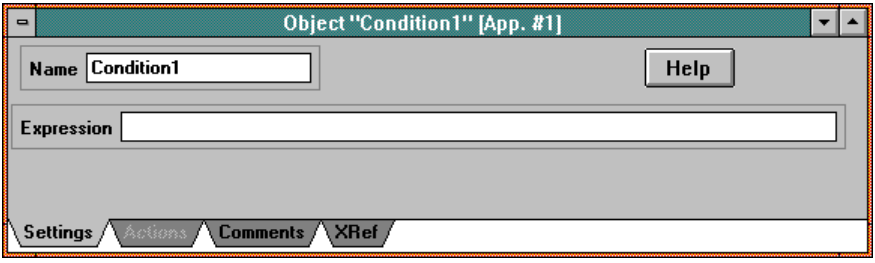
1) Set Counter to 0

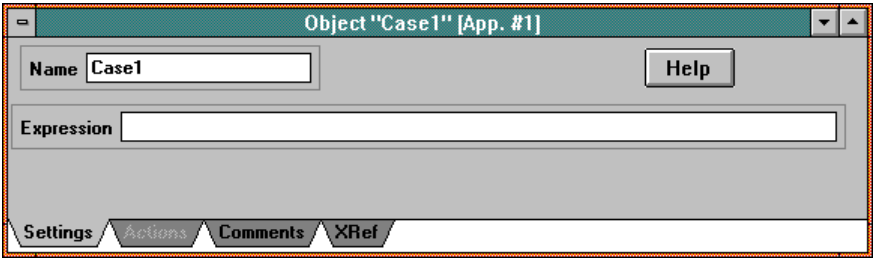
und diese Zeile setzt den Zähler auf den nächsten Wert:

1) Calculate Counter

Mathematik in Conditionals und Cases

Die Conditional und Case Objekte können verschiedenste Aktionen aufgrund des Ergebnisses einer mathematischen Berechnung ausführen. Beide Objekte haben eine Eingabemöglichkeit für einen mathematischen Ausdruck und können die gleichen Funktionen wie das Math Objekt verwenden.





Wenn ein Ausdruck für das Conditional- oder Case Objekt eingegeben wird, ist es meistens ratsam, so häufig wie möglich Variablen zu verwenden. Wenn man beispielsweise eine Bedingung wie diese definiert:

x = 1

können die folgenden Action Zeilen ausgeführt werden, wenn die Werte gleich Eins sind:

1) If/Then Equal-to-one with x=Data-Entry1
2) Execute Action1
3) End Equal-to-one

Wenn man danach nochmals auf Null oder andere Werte überprüfen muß, würde man ein weiteres Conditional Objekt benötigen. Wenn man jedoch statt dessen den Ausdruck:

x = y

verwendet hat, ist es möglich, das gleiche Objekt mehrfach zu verwenden, wie hier:

1) If/Then	x-equal-y	with x=Data-Entry1, y=1
2) Execute	Action1	
3) End	x-equal-y	
...		
10)If/Then	x-equal-y	with x=GPIB1, y=12
11)Execute	Action2	
12)End	x-equal-y	

Anwendung der Formeln auf Vektoren, Listen und Felder

TestPoint unterstützt verschiedene Datentypen: Zahlen, Zeichenketten, Vektoren, Felder und Listen. Die meisten mathematischen Funktionen sind so ausgelegt, daß sie die unterschiedlichen Datentypen akzeptieren und entsprechend damit zusammen arbeiten.

Die arithmetischen Funktionen beispielsweise funktionieren einerseits mit Zahlen, wie man es erwartet. Sie arbeiten aber ebenfalls mit Vektoren, Feldern oder Listen, die Zahlen enthalten. Man kann beispielsweise zwei Vektoren mit einer einzigen mathematischen Operation addieren. Es gibt keine Notwendigkeit für Schleifen, um auf jedes einzelne Element zuzugreifen.

Hier sind einige Beispiele dafür, die zeigen, wie die Addition verschiedene Datentypen unterstützt:

Diese Formel:	ergibt:
3+5	8
3+vector(1,2,3)	vector(4,5,6)
vector(1,2,3)+vector(5,0,2)	vector(6,2,5)
1+list(4,2,3)	list(5,3,4)
list(vector(1,2),5)+2	list(vector(3,4),7)
list(1,2,3)+list(4,3,2)	list(5,5,5)

Diese Möglichkeit von TestPoint, mathematische Funktionen auf verschiedene Datentypen anzuwenden, ist in der Technik als Polymorphismus bekannt und vereinfacht viele Anwendungen dadurch, daß keine Schleifen und Vektor-Indizierungen verwendet werden müssen.

Wenn beispielsweise 3 Eingangskanäle einer A/D-Wandlerkarte jeweils 100 Werte erfassen, besteht der Datenwert des Objektes aus einer Liste mit drei Vektoren, die jeweils 100 Werte enthalten. Wenn diese Werte nun verrechnet werden müssen, mit einem Faktor und einem Offset, ist es in konventionellen Programmiersprachen erforderlich, zwei verschachtelte Schleifen aufzubauen. In TestPoint erledigt dies das Math Objekt mit der Formeleinstellung:

```
input*scale+offset.
```

Dies kann dann, wie hier gezeigt, die Aufgabe in einem Schritt lösen:

1)	Acquire A/D	A/D1	#samples=100, rate=1000 Hz, channel(s)="0,1,2"
2)	Calculate	Corrected	with input=A/D1, scale=2, offset=3

Vektor und Feld Mathematik

Wenn Vektoren oder Felder mit Zahlen verrechnet werden, wird jeder einzelne Wert des Vektors mit der Zahl verarbeitet.

Beispiel: Wenn die Zahl 3 zu einem Vektor addiert wird, wird jeder Wert innerhalb des Vektors um 3 erhöht.

Wenn zwei Vektoren miteinander verrechnet werden, wird jedes Element des Vektors eins zu eins mit dem korrespondierenden Element des anderern Vektors verarbeitet. Dabei müssen die Vektoren die gleiche Länge haben.

Zum Beispiel, die Formel:	ergibt:
vector(1,2,3)+vector(5,6,7)	vector(6,8,10)
vector(1,2,3)+vector(5,6) vector(6,8,3)	
vector(1,2)+vector(5,6,7) vector(6,8)	

Diese Regeln gelten für arithmetische Operatoren, wie +, -, *, /, ^, mod, exp und ebenso für die bitweisen Operatoren, wie and, or, oder xor.

Wenn Vergleiche, wie <, >, usw. im Zusammenhang mit Vektoren verwendet werden, dann ist das Ergebnis ein Indikator dafür, ob die Bedingung für ALLE Elemente des Vektors oder Feldes zutrifft.

Beispiel:	ergibt:
vector(1,2,3) < 4	1
vector(1,2,3) < 3	0
vector(1,2,3) < vector(5,3,6)	1
vector(1,2,3) < vector(5,1,6)	0

Wenn man stattdessen den Vektor elementweise vergleichen möchte und als Ergebnis einen Vektor will, der aufzeigt, ob die Bedingung für das entsprechende Element zutrifft, kann man die Vektoren voneinander subtrahieren und dann mit der **sgn()** Funktion herausfinden, ob das Ergebnis kleiner, gleich oder größer als Null ist. Die **sgn()** Funktion gibt -1 zurück, wenn das Ergebnis negativ ist, 0 bei Null und 1 für positive Werte. Nun kann man die **cliplower()** oder **clipupper()** Funktion verwenden, um die Fälle, für die man sich nicht interessiert, zu beseitigen. Um beispielsweise zwei Vektoren mit einem „>“ zu vergleichen, verwendet man folgende Formel:

```
cliplower( sgn(a - b) , 0)
```

Listen Mathematik

Wenn eine Liste mit einem anderen Datentyp arithmetisch verrechnet wird, wird das Ergebnis von der Operation für jede Listenposition gebildet.

Zum Beispiel:	ergibt:
list(1,2,3) + 3	list(4,5,6)

Wenn zwei Listen arithmetisch verrechnet werden, werden die Positionen einzeln verrechnet. Das Ergebnis besitzt die Elementzahl der Positionen des zweiten Operators (rechte Seite).

Zum Beispiel:	ergibt:
list(1,2,3)+list(0,3,2)	list(1,5,5)
list(1,2,3)+list(0,3)	list(1,5)
list(1,2)+list(0,3,2)	list(1,5,2)

Ändern von Vektoren, Listen und Feldern

Viele Operationen, die man in TestPoint ausführt, wie z. B. die Datenerfassung von A/D-Wandlerkarten, haben normalerweise Vektoren oder Listen mit Vektoren zum Ergebnis. In anderen Fällen kommt es vor, daß man diese Datentypen, z. B. innerhalb von Schleifen, selbst erstellen möchte. Dabei wird man als Anwender von der komfortablen Behandlung dieser Datentypen im TestPoint unterstützt (wie beispielsweise die Darstellung einer Liste von Vektoren in einer Grafik mit einer einzigen Aktion).

Erzeugung eines Vektors

Einige Objekte und Aktionen haben als Ergebnis einen Vektor, wie z. B. die Erfassung von A/D-Daten eines Kanals. Wenn man selber einen Vektor mit Konstanten erzeugen will, verwendet man diese Formel:

vector(1, 2, 3, 4)

Um aus zwei oder mehr Vektoren einen neuen zu erzeugen:

vector(v1, v2)

Aus individuellen Werten wird ein neuer Vektor in einer Schleife erzeugt, indem diese Formel benutzt wird:

vector(previous(), newvalue)

1)	Set	Math-vector	to ____
2)	Linear series	Loop1	from 1 to 100, step by 1
3)	Enter from	GPIB1	up to 256 bytes, stop on EOS=LF
4)	Calculate	Math-vector	with newvalue=GPIB1
5)	End	Loop1	

Zeile 1 initialisiert das Math Objekt durch das Auslassen des Parameters. In der Schleife wird jeder Wert durch die „Calculate“-Aktion in Zeile 4 an den Vektor angehängt. Nach Beendigung der Schleife beinhaltet das Math Objekt einen Vektor mit 100 Werten.

Zerlegung eines Vektors

Ist ein Vektor vorhanden, kann man die Vorteile der vielen Aktionen nutzen, die TestPoint bietet, um mit kompletten Vektoren zu arbeiten: Grafik Erzeugung, Dateiausgabe oder mathematische Operationen. Andererseits gibt es Fälle, in denen man nur einzelne Elemente oder Teile aus Vektoren benötigt.

Um ein einzelnes Element aus einem Vektor zu erhalten, verwendet man diese Formel, die auch ein Teil eines komplexeren Ausdruckes sein kann:

index(vect, i)

Einen Teil eines Vektors, innerhalb eines gegebenen Bereiches, erhält man mit dieser Formel:

subarray(vect, i, j)

Die Größe eines Vektors kann reduziert werden, indem nur jeder n-te Punkt verwendet wird, indem diese Formel angewendet wird:

decimate(vect, N)

Aufbau einer Liste

Der Listen Datentyp erscheint in TestPoint an verschiedenen Stellen, wie beispielsweise bei der Erfassung von A/D-Wandlerdaten mehrerer Eingangskanäle oder beim Einlesen von Daten aus einer Datei mit mehreren Zeilen und Spalten. Eine Liste kann verschiedene Positionen gemischter Datentypen enthalten. Normale Listen, die intern erzeugt werden, haben üblicherweise den gleichen Typ derselben Dimension.

Um eine Liste mit verschiedenen Positionen aufzubauen, verwendet man eine Formel, wie diese:

list(a, b)

Um eine Liste innerhalb einer Schleife aufzubauen, verwendet man die Funktionen **previous()** und **list()** zusammen, wie in diesem Beispiel:

list(previous(), newitem)

1)	Set	Math1	to ____
2)	Linear Series	Loop1	from 1 to 10, step by 1
3)	Acquire	A/D1	#points=100, rate=1000 Hz, channel(s)=0
4)	Calculate	Math1	with newitem=A/D1
5)	End	Loop1	

Nach der Ausführung der Schleife beinhaltet das Math Objekt eine Liste von 10 Vektoren mit jeweils 100 Werten.

Zerlegung einer Liste

Um einen einzelnen Wert aus einer Liste zu lesen, verwenden Sie diese Funktion:

select(list, index)

Für einen Bereich von Punkten verwenden Sie:

sublist(list, i, j)

Sie können die list(), select() und sublist() Funktionen in den verschiedensten Kombinationen verwenden, um neue Listen zu generieren. Wenn Sie beispielsweise eine Position aus einer Liste entfernen wollen, verwenden Sie:

list(sublist(L, 0, index-1), sublist(L, index+1, 9999))

Datentypen konvertieren

Zahlen und Zeichenketten werden von TestPoint normalerweise automatisch umgewandelt. Wenn Sie etwa ein Dateneingabefeld haben, das eine Zeichenkette erzeugt, und Sie verwenden dies in einer arithmetischen Berechnung, wird es zu einer Zahl konvertiert.

Eine Möglichkeit, eine Umwandlung zu einer Zeichenkette zu erzwingen, ist es, einen Null-String Wert anzuhängen, wie hier:

value & ""

Um die Umwandlung in eine Zahl zu erzwingen, addieren Sie eine Null.

Im Referenzteil sind weitere Funktionen zur Konvertierung aufgelistet.

Mathematische Fehler

Das Math Objekt kann eine ganze Reihe von Fehlern beschreiben. Viele davon werden bereits angezeigt, wenn die Formel, die eingegeben wird, fehlerhaft ist. Andere sind Laufzeitfehler, weil die Argumente nicht gültig sind oder eine Ausnahmebedingung, wie die Division durch 0.

In einigen Fällen kann es vorkommen, daß Sie die Fehler selber behandeln wollen, anstatt eine TestPoint Meldung anzuzeigen. Beispielsweise können Sie einen Überlauf während einer bestimmten Berechnung ignorieren und mit einer sehr großen Zahl fortfahren, wenn Sie die Genauigkeitseinbußen akzeptieren, die dabei entstehen können.

Um das zu erreichen, verwenden Sie das Error Handler Objekt in Ihrer Anwendung. Stellen Sie das Objekt so ein, daß es nur auf die von Ihnen gewünschten Fehlercodes reagiert. Dannach können Sie die Aktionen, mit denen Sie den Fehler behandeln wollen, in die Action Liste des Error Handler Objektes. Zum Beispiel:

1) Continue after Overflow error with data=1E300

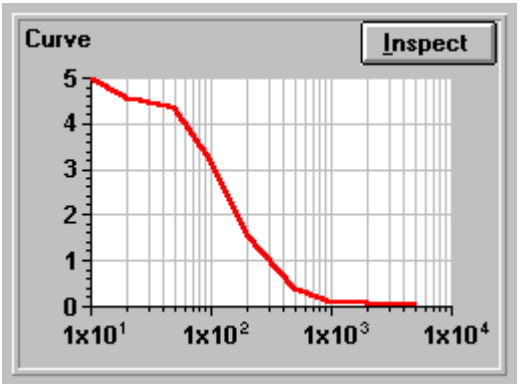
Kapitel 12. Grafik

Was in diesem Kapitel behandelt wird:

- Wie das TestPoint Graph Objekt verwendet wird.
- Die „Inspect“-Funktion.
- Wie man x-y Paare verwendet, um Cursordarstellungen zu erreichen.
- Weitere Grafik-Funktionen.

Graphen zeichnen

TestPoint beinhaltet ein einfach zu verwendendes Graph Objekt mit vielen Funktionen:



Die meisten Grafiken werden mit einer einzigen Action Zeile gezeichnet:

1) Draw Graph Graph1 with A/D1

Dabei sind die Parameter (die Daten, die gezeichnet werden) vom A/D Objekt, vom File Objekt mit Daten von der Festplatte, vom GPIB Objekt, vom Math Objekt, usw.

Die „Draw Graph“ Aktion kann eine beliebige Anzahl von Parametern verwenden, um Daten in dem gleichen Graphen zu zeichnen.

Datentypen

Das Graph-Objekt zeichnet jeden Vektor als eigene Linie in der Grafik, mit Optionen, die die Farbe, Symbole usw. betreffen.

Eine Grafik zeichnet jede Position in einer Liste als eigene Linie. Es können 32 Linien in einer Grafik dargestellt werden, obwohl nur für 8 Linien Einstellungen vorhanden sind. Werden mehrere Parameter angegeben, werden diese wie eine Liste behandelt, die in jeweils verschiedenen Linientypen dargestellt werden.

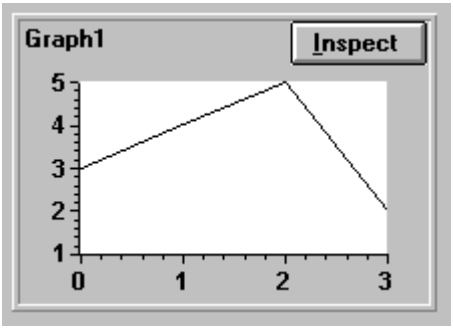
Die zweckmäßigste Datenform für eine Grafik ist somit ein Vektor oder eine Liste aus Vektoren. Außerdem ist dies der Datentyp, den Sie von den meisten Erfassungsoperationen erhalten, wie beispielsweise von der „Acquire A/D“ Aktion des A/D Objektes. In diesem Fall sehen die Action Listen aus, wie die hier gezeigte:

1)	Acquire A/D	A/D1	#samples=100, rate=1000 Hz, channel(s)=0,1,2
2)	Draw Graph	Graph1	with A/D1

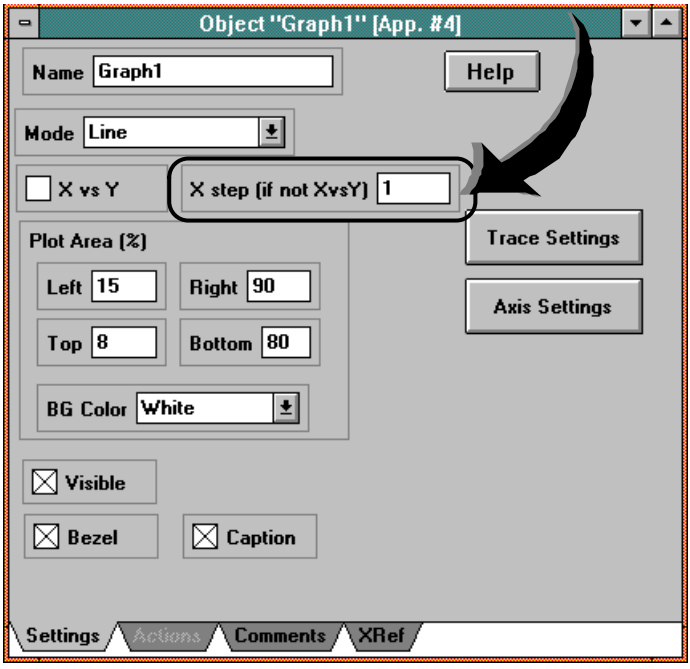
Y- und XY Darstellungen

Die Voreinstellung für den Grafik Modus ist der „Y-Plot“, der alle Daten, die übergeben werden, als y Werte ansieht und diese gegen eine Reihe von gleichförmig verteilten x Werten darstellt.

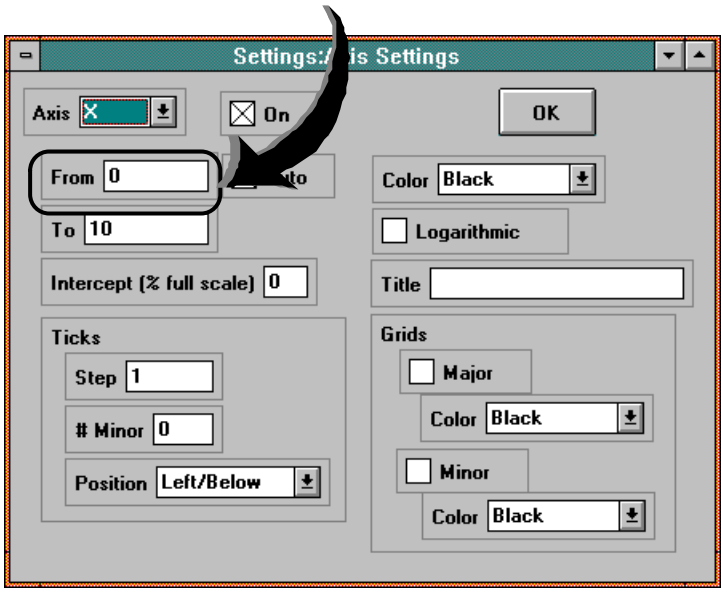
Dieser Vektor: **vector(2,4,5,2)** wird so gezeichnet:



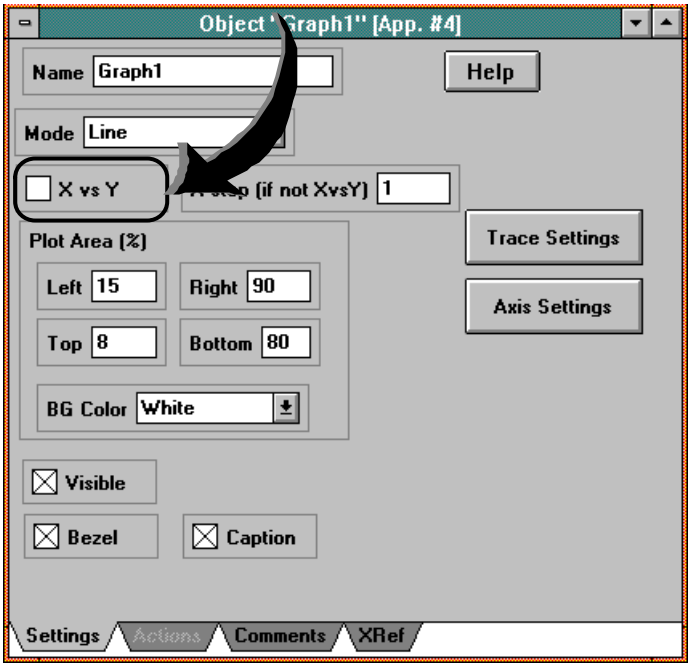
In diesem Modus fangen die x Werte normalerweise bei 0 an und werden jeweils um 1 erhöht. Die Schrittweite der Erhöhung kann mit der Einstellung „X step (when not X vs. Y)“ im Settings Fenster des Graph Objektes verändert werden:



Der Startwert kann durch die Einstellung des „From“ Wertes in den Einstellungen der x-Achse (Axis Settings) eingestellt werden:

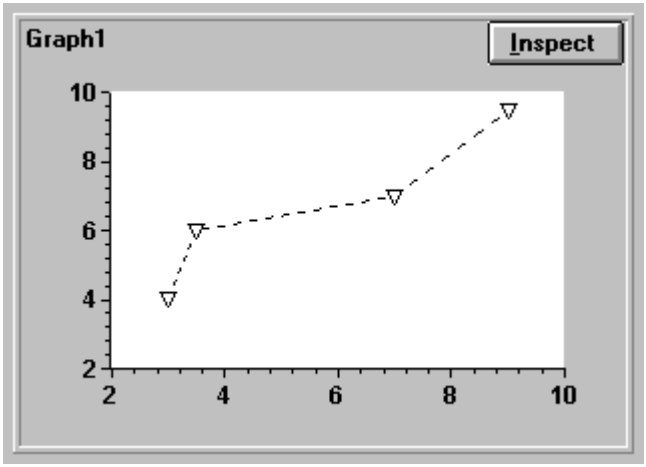


Sie können die x-Werte außerdem explizit vorgeben. Übergeben Sie dazu einen Vektor mit den vorgesehenen Werten als ersten Parameter der „Draw Graph“ Aktion. Die Einstellung „**X vs. Y**“ des Graph Objektes muß dazu eingestellt sein:



In diesem Modus können Sie zwei Vektoren mit der „Draw Graph“ Aktion zeichnen. Dabei wird der erste als x-Datensatz, der zweite als y-Datensatz verwendet. Die Datenpunkte müssen dabei nicht den gleichen Abstand haben.

Die Vektoren: **vector(3,3.5,7,9)** und **vector(4,6,7,9.5),** ergeben:

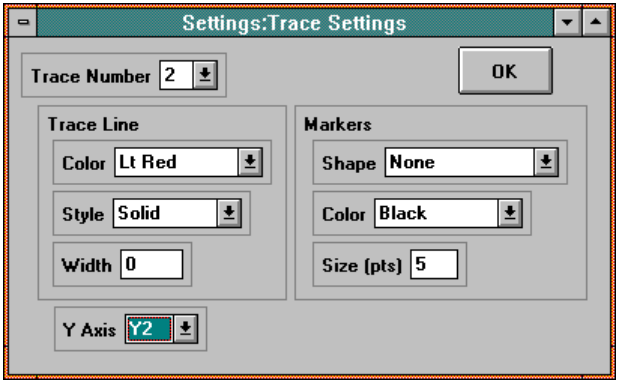


Achsen

Wenn die Voreinstellungen benutzt werden, werden x- und y-Achse automatisch skaliert. Sie können aber sowohl feste Werte als Start- und Stopwert für die Achsen vorgeben, als auch die Schrittweite zwischen den Zahlen an den Achsen und die Anzahl der Unterteilungen zwischen den Zahlen angeben.

Weiterhin unterstützt das Graph Objekt bis zu 4 y-Achsen. Standardmäßig ist nur eine eingeschaltet. Weitere können zugeschaltet werden, indem in der „Axis Settings“ Dialogfenster die Option für die entsprechende Achse gesetzt wird.

Linien werden den entsprechenden Achsen zugewiesen, indem Sie die „Y Axis“ Einstellung im „Trace Settings“ Dialogfenster verwenden. Zum Beispiel wird hier die Linie 2 der Y2 Achse zugewiesen:



Balken Darstellung

Das Graph Objekt stellt weiterhin die Möglichkeit zur Verfügung, eine Linie als Balkengrafik darzustellen. Dabei können auch weitere Linien miteingezeichet werden. Verwenden Sie die Einstellung „Bar (1st trace)“ in den „Mode“ Einstellungen des Graph Objektes.

XY Paare Darstellung

Zusätzlich zu der Möglichkeit, mehrere Y Reihen von Werten gegen eine Reihe von X Werten darzustellen, stellt TestPoint eine Variante zur Verfügung, in der Sie unabhängige Paare von x-y-Werten darstellen können. Wählen Sie dazu den „XY pairs“ Modus in den Graph Einstellungen aus.

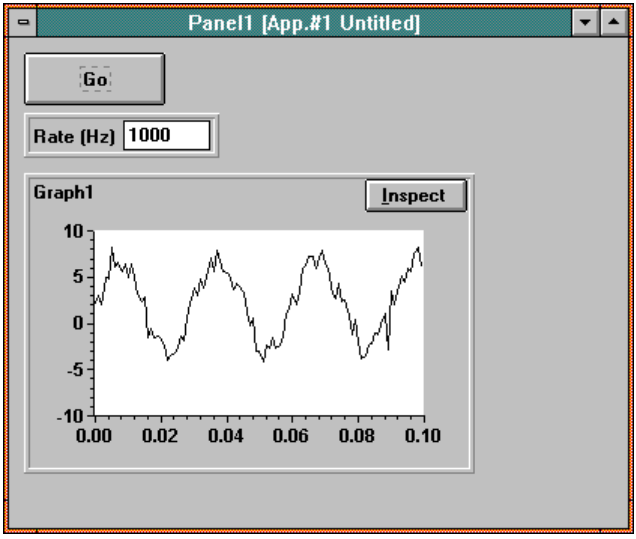
Im „XY pairs“ Modus sollten Sie eine gerade Anzahl von Vektoren in der „Draw graph“ Aktion angeben. Jedes Vektor-Paar wird als Serie von x-y Paaren dargestellt.

Verändern von Grafiken

Für das Graph Objekt gibt es viele Einstellungsmöglichkeiten. In einigen Anwendungen kann es hilfreich sein, diese während der Ausführung zu modifizieren.

Skalierung der x-Achse

Zum Beispiel, wenn eine Kurve, die Sie zeichnen wollen, an der x-Achse anstelle der Zählwerte (0,1,2,3,...) die Zeit in Sekunden dargestellt werden sollen.



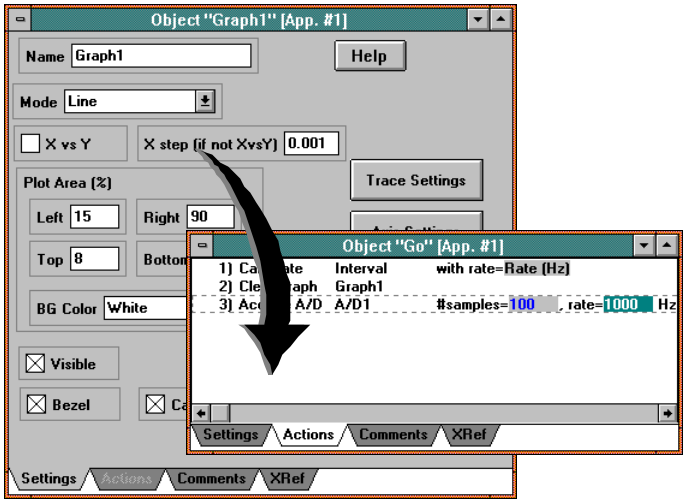
Der darzustellende x-Wert (Schrittweite) ist das Intervall zwischen zwei Punkten der Kurve. Berechnen Sie das Intervall in der Action Liste:

1) Calculate Interval with rate=Rate (Hz)

Ziehen Sie dann das Graph Objekt in die Action Liste und wählen Sie dann die „Clear Graph“ Aktion. Die Grafik wird gelöscht, weil eine neue Einstellung einen neuen Aufbau erforderlich macht. Löschen Sie diese Grafik nicht vorher, wird diese zweimal neu aufgebaut, einmal, wenn Sie die Einstellung ändern und zum zweiten Mal, wenn die „Draw graph“ Aktion später ausgeführt wird.

2) Clear graph Graph1

Öffnen Sie jetzt wieder das Settings Fenster des Graph Objektes und ziehen Sie die Einstellung „X step (if not X vs. X)“ in die Action Liste:



4) Set Graph1(X step if not X vs. Y) to Interval

Als letzten Schritt führen Sie den Neuaufbau der Grafik aus:

5) Draw graph Graph1 with A/D1

Die gesamte Action Liste einschließlich der Datenerfassung sieht so aus:

1)	Calculate	Interval	with rate=Rate (Hz)
2)	Clear graph	Graph1	
3)	Acquire A/D	A/D1	#samples=100, rate=Rate (Hz), channel(s)=0
4)	Set	Graph1(X step if not X vs. Y)	to Interval
5)	Draw graph	Graph1	with A/D1

Linien- und Achseneinstellungen ändern

Sie können die Einstellungen für eine Achse oder eine Linie ändern. Zum Beispiel können die Achsentitel auch vom Anwender eingegeben oder aus einer Datei gelesen werden.

Als erstes wählen Sie die gewünschte Achse oder Linie mit Hilfe der „Axis“ oder „Trace“ Einstellung aus. Erzeugen Sie zwei Zeilen in der Action Liste. Die erste wählt die Achse aus, die zweite ändert die gewünschte Einstellung (Farbe, Titel, Dicke der Linie,...).

Wenn Sie die Titel der Achsen verändern wollen, wechseln Sie zum „Axis Settings“ Fenster des Graph Objektes und erzeugen Sie durch Ziehen der Einstellung folgende Action Liste:

1)	Set	Graph1(Axis)	to "X"
2)	Set	Graph1(Title)	to "X axis title"
3)	Set	Graph1(Axis)	to "Y"
4)	Set	Graph1(Title)	to "Y axis title"

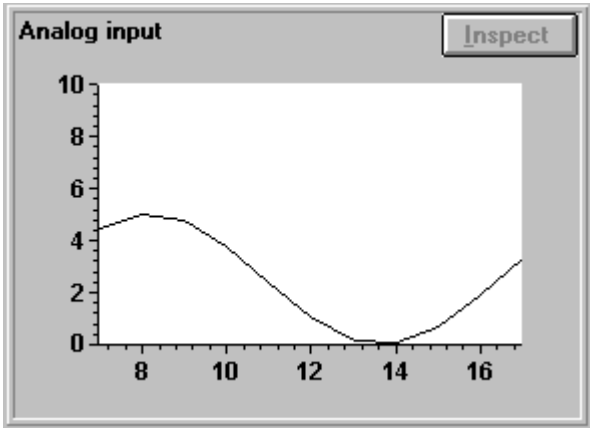
Beachten Sie, daß Zeile 2 und 4 den gleichen Namen haben, aber auf verschiedene Achsen zugreifen, die in den jeweilig vorhergehenden Zeilen ausgewählt werden.

Linienschreiber

Zusätzlich zum Zeichnen statischer Linien und Balkengrafiken, kann das Graph Objekt Linienschreibergrafiken aktualisieren. Um eine Linienschreibergrafik zu erzeugen, stellen Sie den Grafik Modus auf „Strip-Chart“ ein. In der Action Liste verwenden Sie dann die „Add point(s)“ Aktion, die einen oder mehrere Werte an die Grafik anfügt, ohne die vorhandene Linie zu löschen:

1)	Add point(s) to	Graph1	from A/D1, max # points=4000

Bei Erreichen des rechten Randes der Grafik, beginnt das Graph Objekt damit, den Graphen nach links durchzurollen (scrollen), um an der rechten Seite neue Daten einzeichnen zu können.



Bereich des Graphen

Die Breite der x-Achse eines Linienschreibers wird von der „From“ und „To“ Einstellung der x-Achse gesteuert („Auto“ wird ignoriert). Die „From“ Einstellung gibt den Startpunkt vor. Das Intervall auf der x-Achse ändert sich nicht. Die Beschriftung wird laufend angepaßt.

Vorgeschichte

Die Daten, die an der linken Seite herausgeschoben werden, können durch Verwendung des INSPECT Schalters der Grafik, in einem gesonderten Fenster, wieder angesehen werden. Die Tiefe des Speichers dafür wird von dem „max. # points“ Parameter der „Add point(s)“ Aktion gesteuert.

Geschwindigkeit

Linienschreiber brauchen eine Menge Rechenzeit, um die Grafik beim durchrollen zu aktualisieren, weil sowohl die gesamte Grafik, als auch die x-Achse aktualisiert werden müssen. Die Geschwindigkeit der Aktualisierung hängt stark von der Prozessor- und Grafikkartenleistung ab. Normalerweise wird jedoch die Aktualisierungsrate des Linienschreibers die Einschränkung darstellen, verglichen mit der Datenerfassung oder gar dem Speichern auf Festplatte.

Ein Weg, den Durchsatz an Datenpunkten zu erhöhen, ist, immer mehrere Punkte gleichzeitig zu aktualisieren. Die „Add point(s)“ Aktion akzeptiert sowohl eine einzelne Zahl, als auch einen Vektor. Dadurch wird die Grafik seltener aktualisiert und mehr Daten können in vorgegebener Zeit verarbeitet werden.

Bei der Erfassung von A/D Karten, können Sie einfach den „samples/event“ Parameter der „Start A/D“ Aktion so einstellen, daß immer n Punkte verarbeitet werden:

1)	Start A/D	A/D1	#points="continuous", rate=50 Hz, channel(s)=0, samples/event=25

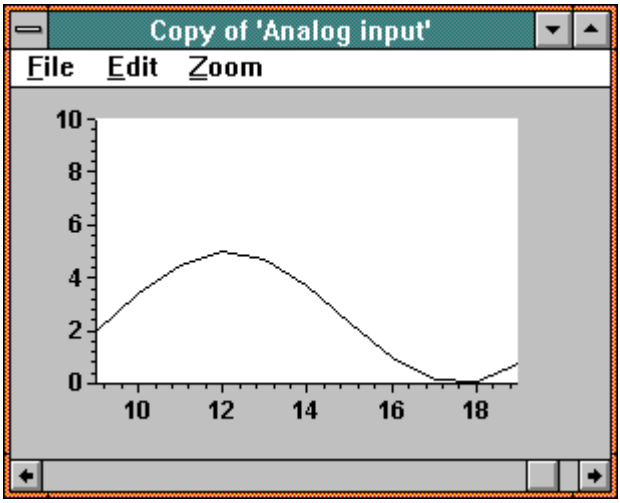
Ein guter Wert für die Aktualisierung liegt im Bereich einiger Samples pro Sekunde, beispielsweise 2 bis 4. Sie können dies häufiger oder seltener machen. Wägen Sie dabei ein ruhiges Durchrollen gegen die verbrauchte Prozessorleistung ab.

Ein weiterer Weg, die Darstellung zu verbessern, ist, weniger Punkte zu zeichnen, als Sie erfaßt haben. Dies bietet sich speziell an, falls Sie hohe Datenraten benötigen und der Computer diese Berechnungen gut durchführen kann. Der Beobachter kann normalerweise nur geringe Datenmengen als Trendgrafik betrachten, um die Entscheidung zu treffen, ob die Daten in Ordnung sind oder nicht. Es gibt verschiedene Möglichkeiten, um eine Datenreduktion zu veranlassen. Eine davon ist die **decimate()** Funktion des Math Objektes, die jeden n-ten Punkt eines Datensatzes ermittelt.

1)	Calculate	Down-sampled	with A/D1, N=10
2)	Add point(s) to	Graph1	from Down-sampled, max. # points=1000

Das „Inspect“ Fenster

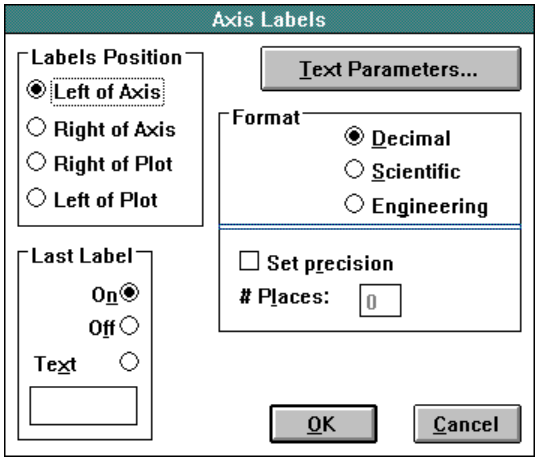
Jedes Graph Objekt besitzt einen „Inspect“ Schalter, der aktiv ist, wenn der Run Modus eingeschaltet ist und Daten verfügbar sind. Wenn Sie auf diesen Schalter klicken, erhalten Sie eine Kopie der Grafik in einem separaten Fenster:



Dieses Fenster kann bewegt und in der Größe geändert werden, ohne, daß es in der Originalgrafik Auswirkungen hat. Die Originalgrafik wird, während der Anwender mit der Kopie arbeitet, weiterhin aktualisiert.

Im „Inspect“ Fenster kann die Grafik gezoomt, gedruckt und in die Zwischenablage kopiert werden. Außerdem können viele Einstellungen verändert werden.

Ein Doppelklick auf eine Achse oder die Achsen-Beschriftung erlaubt die Veränderung dieser Parameter. Bei der y-Achse beispielsweise erscheint dann diese Dialogbox:



die Sie die Beschriftung in verschiedenen Arten modifizieren läßt.

Jede Bearbeitung der Grafik im „Inspekt“ Fenster ist temporär und wird nur auf die Kopie der Grafik angewendet.

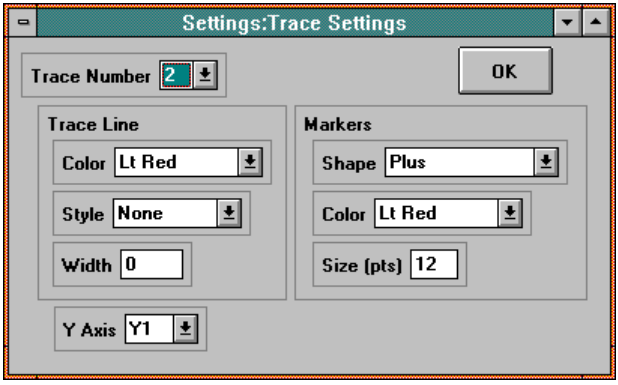
Cursor

Sie können den Anwender die Daten in der Grafik ebenfalls programmiert und interaktiv bearbeiten oder hervorheben lassen.

Im „XY-pairs“ Modus des Graph Objektes können Sie einen oder mehrere Cursor in die Gafik bringen. Erzeugen Sie einfach zwei neue Vektoren, die die entsprechenden x- und y-Koordinaten für die Cursor enthalten und zeichnen Sie diese mit in die Grafik ein. Diese Methode erfordert den „XY pairs“ Modus, da Sie einerseits die Daten messen, andererseits die passenden x-Werte mit dem Math Objekt erzeugen.

Die Schritte um einen Cursor zu erzeugen:

- Entscheiden Sie, wie viele Linien eingezeichnet werden (z. B. 1). Wechseln Sie dann in die Einstellungen für die Linien (Trace) und stellen für die nächste Linie (z. B. Linie 2) den „Trace line style“ auf „none“ und die Option „Markers shape“ auf „plus“. Möglicherweise möchten Sie die Größe der Marker auch einstellen:



- Stellen Sie im Settings Fenster des Graph Objektes den „XY pairs“ Modus ein.

- Erzeugen Sie nun Ihre Action List, wobei Sie die x- und y-Daten in die Grafik bringen. In diesem Beispiel wird ein A/D Kanal erfaßt, dann die x-Daten mit dem Math Objekt erzeugt:

1)	Acquire A/D	A/D1	#points=10, rate=1000 Hz, channel(s)=0
2)	Calculate	Data	with wave=A/D1, rate=1000

wobei dies die Formel für das Math Objekt „Data“ ist:

list(ramp(dim(wave))/rate,wave)

- Jetzt werden die Cursor Daten erzeugt, indem ein Math Objekt mit einer Formel wie der folgenden verwendet wird, die eine Liste aus zwei Vektoren erzeugt. Die Vektoren haben in diesem Fall nur ein Element, da es nur der Cursor ist, der dargestellt werden soll. Es können jedoch auch mehrere Werte darin enthalten sein.

list(vector(x), vector(y))

3)	Calculate	Cursors	with x=X-slider, y=Y-slider

- Zeichnen Sie nun die Grafik mit dem Objekt Data und den Cursor Daten:

4)	Draw graph	Graph	with Data, Cursors

Komplette Beispiele finden Sie in den Dateien CURSOR1.TST und CURSOR2.TST in Ihrem \EXAMPLES-Verzeichnis.

Es sind keine Cursor vorgefertigt, weil es sehr viele Optionen und Möglichkeiten der Bearbeitung gibt. Durch Cursor als aktive Elemente in Ihrer Anwendung können Sie nahezu alle Kombinationen dieser Funktionen, die Sie benötigen, erzeugen, wie z. B. die Positionierung der Cursor zu gemessenen Positionen in der Kurve oder Kurvenglättung zwischen den Punkten, usw.

Verwendung externer Grafik Software

Was tun, wenn Sie Grafikfunktionen benötigen, die über die Funktionen von TestPoint´s Grafik Objekt hinaus gehen?

Sie können externe Grafik Software verwenden. Das kann sein:

- Ein eigenständiges Programm, wie eine Tabellenkalkulation oder ein Grafik Programm:

  Beispiele sind: Microsoft Excel, Lotus 1-2-3, Harvard ChartXL und andere.

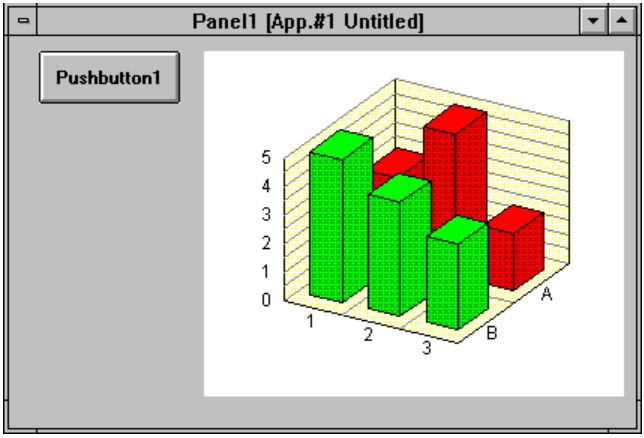
- Ein Visual Basic Custom Control (als eine VBX Datei).

  Ein Beispiel ist: Chart FX

OLE

Ein eigenständiges Programm kann Grafiken direkt im TestPoint Panel erzeugen, indem die OLE Funktionalität beider Programme verwendet wird. OLE wird von TestPoint mit dem OLE Objekt unterstützt. Eine vollständige Bescheibung finden Sie im Kapitel 19.

Hier sehen Sie eine Quattro Pro 6.0 3-D Balkengrafik, direkt im TestPoint Panel dargestellt:

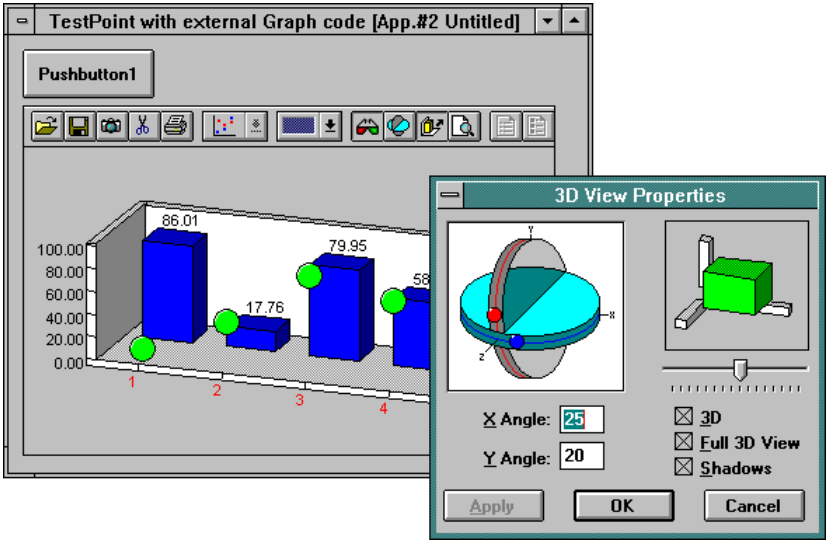


OLE benötigt die entsprechende andere OLE-Software auf dem Computer installiert (oben muß beispielsweise Quattro Pro installiert sein).

VBX

Andere Typen von Grafiken werden von Softwarefirmen in Form von VBX-Tools angeboten. VBX-Tools sind Komponenten, die direkt in TestPoint oder andere Entwicklungsumgebungen, wie Visual Basic und Visual C++, eingebunden werden können.

Ein Beispiel dafür ist Chart FX von Software FX Inc. in Boca Raton, Florida. Dieses Paket unterstützt viele Arten von Graphen, zuzüglich Toolbars und anderer Funktionen.



VBX Custom Controls werden normalerweise mit einer kostenlosen Runtime Lizenz ausgeliefert, so daß Sie diese mit Ihrer TestPoint Anwendung ausliefern können.

Das VBX Objekt wird in Kapitel 14 beschrieben.

Kapitel 13. Berichterzeugung

Was in diesem Kapitel behandelt wird:

- Wie Sie das TestPoint Report Objekt dazu verwenden können, gedruckte Berichte von Messungen und Test Ergebnissen zu erhalten.

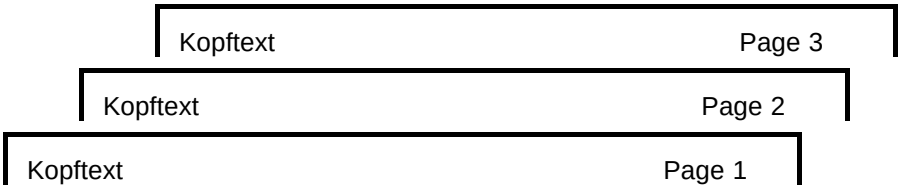
Einfache Berichte

Das Report Objekt  kann verwendet werden, um Texte und Grafiken formatiert auf einem beliebigen Windows Drucker auszugeben. Ein Report wird durch die Verwendung der „**Start**“-, „**Print**“- und „**End**“-Aktion erzeugt.

1)	Start	Report1	
2)	Print	Report1	"Voltage= ", GPIB-Meter
3)	Print	Report1	Graph1
4)	End	Report1	

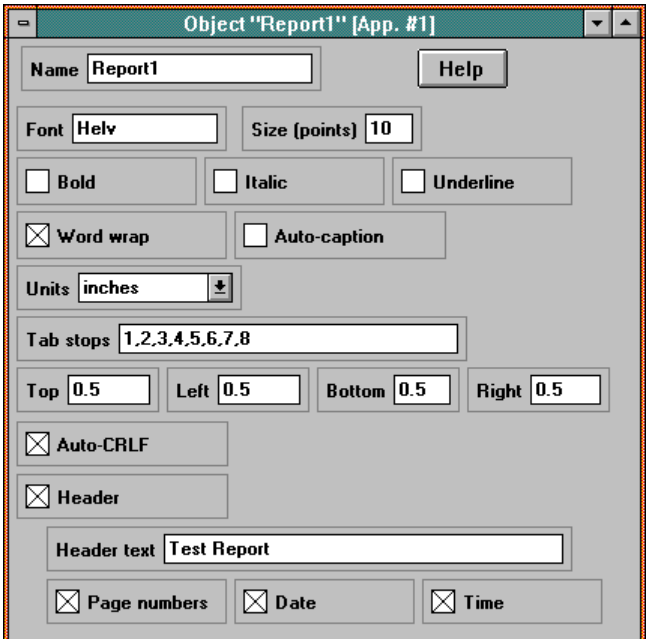
Das Drucken unter Windows wird vom Druckmanager organisiert, der sogenannte „Aufträge“ von Programmen übernimmt und nacheinander ausdruckt. Die „End“ Aktion beendet den „Druck Auftrag“ und veranlaßt Windows dazu, den Bericht zum Drucker zu schicken.

Berichte können aus verschiedenen Seiten bestehen, die Kopftexte und Seitennummern haben können.



Einstellungen

Die Einstellungen des Report Objektes beziehen sich auf Schriftart, Kopfzeilen, Seitenränder, Tabulatoren und Formatierungsoptionen, die später in diesem Kapitel beschrieben werden.



Die Voreinstellungen sind optimiert für gängige Berichte. Trotzdem besteht oft der Wunsch, z. B. Kopftext und Schriftart zu verändern.

Die Print Aktion

Die „Print“ Aktion gibt ein oder mehrere Objekte aus, die entweder eine Zeile oder sogar eine ganze Seite einnehmen können. Wenn das Ende einer Zeile erreicht ist, und der Zeilenumbruch (Word-wrap) eingeschaltet ist (Voreinstellung), wird in der nächsten Zeile weitergedruckt. Nachdem die „Print“ Aktion ausgeführt wurde, wird die Druckposition normalerweise zur nächsten Zeile gesetzt.

Konstanten drucken

Konstante Texte oder Zahlen werden direkt gedruckt. Beispiel:

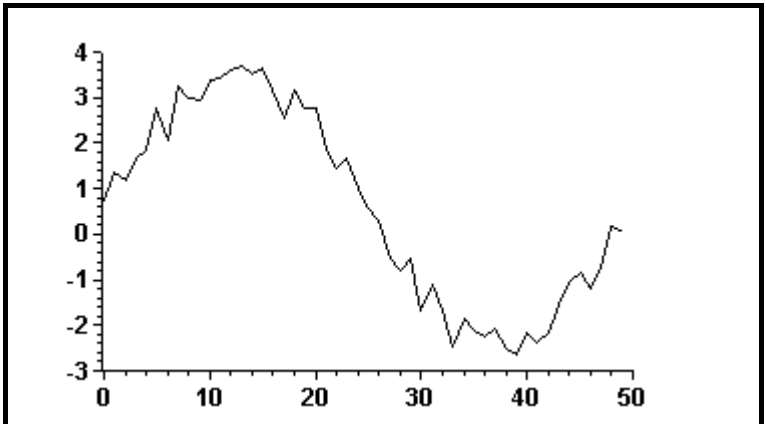
```
1)  Print                Report1          "Das ist Text und x=", 4.567
```

Hat beim Ausdruck zu Ergebnis:

```
Das ist Text und x=4.567
```

Graph-, Panel- oder Picture Objekte drucken

Grafiken, Panels und Bilder werden graphisch gedruckt. Sie werden so skaliert, daß sie auf die Seite passen.



Zur Steuerung der Skalierung dieser Grafiken können Sie die „Print (at)“ Aktion benutzen, die später in diesem Kapitel beschrieben wird (unter „Positionierung von Texten und Grafiken“).

Drucken eines Grid Objektes

Ein Grid Objekt wird die Spaltentitel vor den Daten drucken, falls diese vorhanden sind, und die I/O Formatierung nicht übergangen wird:

```
Spannung        Strom
3.48            5.38
5.82            8.28
```

Anschließend werden die Daten in Zeilen und Spalten, unter Verwendung des TAB Zeichens zwischen den Spalten, gedruckt. Die Tabulator Position, und hieraus die Spaltenbreite, richten sich nach den Einstellungen im Report Objekt. Ist eine Zeile durch zu viele Spalten zu lang, wird in die nächste Zeile auf die erste Tabulatorposition gewechselt. Dabei wird eine Fortsetzungslinine eingefügt.

Drucken anderer Objekte

Alle anderen Objekte drucken ihre Datenwerte als Text formatiert, genauso, wie sie in einer Datei gespeichert würden. Damit könnte diese Action Zeile:

```
2) Print                A/D1
```

eine Ausgabe wie diese erzeugen, wenn der Wert des Objektes eine Zahl ist:

```
3.1415
```

oder diese bei einem Vektor:

```
1.23
2.34
3.45
4.56
```

Der Abschnitt Formatierung, später in diesem Kapitel, gibt Hinweise, wie diese Ausgabeform verändert werden kann.

Beachten Sie, daß die Datenwerte des Objektes gedruckt werden, nicht die Überschrift (Caption) oder Text, der möglicherweise auf dem Panel angezeigt wird. Für Objekte, wie das Display Objekt, ist der Datenwert selbstverständlich der Wert, der angezeigt wird. Beim Text Objekt hat die Anzeige nichts mit dem Datenwert zu tun.

Beachten Sie weiterhin, daß Sie jedes Objekt drucken können, auch solche, die nicht auf dem Panel dargestellt werden können, wie z. B. das Math Objekt.

Seiten

Ein Bericht kann aus mehreren Seiten bestehen. Eine neue Seite wird immer dann angefangen, wenn der untere Rand einer Seite erreicht wurde und mehr Daten gedruckt werden sollen.

Sie können eine neue Seite auch selber an der gewünschten Stelle, mit der „**New Page**“ Aktion, einfügen:

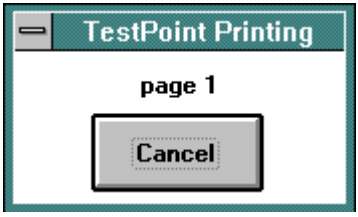
3) New Page Report1

Die Seiten werden, beginnend bei 1, laufend durchnumeriert. Das Drucken der Seitennummer im Kopf und das Drucken des gesamten Kopfes kann im Settings Fenster des Report Objektes ein- oder ausgeschaltet werden.

Beenden oder Abbrechen

Ein Bericht ist nicht beendet und zum Druckmanager übergeben, wenn die „**End**“ Aktion nicht ausgeführt wurde. **Denken Sie daran, die „End“ Aktion auszuführen!**

Während der Bericht erzeugt wird, wird eine Dialogbox auf dem Bildschirm angezeigt, die den Anwender den Bericht abbrechen läßt, wenn der es wünscht:



Die Anwendung wird weiter ausgeführt, aber die „Print“ und „End“ Aktionen kaben keine Wirkung mehr, wenn der Druck des Berichtes manuell abgebrochen wird.

Haben Sie in Ihrer Anwendung Bedingungen, die den Druck programmgesteuert abbrechen sollen, verwenden Sie dazu die „**Cancel**“ Aktion:

5) Cancel Report1

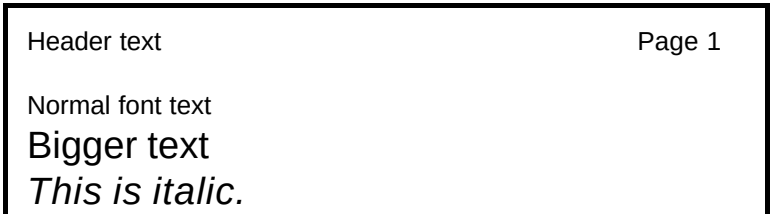
Formatierung

Schriftart

Die Schriftart, die für das Drucken von Berichten verwendet wird, kann über die Einstellungen **Font**, **Size**, **Bold**, **Italic** und **Unterline** im Settings Fenster des Report Objektes formatiert werden. Die voreingestellte Schriftart ist „Arial“, die eine Standardschrift von Windows ist und gut lesbar ist. Die Größe wird in Punkten angegeben, ein Standardmaß für Druckausgaben, ein Punkt entspricht 1/72 eines Zolls.

Sie können eine Schrift voreinstellen und nach Bedarf die Einstellungen des Objektes mitten im Bericht ändern.

Sie können die Einstellungen dazu in den Action Listen verändern, um z. B. die Schriftart zu modifizieren.



Dazu öffnen Sie das Settings Fenster des Report Objektes und ziehen die entsprechende Einstellungsoption in die Action Liste, von der aus die Änderung vorgenommen werden soll.

Hier ist die Action Liste, die das oben gezeigte Beispiel erzeugt:

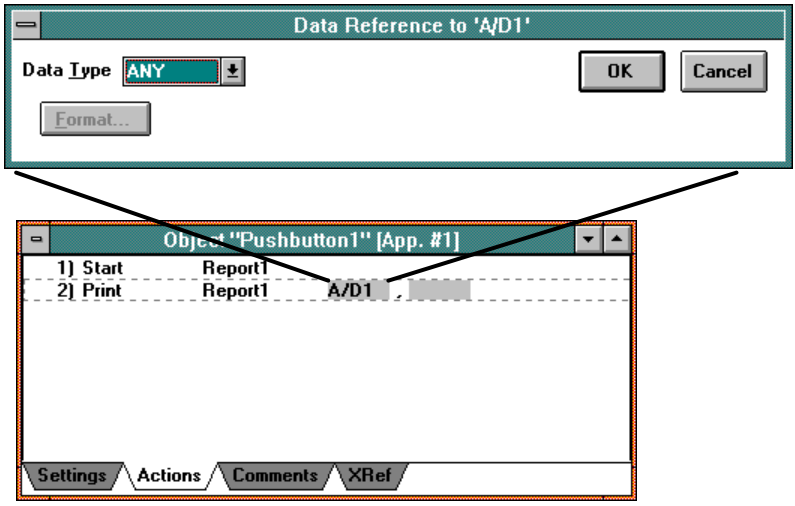
```
1) Start      Report1
2) Print      Report1                "Normal font text"
3) Set        Report1(Size (points))  to 14
4) Print      Report1                "Bigger text"
5) Set        Report1(Italic)        to 1
6) Print      Report1                "This is italic."
```

7) End Report1

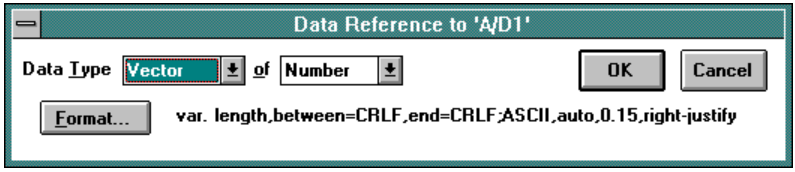
Formatierung von Datenwerten

Wenn Sie ein Objekt, wie A/D, GPIB, Math usw. drucken, wird der Datenwert als Text gedruckt. Die Ausgabeformatierung folgt den gleichen Regeln und Voreinstellungen wie jede andere Textausgabe in TestPoint. In Kapitel 5 wird die Ein- und Ausgabeformatierung ausführlich beschrieben.

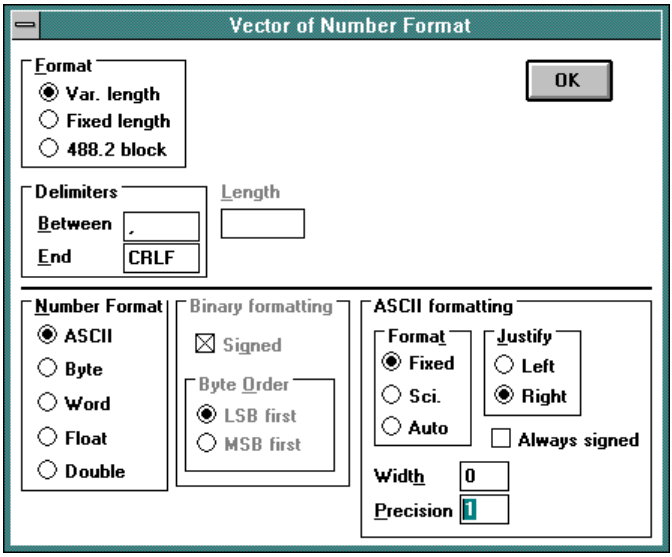
Um die Formatierung einzuschalten, doppelklicken Sie auf den Objekt Parameter der „Print“ Aktion. Sie erhalten folgende Dialogboxen:



Sie können dann den Datentyp wählen, der für Ihre Anwendung erforderlich ist:



Der Format Schalter öffnet ein Fenster, in dem Sie alle Formatierungseinstellungen, wie Anzahl der Nachkommastellen, Begrenzungszeichen zwischen den Elementen eines Vektors, usw. einstellen können.



So könnte beispielsweise der Vektor aussehen, wenn anstelle der CRLF Begrenzungszeichen zwischen den Elementen ein Komma eingesetzt wurde und nur eine Nachkommastelle angezeigt wird:

```
3.4,1.2,1.3,1.4,5.7
```

Positionierung von Texten und Grafiken

Wenn Sie die „Print“ Aktion verwenden, wird jedes Element an der aktuellen Position gedruckt. Diese Position beginnt am Anfang der Seite unter dem eingestellten Rand und der Kopfzeile und wird bei jeder Ausgabe aktualisiert.

Wenn Sie mehrere Elemente in einer einzelnen „Print“ Aktion drucken, sieht das Ergebnis so aus:

```
2) Print        Report1     "Text ", "Mehr Text, x=",
                            A/D1
```

```
Text Mehr Text, x=3.483
```

Die aktuelle Position wird nach jedem Element aktualisiert und in der gleichen Zeile nach rechts verschoben. Ist der rechte Rand erreicht, wird am Anfang der nächsten Zeile gedruckt oder der Druck angehalten, jenachdem, ob der Zeilenumbruch (Word-wrap) eingeschaltet ist oder nicht.

Nach der „Print“ Aktion wird die aktuelle Position auf den Anfang der nächsten Zeile eingestellt (wenn die Auto-CRLF Einstellung im Settings Fenster des Report Objektes eingeschaltet ist):

```
3) Print        Report1     "Text in einer Zeile"
4) Print        Report1     "Mehr Text und eine Zahl: ",
                            5
```

erzeugt den Ausdruck:

```
Text in einer Zeile
Mehr Text und eine Zahl: 5
```

Sie können die aktuelle Druckposition selber einstellen, indem Sie die „**Set position**“ Aktion verwenden, die x (über die gesamte Seitenbreite) und y (die Seite hinunter) als Argumente benötigt. Diese Position ist relativ zu den Seitenrändern. Die Einheit wird im Setting Fenster des Report Objektes unter der Option „**Units**“ eingestellt.

5)	Set position in	Report1	x=2, y=3.5

Um die Plazierung eines Elementes zu steuern, verwenden Sie die „**Print (at)**“ Aktion, die es erlaubt, einzelne Elemente an entsprechende x- und y-Positionen auf der Seite zu drucken.

6)	Print (at)	Report1	x=2, y=3.5, w=3, h=1.5, value=Graph1

Die ist sehr nützlich bei der Skalierung von Grafiken, Bildern und Panels auf eine bestimmte Größe.

Sie können auch x- und y-Werte der „Print (at)“ Aktion mit -1 besetzen. Dies bedeutet, daß die aktuelle Position verwendet wird. Dies läßt Sie etwas an die aktuelle Position drucken und trotzdem die Größe bestimmen:

7)	Print (at)	Report1	x=-1, y=-1, w=3, h=1.5, value=Graph1

Auto-CRLF

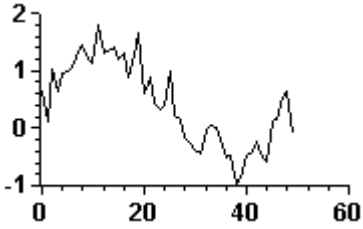
Mit der aktiven Einstellung Auto-CRLF (Voreinstellung) wird bei einer „Print“ Aktion immer eine neue Zeile zur aktuellen Positionierung verwendet. Dies hat zur Folge, daß fortlaufende Aktionen in fortlaufenden Zeilen dargestellt werden.

Durch das Abschalten dieser Einstellung können Sie mehrere „**Print**“ Aktionen in die gleiche Zeile drucken lassen. Das ist teilweise sehr nützlich, in Verbindung mit der „**Print (at)**“ Aktion an die aktuelle Stelle (x und/oder y auf -1) aber mit einer vordefinerten Größe:

1)	Start	Report1	
2)	Print (at)	Report1	x=-1, y=-1, w=2, h=3, value=Data-Entry1
3)	Print (at)	Report1	x=-1, y=-1, w=4, h=3, value=Graph1
4)	End	Report1	

Das oben dargestellte Beispiel druckt einigen Text, dann eine Grafik, die skaliert ist auf 4 cm mal 3 cm. Wenn die Auto-CRLF Option eingeschaltet ist, wird die Grafik in die nächste Zeile gedruckt. Ist die Option ausgeschaltet, wird der Text neben der Grafik gedruckt, wie hier:

This is the text that was in the Data-Entry1 object when line 2 of the above action list executed.



Auto-caption

Die „Auto-caption“ Option im Settings Fenster des Report Objektes verwendet den Namen eines Objektes und ein Gleichheitszeichen, das automatisch vor dem Datenwert gedruckt wird.

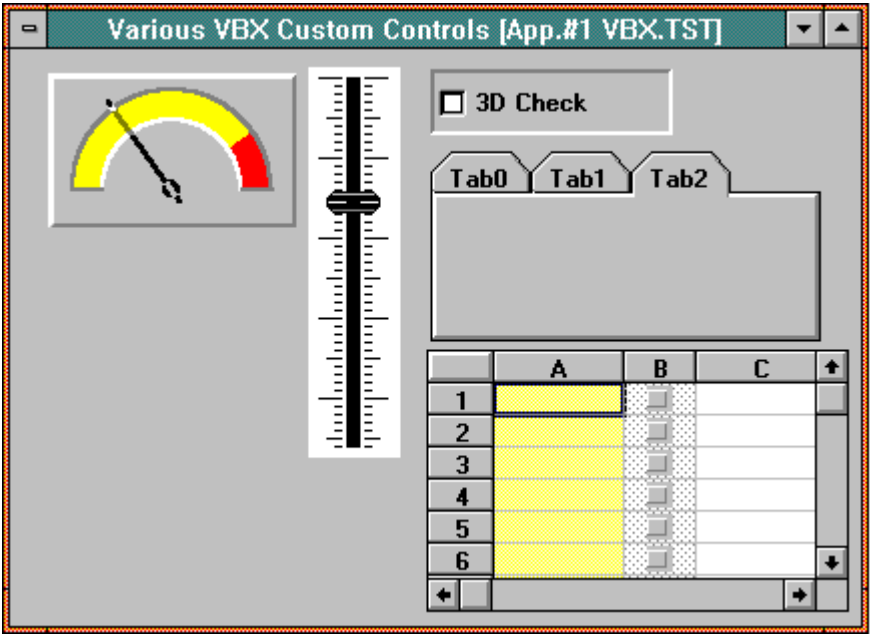
Kapitel 14. VBX´s: Custom Controls benutzen

Was in diesem Kapitel behandelt wird:

- Was sind VBX Custom Controls und Software Komponenten?
- Wie man VBX Custom Controls in TestPoint verwendet.

Was sind VBX Custom Controls?

VBX Controls sind eine Form von Industriestandard-„Software Komponenten“, die Sie kaufen und in TestPoint einbinden können.



Hunderte dieser VBX Controls sind von Dutzenden von Firmen verfügbar und unterstützen Funktionen wie Knöpfe, analoge Anzeigen, Schieberegler, Karteireiter, Tabellenkalkulationen, Grafiken, Datenbank-Werkzeuge, usw. Das Beispiel VBX.TST, das mit TestPoint ausgeliefert wird, zeigt nur 5 dieser Controls vom Markt.

Der VBX Standard wurde ursprünglich als Methode definiert, um Funktionen in Microsoft´s Visual Basic Sprache zu integrieren. VBX steht für „Visual Basic eXtension“. VBX Dateien werden heute nicht nur von Visual Basic unterstützt, sondern auch von Borland und Microsoft C++, und auch TestPoint.

Da die VBX Technologie ein offener Standard ist, gibt es ein komplettes und gut dokumentiertes Entwicklungs-Kit, und es ist eine sehr aktiver Markt um diese VBX Controls enstanden. Es gibt sogar zwei Magazine, die sich ausschließlich mit Visual Basic und diesen Add-Ons beschäftigen: *VB Tech Journal* und *VBX/OCX Developer*.

Beachten Sie, daß es verschiedene Ebenen der VBX Schnittstellen Definition gibt. TestPoint, C++ und einige andere Tools verwenden VBX Ebene 1 (Level 1). Die Mehrheit von VBX Controls arbeiten auf diesem Level, aber einige verwenden die speziellen Funktionen von VB 2.0 oder 3.0. Wenn die Controls mit Borland oder Microsoft C++ arbeiten, sind sie auch kompatibel zu TestPoint.

Ein populäres Paket im Bereich der VBX Controls sind VBTools von MicroHelp. Das Paket beinhaltet über 50 Controls zum Preis von weniger als 500 DM. In Deutschland wird der Vertrieb von MicroBasic organisiert, Telefon: 089/90499049.

Wie VBX Controls in TestPoint verwendet werden

Die VBX Controls werden in Form von VBX Programm-Dateien mit Dokumentation der Funktionen, Eigenschaften (Properties) und Ereignisse angeboten. VBX Eigenschaften korrespondieren zu TestPoint Einstellungen.

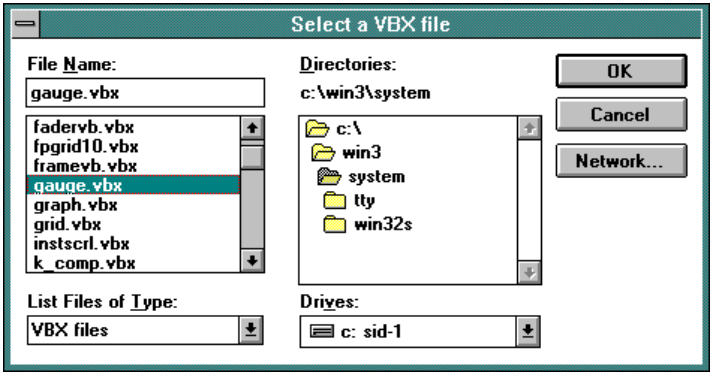
Hinweis: Bei den meisten VBX Controls gibt es kostenlose Lizenzen für Runtimes, wie bei TestPoint. Die VBX Dateien, die mit TestPoint geliefert werden, sind dieser Art. Sie können nicht in den TestPoint Editor geladen werden, ohne daß eine Fehlermeldung erzeugt wird. Wenn Sie diese Controls vom Anbieter installieren, wird die Lizenz zum Erzeugen neuer Anwendungen automatisch mit eingerichtet.

Laden eines VBX Controls

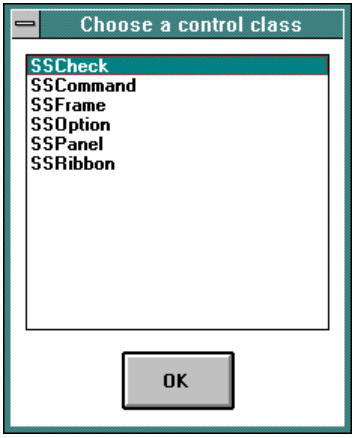
Wenn Sie ein VBX Control in Ihrer Anwendung benutzen wollen, ziehen Sie das VBX Objekt 

in Ihr Panel.

Klicken Sie den „Load VBX“ Schalter im Setting Fenster des VBX Objektes an und wählen Sie die gewünschte VBX Datei aus. Die meisten VBX Dateien werden im \WINDOWS\SYSTEM Verzeichnis installiert.

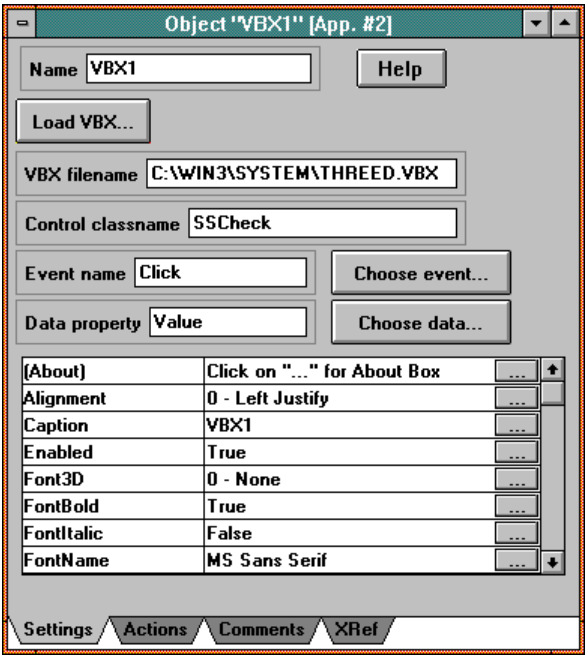

Select a VBX file
File Name:
gauge.vbx
fadervb.vbx
fpgrid10.vbx
framevb.vbx
gauge.vbx
graph.vbx
grid.vbx
instscrl.vbx
k_comp.vbx
List Files of Type:
VBX files
Directories:
c:\win3\system
c:\
win3
system
tty
win32s
Drives:
c: sid-1
OK
Cancel
Network...

Einige VBX Dateien beinhalten mehrere Arten von Controls. In diesem Fall erscheint ein Menü am Bildschirm, aus dem Sie den Typ, den Sie benutzen wollen, auswählen können:



Die VBX Datei ist geladen, und das Control ist nun ein TestPoint Objekt, auf das in Action Listen zugegriffen werden kann und das Datenwerte hat. Was auf dem Panel erscheint und welcher Code ausgeführt wird, wenn Sie Einstellungen ändern, ist in erster Linie von der VBX Datei abhängig. Viele verschiedene Möglichkeiten können somit in TestPoint verwendet werden.

Die VBX Einstellungen erscheinen, abhängig davon, welches Control Sie gewählt haben, in einer Liste mit Rollbalken im unteren Bereich des Fensters:



Der VBX Dateiname und der Control Klassenname werden automatisch eingetragen, wenn Sie das Control laden. Diese Einstellungen sollten nicht verändert werden.Der Ereignisname und die Daten-Eigenschaft werden später in diesem Kapitel behandelt - sie erlauben es, die Ereignisse und Daten des Controls dem TestPoint Objekt anzupassen.

Das Objekt Symbol für das VBX Objekt ändert sich ebenfalls, vom allgemeinen Symbol mit dem Namen „VBX“ zu einem Symbol, das in dem Custom Control Code enthalten ist, wie z. B.  .

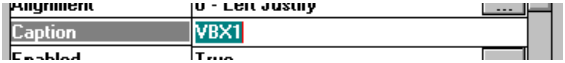
Eigenschaften sind noch mehr Einstellungen

Die Liste mit dem Rollbalken im Settings Fenster des VBX Objektes zeigt die **Eigenschaften** (Properties) dieses Controls. Verschiedene Controls haben verschiedene Eigenschaften, wie es in der Dokumentation der Controls beschrieben ist.

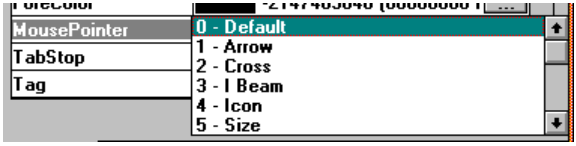
Einige Eigenschaften sind allgemeiner Art, wie z. B. die Hintergrundfarbe (BackColor), Vordergrundfarbe (ForeColor) und die Schriftart (Font), ... Diese Eigenschaften steuern den Text, der als Überschrift (Caption) oder Text Eigenschaft dargestellt wird.

Eigenschaften können numerischer Art (Ganzzahl oder Fließkommazahl), Zeichenketten, Farben, bool´sche Werte (Wahr/Falsch) oder Auswahlmenüs sein. Einige Eigenschaften bestehen aus Feldern (in der TestPoint Terminologie sind das Vektoren). Sie können diese Eigenschaften ändern, indem Sie auf den ... Schalter an der rechten Seite der entsprechende Eigenschaft klicken.

Für Zahlen oder Zeichenketten erhalten Sie eine Bearbeitungszeile, wie diese:



Für Farben bekommen Sie die Standard Windows Farbauswahlbox und für bool´sche Werte oder Auswahlmenüs eine „Dropdown“-Liste.



Es können ebenfalls Eigenschaften wie „runtime only“ oder „array - runtime only“ erscheinen. Diese Eigenschaften können nicht von hier

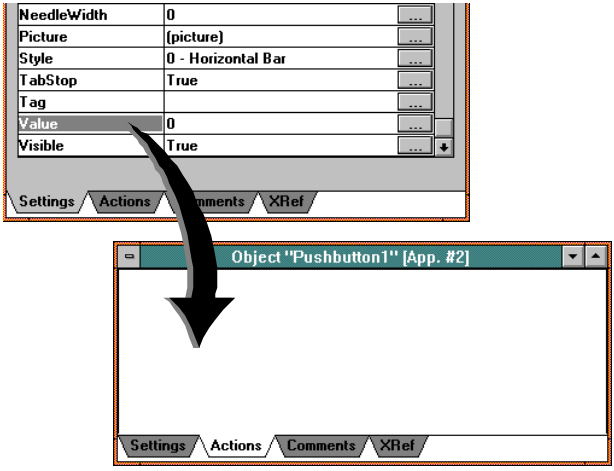
aus gesetzt werden, sondern müssen in Action Listen eingestellt werden, wie im nächsten Abschnitt beschrieben.

Ändern der Eigenschaften in Aktionen

Fast die gesamte Zusammenarbeit zwischen TestPoint und den VBX Controls wird durch Setzen und Lesen der VBX Eigenschaften erledigt. Sie können diese Eigenschaften im TestPoint Editor voreinstellen, wie bereits oben beschrieben.

Ziehen vom Settings Fenster

Der einfachste Weg, um in einer Action Liste auf eine Eigenschaft zuzugreifen, ist der gleiche, wie Sie auf andere Einstellungen in TestPoint zugreifen: Ziehen Sie die entsprechende Einstellung vom Settings Fenster in die Action Liste:



Sie können jede VBX Eigenschaft ändern durch Ablegen in eine neue Action Zeile, wie hier:

1) Set Gauge(Value) to83

oder durch Ablegen als Parameter verwenden, wie hier:

2) Calculate Math1 with x=Slider(Value)

Benutzen der „Set“ und „Get“ Aktionen

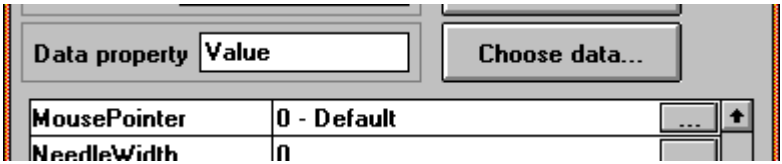
Das VBX Objekt stellt außerdem die „Set property“ und „Get property“ Aktion zur Verfügung, die auf Eigenschaften per Namen oder Index zugreifen kann.

Für die meisten Eigenschaften ist dies nur eine Alternative zum Ziehen aus dem Settings Fenster. Bei den Eigenschaften, die Felder verwenden, müssen Sie diese Methode verwenden, um auf einen Feldindex zuzugreifen:

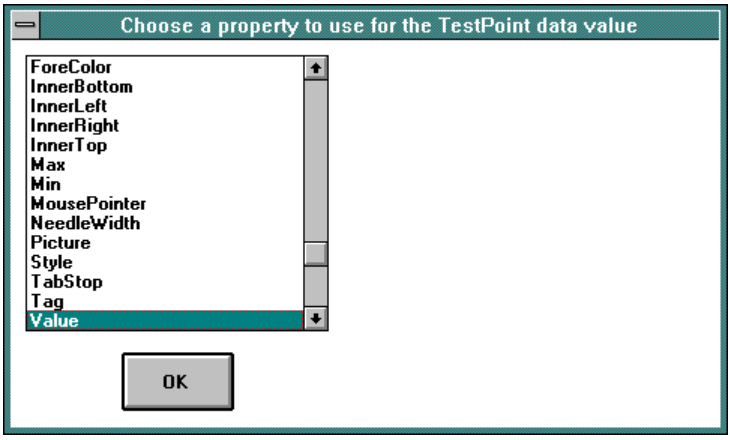
1) Set property of Fader property="FaderLabels", array index (optional)=2, value="Label2"

VBX Datenwerte

TestPoint Objekte haben einzelne Datenwerte. VBX Controls haben mehrere Eigenschaften. Die Einstellung „Data property“ im Settings Fenster des VBX Objektes beschreibt, welche Eigenschaft als Datenwert des Objektes verwendet werden soll.



Wenn ein VBX Control geladen wird, ist die Voreinstellung hier die „Data“ Eigenschaft. Sie können diese Eigenschaft jedoch sehr einfach ändern, indem Sie den „Choose Date“ Schalter anklicken und aus der Liste der verfügbaren Eigenschaften auswählen.



Viele Controls, wie Schieberegler, Knöpfe oder Dateneingabefelder werden diese „**Data**“ Eigenschaft haben, wie z. B. „**Value**“, die sich dann ändert, wenn der Anwender das Control beeinflußt.

Somit lassen sich die VBX-Objekte direkt in einer Aktions Liste benutzen:

1)	Set	Display1	to VBX-Knob

Wenn Sie gelegentlich auf andere Eigenschaften zugreifen möchten, verwenden Sie das Ziehen der Eigenschaft vom Settingsfenster oder die „Get property“ Aktion:

2)	Set	Display2	to VBX-Knob(Max value)
3)	Get property of	VBX-Knob	property="Min value", array index(optional)=___

Wann ist der Datenwert nicht der Datenwert?

Normalerweise ist der Datenwert des VBX Objektes die Eigenschaft, die von der „Data property“ Einstellung ausgewählt ist.

Trotzdem können bestimmte TestPoint Aktionen den Datenwert **vorübergehend** ändern. Das sind diese Aktionen:

- Get property
- Get last event

Diese ändern den Datenwert des Objektes, damit Sie das Ergebnis der Aktionen in der nächsten Action Zeile benutzen können (Beispiel: Um eine Anzeige zu erzeugen oder für eine IF/THEN/ELSE Abfrage). Diese Änderungen sind **vorübergehend**, weil sie nur für **einen** Zugriff auf die Daten gelten. Wenn Sie das Objekt zum nächsten Mal verwenden, wird der Datenwert wieder der sein, der definiert wurde.

Wenn Sie eine Eigenschaft als Feld auslesen und diesen Wert zur Anzeige bringen, ist es erforderlich, bevor Sie z. B. einen Vergleich durchführen, den Wert erneut zu lesen.

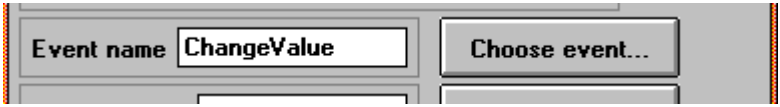
1)	Get property of	VBX1	property="Labels", array index(optional)=0
2)	Set	Display1	to VBX1
3)	Get property of	VBX1	property="Labels", array index(optional)=0
4)	If/Then	Equal	with x=VBX1, y="ABCD"
5)	Execute	Action1	
6)	End If	Equal	

VBX Ereignisse

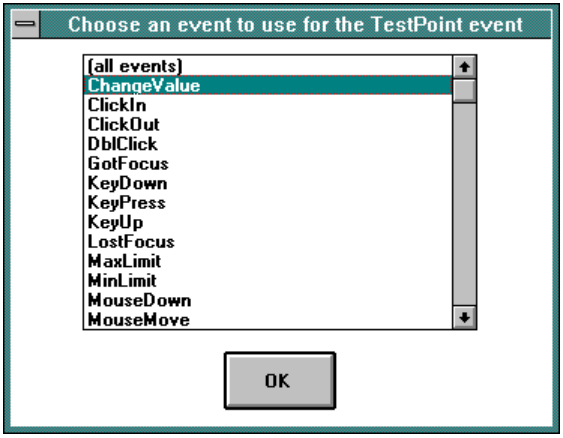
TestPoint Objekte haben eine Action List, die mit ihnen verknüpft ist. Viele Objekte führen ihre Action Liste aus, wenn ein Ereignis eintritt, die Objekte antworten jedoch immer nur auf **ein** Ereignis.

VBX Controls andererseits beinhalten **viele** Ereignisse, auf die sie reagieren können.

Der „Event name“ (Ereignis Name) in den Settings Fenster des VBX Objektes stellt eine Auswahl zur Verfügung, auf welches Ereignis die TestPoint Action Liste für das Objekt ausgeführt werden soll.



Beim Laden des Controls wird eine Voreinstellung gemacht, die Sie jedoch, durch einen Klick auf den „Choose event...“ Schalter und eine Auswahl aus der nachfolgenden Liste, ändern können:



Viele Eingabe Controls (Schieberegler, Knöpfe,...) haben verschiedenste Ereignisse, wie „Change“ oder „Change value“, die ausgelöst werden, wenn der Anwender den Wert des Controls modifiziert. Dadurch wird es leicht, mit einer Action Liste auf Änderungen der Eingabe Controls zu reagieren.

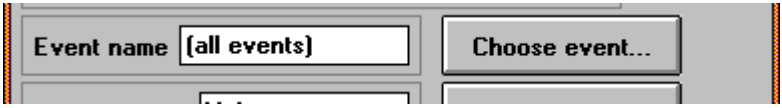
Der Schieberegler der VBX-Tools hat beispielsweise einige Funktionen, die der TestPoint Slider nicht hat, er kann aber auch genauso verwendet werden. Ist der VBX Schieberegler einmal geladen, kann er in die Action Liste gezogen werden und eine Action Zeile oder Parameter erzeugen:

1) Set Display1 to VBX-Slider

Diese Anzeige (Display1) wird immer dann aktualisiert, wenn der Schieberegler sich verändert hat, exakt so, wie das TestPoint Slider Objekt.

Mehrere Ereignisse

Wie aber geht man vor, wenn mehr als ein Ereignis der VBX Controls abgefragt werden muß? Klicken Sie einfach auf den „**Choose event...**“ Schalter und wählen Sie die „**(all events)**“ Option:



Die Action Liste wird jetzt ausgeführt, wenn irgendeins der VBX Ereignisse eintritt: Mausklick, Tastendruck, Wertänderung,...

Um herauszufinden, welches Ereignis nun wirklich eingetreten ist, verwenden Sie die passende Aktion für diesen Fall: die „Get last event“ Aktion des VBX Objektes. Diese Aktion übergibt eine Liste, die folgenden Inhalt hat: eine Zeichenkette mit dem Ereignisnamen und

dann kein oder mehrere Einträge, die Ereignis Argumente sind und von dem Ereignis selbst abhängen.

Die Ereignisse:

- Click
- DblClick
- GotFocus
- LostFocus

geben Listen mit einer Position zurück: Dem Ereignis Namen. Sie haben keine Ereignis Argumente.

Diese allgemein formulierten Ereignisse geben die folgenden Daten zusätzlich zum Ereignis Namen zurück:

Event	**Data list**
KeyDown	keycode, flags*
KeyPress	character-code
KeyUp	keycode, flags*
MouseDown	buttonflags**, flags*, x, y
MouseMove	buttonflags**, flags*, x, y
MouseUp	buttonflags**, flags*, x, y

* flags geben den Status der Umschalt- und STRG Tasten an: 0 für keine, 1 für Umschalttaste, 2 für STRG-Taste und 3 für beide.

** buttonflags geben den Status der Maustasten: 0 für keine, 1 für linke Taste, 2 für rechte Taste und 3 für beide Mausatsten.

Hier kann nun die mathematische Funktion select() angewendet werden, um auf einzelne Positionen der Liste zuzugreifen. Zum Beispiel mit einem Case Objekt mit folgender Formel:

select(event,0)

könnte in einer VBX Action Liste wie dieser verwendet werden:

1)	Get last event	VBX	
2)	Select	Case	with event=VBX
3)	When	Case	is "KeyPress"
4)	Execute	Action1	
5)	When	Case	is "MouseDown"
6)	Execute	Action2	
7)	End	Case	

Methoden

Einige VBX Controls benutzen das, was in der VBX Terminologie Methode genannt wird. Das sind Aktionen in TestPoint. Das wichtigste daran sind die „Add item“ und „Remove item“ Methoden, die oft dann Anwendung finden, wenn von VBX Controls Listen mit Informationen angeboten werden:

1)	Add item to	VBX-Listbox	item=Data-Entry, index=VBX-Listbox(ListCount)

Die „Move“ Methode ist ebenfalls nutzbar, um die Position und Größe eines Controls auf dem Panel zu verändern. Diese Möglichkeit steht auch für TestPoint Objekte zur Verfügung, als Aktion des Panel Objektes (z. B. auch für das Hauptpanel).

Hinweise zu VBX Controls

Hinweis: Bei den meisten VBX Controls gibt es kostenlose Lizenzen für Runtimes, wie bei TestPoint. Die VBX Dateien, die mit TestPoint geliefert werden, sind dieser Art. Sie können nicht in den TestPoint Editor geladen werden, ohne daß eine Fehlermeldung erzeugt wird. Wenn Sie diese Controls vom Anbieter installieren, wird die Lizenz, um neue Anwendungen zu erzeugen, automatisch mit eingerichtet.

Beachten Sie, daß es verschiedene Ebenen der VBX Schnittstellen Definition gibt. TestPoint, C++ und einige andere Tools verwenden VBX Ebene 1 (Level 1). Die Mehrheit von VBX Controls arbeitet auf diesem Level, aber einige verwenden die speziellen Funktionen von VB 2.0 oder 3.0. Wenn die Controls mit Borland oder Microsoft C++ arbeiten, sind sie auch kompatibel zu TestPoint. Gute VBX Controls geben Warnungen beim Laden aus, wenn Sie mehr als Ebene 1 voraussetzen. **Einige kommerzielle Controls beachten diese Richtlinien nicht und stürzen einfach ab, wenn sie mit der falschen Version verwendet werden.**

VBX Controls bestehen aus externem Code, der in TestPoint verwendet werden kann. Wir fanden heraus, daß die meisten von hoher Qualität sind. Trotzdem sind einige mit bekannten Fehlern behaftet, die vermieden werden müssen. Beispielsweise erzeugt das 3D-Menü Control der VBTools eine Schutzverletzung, wenn die „Popupcreate“ Eigenschaft ohne die Initialisierung anderer Eigenschaften verwendet wird. Das gleiche gilt, wenn Sie dieses Control in Visual Basic verwenden.

Wenn Sie auf irgendein Problem mit VBX Control treffen, versuchen Sie, das Problem mit Visual Basic, Borland- oder Microsoft C++ zu reproduzieren. Dadurch erhalten Sie Aufschluß darüber, ob dies ein Problem des VBX Controls selber ist oder nicht.

Eigene VBX Controls erzeugen

VBX Controls sind ein allgemein verfügbarer Standard, der von Microsoft unterstützt wird. Microsoft verkauft ein Control Entwicklungs Paket, das auch zur Visual Basic 3.0 Professional Version gehört. Auch bei Microsoft Entwickler Softwarepaketen, wie C Compiler, ist dies enthalten.

Normalerweise werden VBX Controls in C geschrieben (obwohl auch Pascal verwendet werden könnte). Microsoft´s Pakete enthalten einige Beispiele.

Außerdem gibt es einige Magazine und Bücher, die sich mit diesem Thema befassen.

Eigene VBX Controls sind dann sinnvoll, wenn Sie eigene Benutzerschnittstellen Objekte schaffen oder eine Schnittstelle zu eigener Hardware zur Verfügung stellen wollen. Durch VBX Controls haben Sie die Möglichkeit, zum einen von TestPoint, aber auch von anderen Entwicklungssystemen, auf diese Schnittstelle zugreifen zu können.

Weitere Techniken

Kapitel 15. Objekte modifizieren

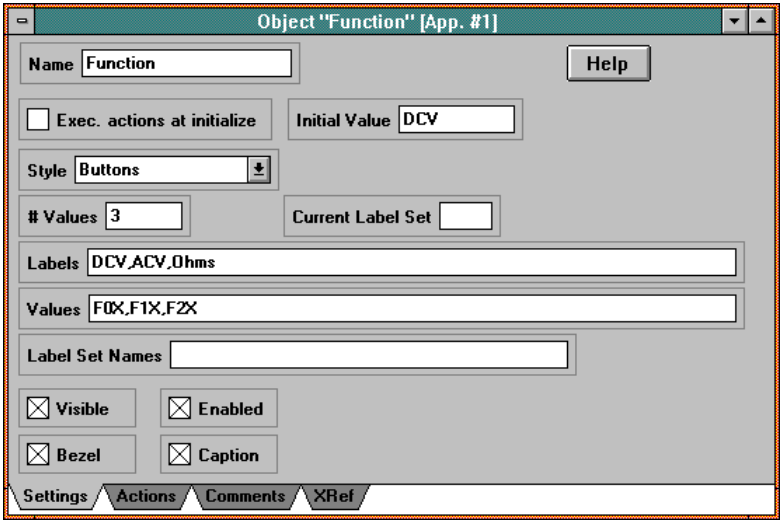
Was in diesem Kapitel behandelt wird:

- Was Objekteinstellungen sind.
- Wie man Einstellungen bei laufender Anwendung ändert.

Verändern von Einstellungen

TestPoint Objekte bieten Einstellungen, welche deren Aussehen auf dem Arbeitsfenster und das Verhalten zur Laufzeit bestimmen.

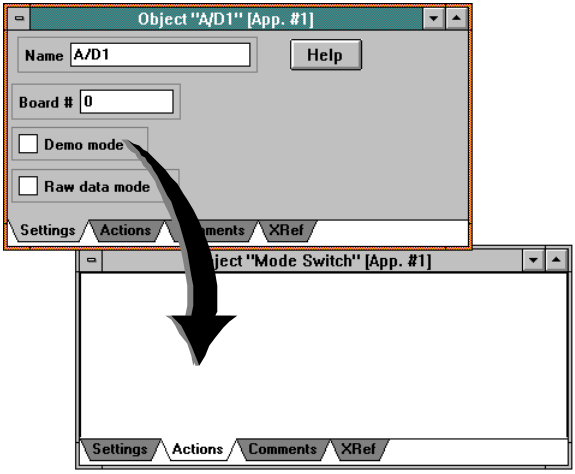
Sie können verändert werden, indem Sie im TestPoint Editor auf das zu verändernde Objekt im Arbeitsfenster oder in der Objektliste doppelklicken. Das ‘settings’ Fenster für dieses Objekt erscheint und beinhaltet immer den Objekt Namen.



Informationen über ein Objekt sind in den Einstellungen, die gewöhnlicherweise vom Programmentwickler voreingestellt werden, enthalten. Informationen, die sich von einem Durchlauf der Action Liste zum nächsten Durchlauf variieren, werden als Parameter implementiert und erscheinen als leeres auszufüllendes Feld in der Action Liste. Zum Beispiel: die Timeout Einstellung für ein GPIB Gerät Objekt ist eine Einstellung, weil gewöhnlicherweise ein konstanter Wert genügt. Die maximale Anzahl Bytes, gelesen von der "Enter from" Aktion des GPIB Objekts, ist ein Parameter, da er jedesmal, wenn er in der Anwendung verwendet wird, anders sein kann.

Ändern während der Ausführung

Oft ist es jedoch notwendig, diese eigentlich konstanten Werte während der Laufzeit des Programmes zu verändern. Man kann dazu die mit einem grauen Rahmen versehenen Optionen des ‘settings’-Fensters in die Befehlsliste ziehen. In den Aktionszeilen verhalten sich diese editierten Einstellungen genauso wie alle anderen Objekte.



Um eine Aktion hinzuzufügen, die ein ‘setting’ verändert, öffnen Sie nur das Fenster, ziehen dann das Feld in die Action Liste. Abhängig vom Typ des Objekts, das für das Setting verwendet wurde, bekommen Sie eine entsprechende Befehlszeile. In einer Befehlszeile erscheint der Name der Einstellung eines Objekts immer in Klammern nach dem Objektnamen.

Kapitel Rückblick:

- **Settings**: Jedes Objekt besitzt Einstellungen, welche das Erscheinungsbild und das Verhalten des Objekts darstellen. Diese ‘settings’ werden zugeordnet, wenn ein Objekt vom ‘stock’ hinzugefügt wird und können später durch Doppelklick auf das Objekt wieder editiert werden.
- **Settings actions**: ‘Settings’ können auch während der Laufzeit eines Programms geändert werden, indem Sie den gewünschten Parameter in die Action Liste ziehen. Der neue Wert für diese spezielle Einstellung kann eine Variable oder ein fester Wert sein.

Kapitel 16. Datenspeicher

Was in diesem Kapitel behandelt wird:

- Arbeitsweise des Container Objects.
- Zwischenspeichern von Daten.
- Sammeln von Daten innerhalb einer Schleife.

Das Container-Objekt ist ein Datenspeicher.

Alle Objekte innerhalb von TestPoint speichern Daten. Das GPIB Objekt z. B. bekommt seine Werte von den Geräten, die aus dem TestPoint heraus gesteuert werden. Als zusätzliche Speicherstelle eignet sich das Container-Objekt hervorragend.

Sie sind mit einer Variablen in den herkömmlichen Programmier-sprachen vergleichbar. Im Speicher können alle Arten von Daten im TestPoint gespeichert werden: 'numbers', 'strings', 'vectors' und andere. Die ‘Container’ sind Zwischenspeicher, deren Inhalte bei jedem Neustart einer Applikation gelöscht werden. Sollen Daten langwierig gespeichert werden, können sie mit dem File Objekt auf Diskette oder Festplatte geschrieben werden.

Datenspeicherung

Das Container Objekt bildet einen Zwischenspeicher für eingehende Daten. Er ist dann besonders sinnvoll, wenn die Daten für den späteren Ablauf einer Actions Liste festgehalten werden sollen.

1)	Enter from	Meter	up to 256 bytes, stop on EOS=LF
2)	Store in	Container	from Meter
3)	Enter from	Meter	up to 256 bytes, stop on EOS=LF
4)	Calculate	Smoothed	with x=Meter y=Container

Der Container sammelt Meßwerte ein, während bereits ein neuer Wert aufgenommen wird. Danach erfolgt mit dem Mathematik Objekt eine Mittelwert- und Glättungsberechnung. Würde der Container nicht angesprochen werden, würde die zweite GPIB 'Enter-Action' den ersten Meßwert ersetzen und es gäbe keine Möglichkeit, aus zwei verschiedenen Werten etwas zu berechnen.

Daten hinzufügen

Neben der beschriebenen Zwischenspeicherung existiert noch die Möglichkeit einer fortlaufenden Speicherinhaltserweiterung. Mit „Append“ werden Werte zu den bereits existierenden Daten im Container hinzugefügt. Dies unterscheidet den Container von anderen Objekten, in denen nach Ablauf die neuen Werte einfach die vorhergehenden Daten überschreiben. Die Append Funktion bildet einen Vector aus einzelnen Werten.

Sie können sich diese Append-Funktion des Containers als einen Papierstreifen einer Rechenmaschine oder Registrierkasse vorstellen. Jede neue Append-Funktion addiert eine neue Zeile von Werten auf dem Streifen hinzu. Es ergibt sich ein zeitliches Protokoll aller ausgeführten Append-Funktionen. Mit 'Clear' wird der Speicherinhalt vor dem Neustart gelöscht.

Folgende Applikation liest Meßwerte in einer Schleife ein und speichert die aufgenommenen Daten in einem Container (Curve). Die Schleife durchläuft mehrere Frequenzwerte eines Funktionsgenerators, liest die Spannungswerte ein und zeichnet nach Ablauf der Messung die Kennlinie.

1)	Clear	Curve	
2)	Decade series	Frequency	from 10^1 to 10^3,
3)	Output to	FnGen	with "FREQ" , Frequency term.=LF
4)	Enter from	Meter	up to 256 bytes, stop on EOS=LF
5)	Append to	Curve	from Frequency, Meter
6)	End	Frequency	
7)	Draw graph	Response	with Curve

Der Container ‘Curve’ wird bei dem Start der Action Liste gelöscht. Innerhalb der Schleife werden die Werte für die Frequenz und die Spannung in einer Liste mit einem Wertepaar durch die ‘Append’-Funktion angehängt. Ist die Schleife durchlaufen, besteht der Containerinhalt aus einer Liste zweier numerischer Vektoren.

Kapitel Rückblick:

- Container: Das Container Objekt bildet einen Zwischenspeicher für Daten, damit sie später in derselben oder einer anderen Action Liste der Applikation zur Verfügung stehen.
- Daten anfügen: Mit der Append-Funktion werden einzelne Werte in einen Vektor von Werten geschrieben. Jede Append Operation fügt ein neues Vektorelement, wie z.B. neue Zeilen einer Tabelle, hinzu. Außerdem können auch Listen mehrere Werte, wie z.B. neue Spalten einer Tabelle, hinzugefügt werden.

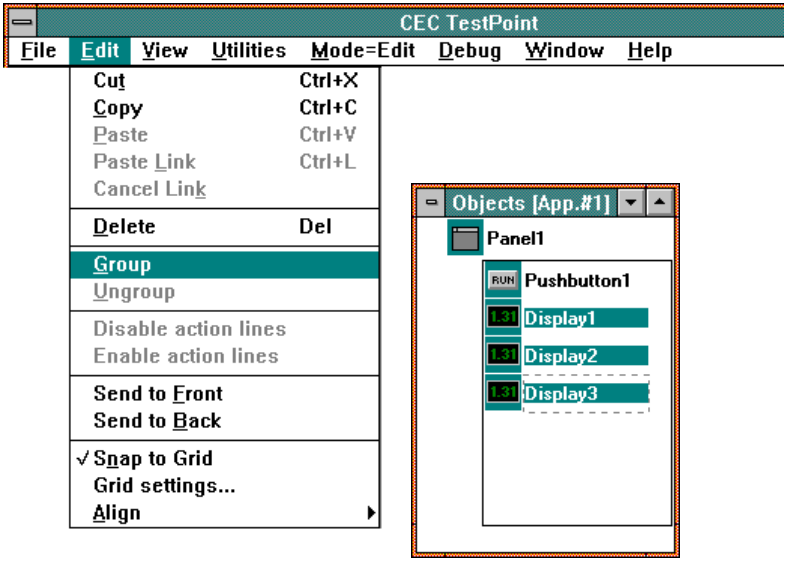
Kaitel 17.
Gruppen & Indirekte Objekte

Was in diesem Kapitel behandel wird:

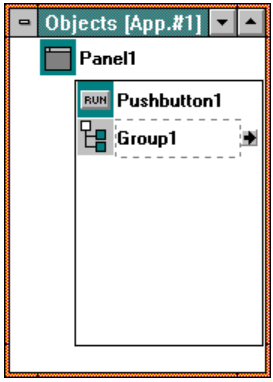
- Gruppieren von Objekten.
- Ganze Objektgruppen in Aktionen verwenden.
- Indirekte Objekte erzeugen und wozu diese zu verwenden sind.

Gruppen - Vereinfachen der Objekt Liste

TestPoint Objekte können in Gruppen zusammengefaßt werden, indem sie (in der Objekt Liste oder auf dem Panel) markiert werden, und das Menükommando EDIT/GROUP angewendet wird:

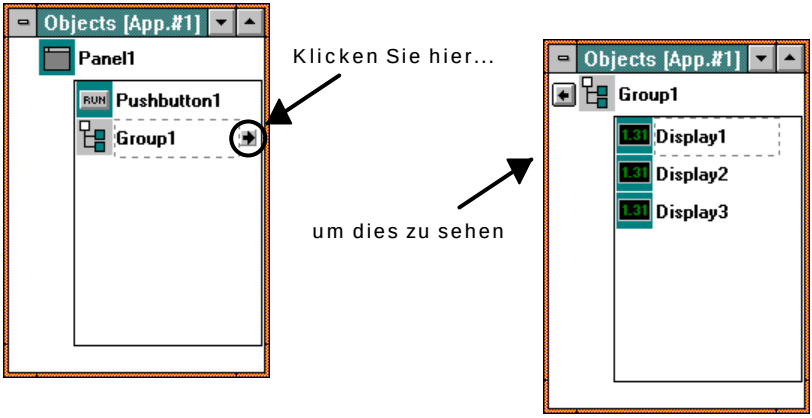


Dadurch wird ein neues „Group“-Symbol in der Objekt Liste erzeugt, das die Original Objekte enthält:



+ Wenn Ihre Anwendung anfängt, komplex zu werden, und Sie die Objekt Liste kürzer gestalten wollen, können Sie verwandte Objekte gruppieren.

Um einzelne Objekte zu sehen, klicken Sie nur auf dem Pfeil rechts neben dem Group Objekt:



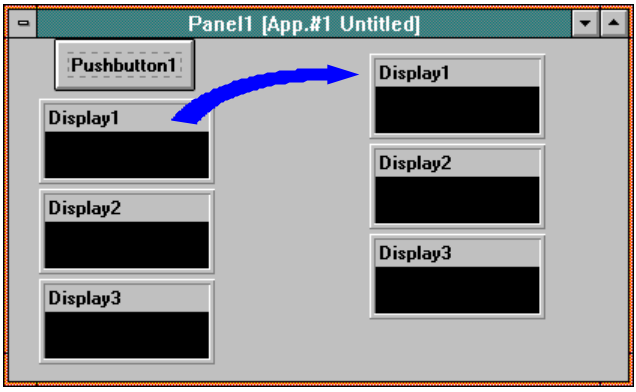
+ Sie können trotzdem die Objekte in Action Listen ziehen.

Hinweis: Sie können jede Art von Objekten zusammen gruppieren. Wenn Sie diese als Objektfeld oder für einen einfacheren Zugriff gruppieren, sollten Sie nur gleiche Objekte verwenden (siehe Abschnitt „Gruppieren als Objektfeld“ später in diesem Kapitel).

+ Sie können die Gruppierung rückgängig machen, indem Sie die Gruppe markieren und das Menükommando EDIT/UNGROUP wählen.

Gruppen - Objekte zusammen verschieben

Wenn Objekte gruppiert sind und ein Objekt dieser Gruppe auf dem Panel verschoben wird, bewegen sich alle Objekte:



+ Um nur ein Objekt der Gruppe zu verschieben, halten Sie während des Verschiebens die Umschalt-(Shift)Taste gedrückt.

Vorder-/Hintergrund und Tabulator-Reihenfolge

Wenn Objekte in Objekt Listen gruppiert sind, sind sie dies aus Sicht der Anzeige und der Tabulator Reihenfolge.

Wenn Sie **ein** Objekt der Gruppe **auf dem Panel** markieren und das Menükommando EDIT/SEND TO FRONT oder SEND TO BACK benutzen, wird diese Reihenfolge **nur innerhalb der Gruppe** angewendet.

Wenn Sie die **gesamte Gruppe in der Objekt Liste markieren** und entsprechend behandeln, wird die Gruppe in den Vorder- oder Hintergrund gebracht.

Gruppieren als Objektfeld

Wenn Sie Objekte gruppieren, hat das neue Group Objekt Aktionen, die auf die gesamte Gruppe anwendbar sind. Somit können gleiche Objekte in Gruppen wie ein Feld behandelt werden, auf das über einen Index zugegriffen werden kann.

Aktionen

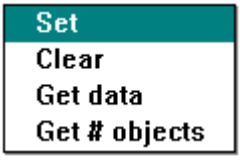
Gruppieren Sie beispielsweise drei Display Objekte und ziehen Sie das sich daraus ergebende Group Objekt in eine Action Liste. Sie erhalten ein Popup Menü mit folgenden Aktionen:

Die „**Get Data**“ und „**Get # objects**“ Aktionen werden vom Group Objekt angeboten und werden unten beschrieben.

Die erste Aktion, „**Set**“, ist die Aktion des Display Objektes.

+ **Eine Group Objekt besitzt alle Aktionen, die für das erste Objekt in dieser Gruppe verfügbar sind.**

Gruppieren Sie beispielsweise ein Indicator Objekt, bekommen Sie ein Popup Menü, dessen ersten beiden Aktionen vom Indicator Objekt kommen.



Verwenden Sie die „Set“ Aktion, erhalten Sie folgende Action Zeile:

1) Set Group1 to _____ [index=_____]

Dies ist die normale „Set“ Action Zeile, die den **zusätzlichen Parameter** „index=“ enthält.

+ **Group Aktionen haben einen Index Parameter, der Objekte aus der Gruppe auswählt, die entsprechende Aktionen ausführen sollen.**

Indizierung

Mit dem Index Parameter können Sie eines der drei Display Objekte in einer Action Zeile verwenden:

1) Set Group1 to "abc" [index=0]

setzt das erste Display Objekt in der Gruppe auf den Wert „abc“. Beachten Sie, daß die Indizierung bei 0 beginnt und der Index eine Konstante oder eine Variable sein kann.

2) Set Group1 to "xxx" [index="all"]

setzt alle drei Display Objekte auf „xxx“.

3) Set Group1 to A/D1 [index="each"]

setzt die Display Objekte in eine Reihenfolge zu den Werten des A/D1 Objektes, in der Annahme, im A/D1 Objekt befindet sich ein Listen Datentyp mit mehreren A/D Eingangskanälen.

4) Set Group1 to A/D1 [index="0,2"]

setzt das erste und dritte Display Objekt, basierend auf dem Listen Datentyp des A/D1 Objektes. Beachten Sie, daß Sie hier die Kanalliste, die für die Datenerfassung verwendet wurde, einsetzen können und die A/D Daten direkt mit den Display Objekten korrespondieren! (Siehe Beispiel ADGROUP.TST)

Mögliche Index Parameter sind:

eine Zahl zeigt auf ein einzelnes Objekt in der Gruppe. Der Index beginnt bei Null. Ist der Index größer als die Anzahl der Objekte in der Gruppe, wird keine Aktion ausgeführt.

"all" führt die Aktion für alle Objekte in der Gruppe mit den gleichen Parametern aus. Die Reihenfolge der Ausführung ist die der Objekte in der Gruppe.

"each" führt die Aktion für alle Objekte in der Gruppe aus, jedoch sind die Parameter so angepaßt, daß sie auf die Indizes des Objektes passen. Parameterwerte sind:

Objekte mit Vektordaten	werden indiziert. Die Argumente werden an die Aktionen jedes Objektes übergeben, das mit dem Objektindex in der Gruppe korrespondiert.
Objekte mit Listendaten	werden indiziert. Die Argumente werden an die Aktionen jedes Objektes übergeben, das mit dem Objektindex in der Gruppe korrespondiert, wenn das Listenelement eine Zahl oder eine Zeichenkette ist. Andere Listenelemente, wie Vektoren, werden zu „nodata“ Argumenten.
alle anderen	werden nur für die Aktion des ersten Objektes verwendet. Alle anderen Objekte in der Gruppe werden zu „nodata“ für diesen Parameter gesetzt.

eine Liste mit Zahlen die in Form einer Zeichenkette, wie „0,2,3“ oder eines Objektes, dessen Datenwert eine Liste oder

ein Vektor ist, vorliegen kann, führt das Objekt aus, dessen Aktionen durch die Objekt Indizes spezifiziert werden. Dabei muß die Reihenfolge nicht sequenziell sein. Die Argumente werden wie im „Each“ Fall behandelt.

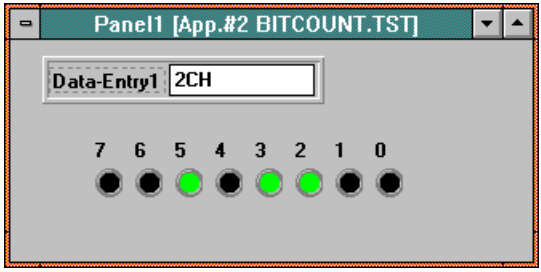
Einstellen aller Objekte

Eine Gruppe von Display-, Indicator oder anderen Objekten kann mit dem „**all**“ Parameter auf einen bestimmten Wert zurückgesetzt werden:

1) Set Display-Group to _____ [index="all"]

Vektordaten auf mehrere Objekte setzen

Vektoren, die von Messungen oder Berechnungen kommen, können mit Hilfe mehrerer Objekte angezeigt werden, indem eine Gruppe und der „each“ Index verwendet wird. Dies ist hilfreich, wenn Sie zum Beispiel numerische Werte in einer Gruppe von Indicator Objekten darstellen wollen:



Hier ist die Action Liste für das Dateneingabefeld:

1) Calculate BitVector with x=Data-Entry1, n=8
2) Set Ind-Group to BitVector [index="each"]

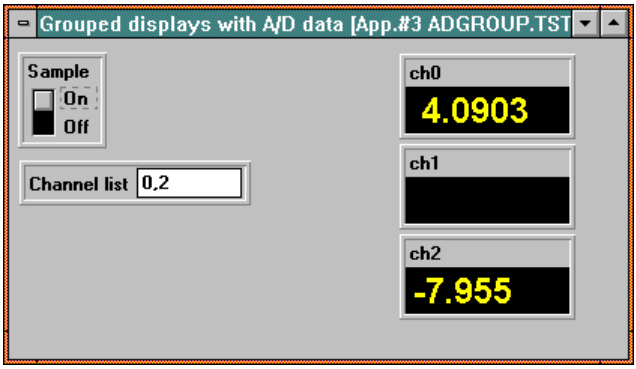
und die Formel für das Math Objekt „BitVector“, der eine Zahl in einen Vektor aus Nullen und Einsen umwandelt, lautet:

sgn(x and pow2(ramp(n)))

Zeile 1 erzeugt den Vektor aus Nullen und Einsen und Zeile 2 setzt den gesamten Vektor in die Gruppe aus Indicator Objekten, um eine binäre Anzeige zu bekommen.

A/D Daten auf mehrere Objekte setzen

Wenn Sie Meßdaten von mehreren A/D Kanälen erfassen, ist es eine typische Anwendung, diese Daten in verschiedenen Anzeigen oder Balkengrafiken darzustellen. (Dieses Beispiel ist ADGROUP.TST)



Gruppen lösen diese Aufgabe leicht. Die Verwendung der Kanalliste in der A/D-Aktion und in der „Set“ Aktion der Gruppe, stellt die Kanäle immer in den entsprechenden Fenstern dar, sogar wenn die Reihenfolge nicht entsprechend ist.

Die Action Liste des „Sample“ Schalters ist:

1)	Do loop	Loop1	while Sample is true
2)	Sample A/D	A/D1	once, channel(s)=Channel-list
3)	Set	Displays	to A/D1 [index=Channel-list]
4)	End	Loop1	

Diese nimmt solange Werte der im Eingabefeld „Channel list“ angegebenen Kanäle auf und zeigt sie in den Anzeigeobjekten an, bis der Schalter auf „off“ geschaltet wird.

Die Action Liste des „Channel list“ Objektes ist:

1) Set Displays to _____ [index="all"]

Diese löscht alle Anzeigen (Display Objekte) immer dann, wenn die Kanalliste verändert wird. Wenn z. B. die Liste „0,1,2“ gewesen ist, sind in jeder Anzeige Werte dargestellt. Wird nur „0,2“ eingegeben, muß der Wert für die Anzeige „ch1“ bei der nächsten Aktualisierung gelöscht werden.

Daten von mehreren Objekten lesen

Die vorhergehenden Beispiele handeln alle vom Einstellen der Objekte durch die Indizierung. Um Datenwerte von mehreren Objekten einzulesen, können Sie die „**Get data**“ Aktion verwenden, die das Group Objekt zur Verfügung stellt:

```
1) Get data          Group1          [index="all"]
```

Die Indizierung wurde bereits vorher beschrieben. Die Aktion setzt den Datenwert des Group Objektes auf eine Liste mit allen Datenwerten der Objekte der Gruppe.

Wie viele Objekte?

Generell erkennen Gruppenobjekte die Anzahl der Objekte in der Gruppe automatisch. Der „all“ Indexwert behandelt alle Objekte, egal wie viele. Indizes, die außerhalb des momentan gültigen Bereichs liegen, werden übersprungen (z. B. eine Kanalnummer größer als 2 im vorhergehenden Beispiel).

Es kann auch Fälle geben, in denen Sie eine Anwendung schreiben, in der auf die Objekte der Gruppe einzeln in einer Schleife zugegriffen werden muß. Dies ist einfach einzufügen. Die „**Get # objects**“ Action wird dazu angeboten:

```
1) Get # objects     Group1
2) Calculate         N-1             with N=Group1
3) Linear series     Loop1           from 0 to N-1
4) Set               Group1          to Loop1  [index=Loop1]
5) End               Loop1
```

Die „Get # objects“ Aktion setzt den Datenwert der Gruppe auf die Anzahl der Objekte in der Gruppe.

Gemischte Objekttypen in der Gruppe

In Gruppen können Sie beliebige Objekte kombinieren. Die Objekte müssen nicht gleich sein.

Wenn Sie indizierte Aktionen mehrerer Objekte benutzen, sollten die Objekte der gleichen Art sein. Die Aktionen des Group Objektes sind immer die des ersten Objektes in der Gruppe und können unbeabsichtigte Ergebnisse hervorrufen, wenn sie auf andere Objekte angewendet werden.

Die einzige Ausnahme davon ist die Kombination von Objekten mit nur einer Aktion: „**Set**“. Viele Benutzerschnittstellen Objekte, wie Display- und Bar Objekte haben nur eine Aktion. Werden diese gruppiert, kann die „Set“ Aktion für alle Objekte dieser Art verwendet werden.

Indirekte Objekte - Platzhalter oder Zeiger

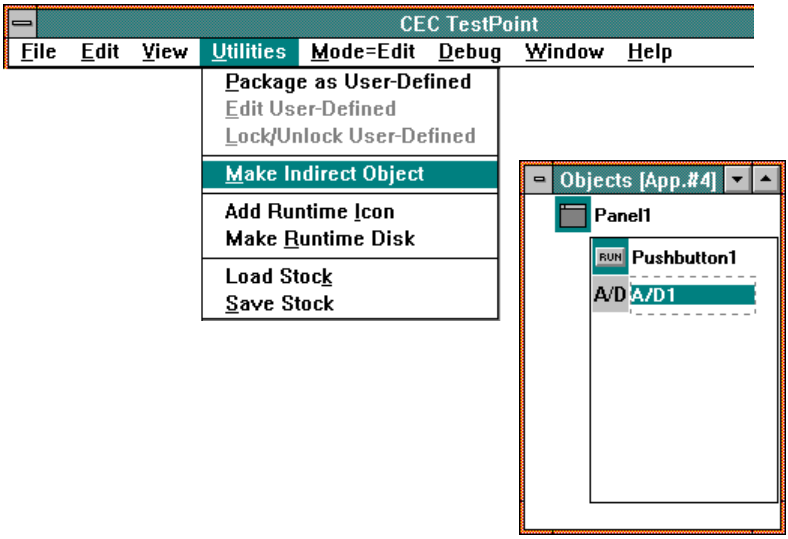
Ein indirektes Objekt ist ein Platzhalter und kann in Action Listen verwendet werden, die später mit einem anderen Objekt ausgeführt werden sollen. Dies ist eine leistungsstarke erweiterte Programmier Funktion, sehr nützlich bei der Erstellung von Action Objekte, die in Ihrer Anwendung an vielen Stellen wiederverwendet werden.

Es ist außerdem hilfreich bei der Entwickung benutzerdefinierter Objekte, wenn Sie Aktionen eines Objektes ausführen wollen, das erst vom Anwender des Objektes bereitgestellt wird. Da das Objekt, mit dem Sie arbeiten, nicht vorhanden ist, wenn Sie das benutzerdefinierte Objekt erstellen, können Sie es auch nicht in die Action Liste ziehen. Statt dessen verwenden Sie das indirekte Objekt als Platzhalter.

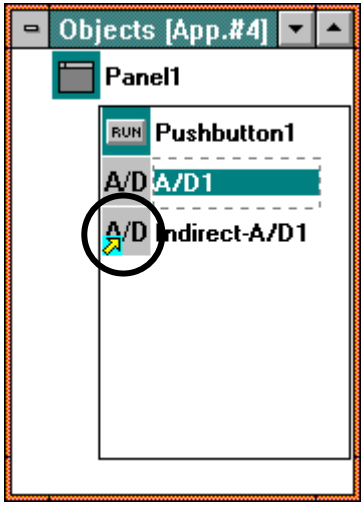
Indirekte Objekte sind gleichzusetzen mit Zeigern (Pointer) in konventionellen Programmiersprachen, wie Pascal oder C.

Erzeugen indirekter Objekte

Sie erzeugen indirekte Objekte, indem Sie ein existierendes Objekt markieren und das Menükommando UTILITIES/MAKE INDIREKT OBJEKT auswählen:



Dadurch wird ein neues, indirektes Objektsymbol erzeugt, das aussieht wie das Original, jedoch in der Ecke einen kleinen Pfeil hat:



+ Ein indirektes Objekt ist ein Platzhalter für ein spezifisches Objekt.

Daher kann ein indirektes Objekt eines A/D Objektes nur für Aktionen anderer A/D Objekte verwendet werden.

Benutzung indirekter Objekte in Aktionen

Wenn Sie ein indirektes Objekt in eine Action Liste ziehen, erhalten Sie das gleiche Popup Menü wie bei einem normalen Objekt dieser Art. Die Action Zeile, die erzeugt wird, hat nur einen Parameter mehr, den „**actual object**“ Parameter.

1) Set Indirect-Display to _____[actual object=_____]

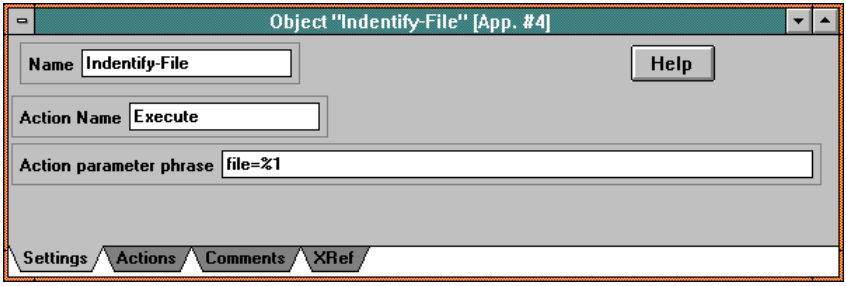
Dieser Parameter muß eine Referenz zu einem anderen Objekt der richtigen Gattung besitzen (in der dargestellten Zeile muß dies ein Display Objekt sein).

Als Beispiel, wie dies angewendet wird, stellen Sie sich vor, Sie wollen eine Action List erzeugen, die, bei gegebener Datei, versucht zu bestimmen, welcher spezielle Dateityp vorliegt (wie selbsterstellte Datendateien, Excel Dateien, usw.). Dazu sind einige Schritte erforderlich und Sie planen, später weitere anzufügen.

Das wichtigste ist, **Sie wollen eine wiederverwendbare Software Komponente schaffen**, mit der Sie verschiedene TestPoint File Objekte in der Anwendung verwenden können. Diese Action Liste muß Aktionen des File Objektes ausführen, um Dateien einzulesen**, darf aber kein vorgefertigtes Objekt verwenden**, um diese Action Zeilen zu erzeugen. Stattdessen **erstellen Sie die Action Liste mit einem Platzhalter und setzen das reguläre Objekt später ein**.

Auf diese Art erzeugen Sie ein indirektes File Object. Nehmen Sie nun ein neues Action Objekt in Ihre Anwendung auf.

Mit Action Objekten können Sie Parameter an Action Listen übergeben, indem Sie die „Action parameter phrase“, wie in diesem Beispiel, definieren:



Benutzen Sie dann das indirekte File Objekt, um eine Action Liste für das Action Objekt zu definieren:

```
1) Open              Indirect-File    [actual object="%1"]
2) Input from        Indirect-File    up to 80 bytes, stopping at ___
                                          [acutal object="%1"]
```

... usw.

Das Argument **%1** wird zum Action Objekt übergeben und wird später das „echte“ File Objekt beinhalten. Zeile 1 und 2 benutzen aber bereits Aktionen dieses Objektes.

Sie können das Action Objekt wie folgt verwenden:

```
1) Execute           Identify-File    with file=File1
```

An verschiedenen Stellen Ihrer Anwendung kann das Objekt File1, File2 usw. in die gleiche Action List übergeben werden.

Das Beispiel INDIREKT.TST, das mit TestPoint ausgeliefert wird, zeigt ein weiteres Beispiel für indirekte Objekte.

Benutzung indirekter Objekte in benutzerdefinierten Objekten

Wenn Sie bereits Action Objekte erstellt haben, die andere Objekte übergeben bekommen und indirekte Objekte verwenden, um auf diese zuzugreifen, können Sie diese in benutzerdefinierte Objekte packen.

Dadurch erstellen Sie ein neues Objekt, dessen interne Einzelheiten versteckt sind, das universell eingesetzt werden und sogar Aktionen anderer Objekte übernehmen kann.

Benutzerdefinierte Objekte werden in Kapitel 22 behandelt.

Kapitel 18.
Initialisierung & Beenden

Was in diesem Kapitel behandelt wird:

- Initialisierungsphase.
- Wann Initialisierung stattfindet.
- Wie die Initialisierung verwendet wird.
- Beendigungsphase.

Initialisierung

TestPoint ist ereignisgesteuert. Das bedeutet, daß Befehlslisten ausgeführt werden, wenn ein Hardware Ereignis eintrifft (wie z. B. Interrupts) oder als Antwort auf eine Benutzereingabe.

Wie auch immer, es ist oft nützlich, Aktionen schon beim Starten der Anwendung auszuführen, noch bevor ein Anwender eine Taste gedrückt oder andere Information eingegeben hat. TestPoint stellt für diesen Zweck Initialisierungsfunktionen bereit.

Initialisierung findet statt, wenn TestPoint eine Anwendung startet. Im TestPoint Editor ist es der Moment, wenn das Menükommando 'Mode' auf 'Run' gesetzt wird.

Die Initialisierung wird in drei Schritten durchgeführt: Hardware, Datenwerte, Aktionen.

Hardware Initialisierung

Die Hardware Initialisierung kommt immer für jede Hardware vor, die Sie in Ihrer Anwendung verwenden. Für korrektes Funktionieren ist es notwendig, jede Hardware in einen definierten Zustand zu bringen. Die in Ihrer Anwendung über Objekte eingebundene Hardware wird einmalig initialisiert.

Initialisierung von Datenwerten

Alle Daten von Objekten werden beim Starten auf einen Initialisierungswert gesetzt. Nicht alle Objekte haben derartige Initialisierungswerte (einige besitzen einen festen Wert). Objekte, deren Daten nicht sinnvoll sind, bis eine Aktion für dieses Objekt ausgeführt wird (wie z. B. MATH Objekt), besitzen keine Initialisierungswerte. Der Wert kann auf 'blank' oder 'no data' gesetzt sein, wenn keine guten Voreinstellungswerte zur Verfügung stehen.

Initialisierung von Aktionen

Die Initialisierung der Action Listen ist optional. Generell haben Benutzerschnittstellen-Objekte eine Einstellung, die ermöglicht, daß die Befehlsliste beim Initialisieren ausgeführt wird. Wenn mehrere Action Listen zur Initialisierung vorliegen, so ist die Reihenfolge der Ausführung zufällig. Es ist dadurch wichtig, in manchen Anwendungen die Initialisierung in genauer Reihenfolge zu durchlaufen, wobei die Initialisierung für nur eine Action Liste eingeschaltet werden sollte. Dies könnte z. B. die Befehlsliste eines ‘Action’ Objekts, eines Druckschalters (welcher auf ‘invisible’ gesetzt werden könnte), oder die Verwendung eines ‘Task’ Objektes mit der Option ‘Execute Action List’ als Initialisierung sein.

Beendigung

Eine TestPoint Anwendung wird beendet, wenn Sie diese schließen oder wenn Sie vom Run Modus zum Edit Modus wechseln.

Die Beendigung hat zwei Phasen: Aktionen und Hardwarebeendigung.

Beendigungs Aktionen

Es kann nützlich sein, eine Action Liste ablaufen zu lassen, wenn eine Anwendung beendet wird, um Geräte in einen definierten Zustand zu bringen oder andere „Aufräumarbeiten“ zu verrichten. Weil der Endanwender die Anwendung auf viele Arten schließen kann, brauchen Sie einen Weg, eine Beendigungs Action Liste auszuführen.

Das **Task** Objekt hat eine Einstellungsoption, die ermöglicht, dessen Action Liste bei Beendigung ausführen zu lassen. Um also eine Beendigungs Action Liste zu erzeugen, verwenden Sie das Task Objekt, setzen dessen Modus entsprechend und fügen die Aktionen in seine Action Liste.

Hardware Beendigung

Alle Geräte wie A/D Karten, RS-232 Ports, usw. werden geschlossen, wenn TestPoint beendet wird. Auch die File Objekte schließen Ihre Dateien.

Kapitel Rückblick:

- **Initialisierungsschritte:** Es gibt drei Phasen der Initialisierung: Hardware, Datenwerte und Actions. Die meisten Benutzerschnittstellenobjekte wie z. B. Dateneingabefelder, Kippschalter, usw. haben Einstellungen für Initialisierungswerte und für das Einschalten der Action Liste in der Initialisierungsphase.
- **Benutzung von Aktionen:** Sie wollen vielleicht einige Action Listen als Initialisierung starten. Normalerweise ist es am besten, nur eine Action Liste zur Initialisierungsphase zu starten. Will man mehrere Listen starten, so ist die Reihenfolge der Abarbeitung unbestimmt.

Kapitel 19. DDE & OLE: Zusammenarbeit mit anderen Programmen.

Was in diesem Kapitel behandelt wird:

- Was ist DDE und wie wird es benutzt.
- Wie werden Daten in Tabellenkalkulationen gebracht.
- Wie in Textverarbeitungsprogrammen.
- Meßbericht Erzeugung.
- DDE und Netzwerke.

Was ist DDE?

DDE steht für ‘Dynamic Data Exchange’ (dynamischer Datenaustausch), eine Möglichkeit in Microsoft Windows, um Informationen von einem in ein anderes Programm zu übertragen. DDE ermöglicht den Datentransfer beispielsweise von Tabellenkalkulationen, Textverarbeitungsprogrammen und TestPoint zu bewegen.

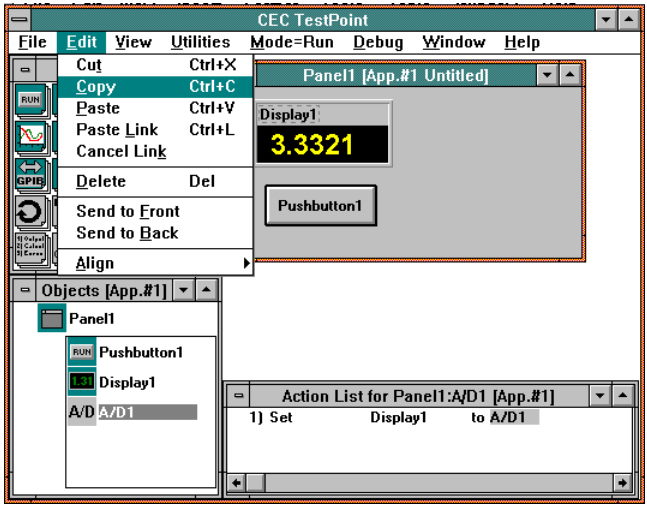
Mit DDE ist es möglich, automatisierte Lösungen zu einem Problem zu schaffen. Die dazu benötigten Funktionen müssen in dem verwendeten Programm nicht vorhanden sein. Es muß auch kein neues spezifisches Programm geschrieben werden.

Beispielsweise könnte eine TestPoint Anwendung eine automatisierte Test Prozedur für ein Produkt ausführen und dann die gewonnenen Informationen in ein Textverarbeitungsprogramm übertragen, welches die Formatierung und den Ausdruck eines speziell formatierten Meßprotokolls ausführt, das dem Produkt beigelegt wird.

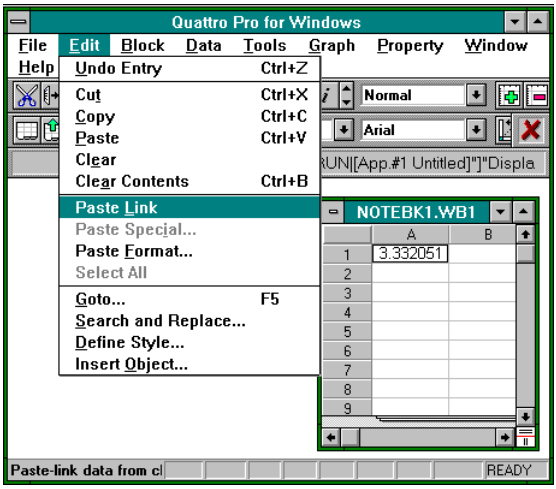
DDE Links: DDE ohne zu programmieren

Der einfachste Weg, um Informationen zwischen Programmen auszutauschen, ist das manuelle Erstellen eines DDE ‘link’, einer DDE Verbindung. Ein DDE ‘link’ frischt automatisch ein Programm auf, wenn sich die Daten im anderen ändern.

Dazu geht man zuerst in das Programm, welches die Quelle der Information darstellt. Zum Beispiel könnte eine Anwendung in TestPoint ein Display Objekt enthalten, das alle Sekunden erneuert wird und auch in einer Tabellenkalkulation angezeigt werden soll. In dem Quellprogramm (TestPoint) wählt man die Daten aus und verwendet von der Menü Zeile das Kommando ‘EDIT’ ‘Copy’:



Nun sucht man sich das Zielprogramm. In diesem Beispiel ist das Quattro Pro für Windows, ein Tabellenkalkulationsprogramm. Der Cursor wird an die gewünschte Stelle im Programm gestellt und aus der Menü Zeile ‘EDIT’ ‘Paste Link’ ausgewählt:



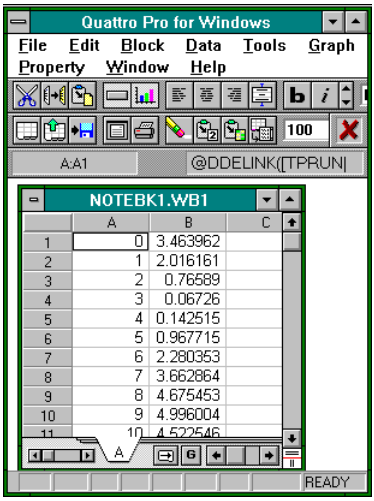
Einige Programme verwenden ‘EDIT’ ‘Paste Special’ anstatt „Paste Link“ und einige erlauben eine Wahl des Datenformats, wenn ein DDE ‘link’ hergestellt werden soll.

Das Zielprogramm erneuert nun automatisch die übergebenen Daten, wenn sich die Werte in dem Ausgangsprogramm ändern.

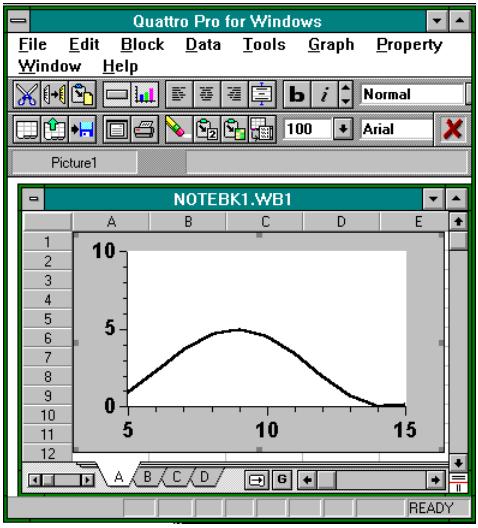
Um Link-Verbindungen in die andere Richtung zu erzeugen, kehrt man den Vorgang einfach um. TestPoint Objekte, die auf einem Panel erscheinen, können Daten von DDE ‘link’ sowohl empfangen als auch senden.

Verbinden von Daten aus einer Grafik

Das TestPoint Grafikobjekt kann über DDE eine Datenquelle für ein anderes Programm sein. Einfach auf den Graphen im Panel klicken, kopieren und in eine Tabellenkalulation einfügen, damit bekommt man:

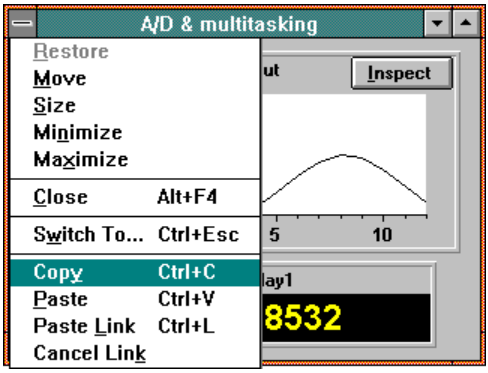


Ermöglicht die Tabellenkalkulation die Wahl der Einfügeformate, so kann ‘Bild’ anstatt ‘Text’ als Format gewählt werden und man erhält:



Verbinden von TestPoint Runtimes

Arbeitet eine TestPoint Anwendung außerhalb des TestPoint Editors (Runtime Version), so sind Link-Verbindungen durch kopieren und einfügen mit Hilfe des Systemmenüs des Fensters erzeugbar. Dazu klickt man auf die obere linke Ecke des Fensters:



Ebenso kann die Tastenkombination Strg-C oder Strg-L benutzt werden.

Verbinden zu TestPoint

TestPoint kann nicht nur als Quelle für eine Datenverbindung verwendet werden, sondern auch als Ziel. Verwenden Sie dazu das **Paste Link** Menükommando von **TestPoint**.

Abbrechen einer Verbindung

Wenn TestPoint der Empfänger eines DDE ‘links’ ist, möchte man möglicherweise das ständige Erneuern stoppen. Um dies zu tun, klickt man auf das Objekt im Panel und benutzt im TestPoint Editor das Menü ‘EDIT’ ‘Cancel Link’. In einer Runtime Version siehe Bild oben.

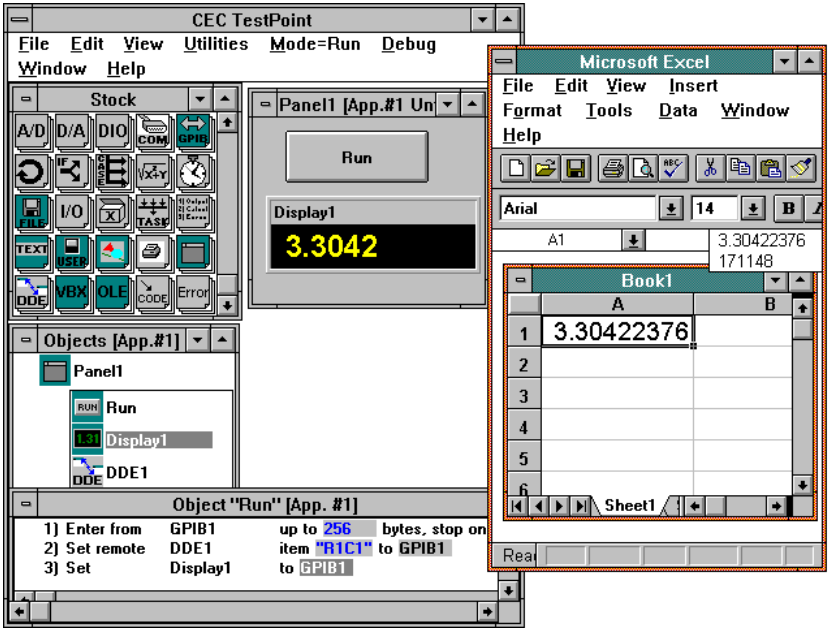
Das DDE Objekt: Automatisiertes DDE

Mit Kopieren und Einfügen wird die Link-Verbindung manuell erzeugt. Jeder, der eine geschriebene TestPoint Anwendung benützt, kann dies tun, aber es ist ein manueller Vorgang. Damit DDE automatisiert und ein eingebauter Bestandteil der Anwendung wird, steht das DDE Objekt zur Verfügung.

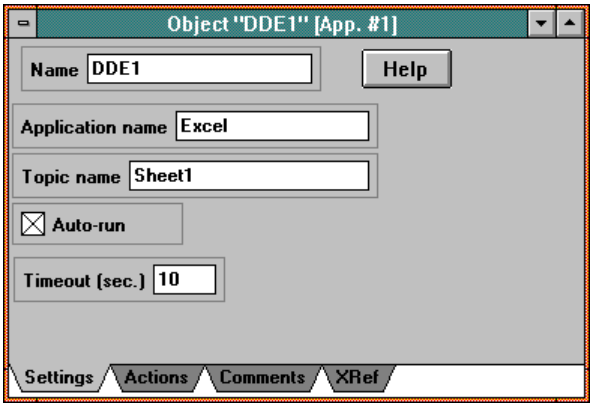
Der Gebrauch des DDE Objekts verlangt, daß über das andere Programm, mit dem gearbeitet werden soll, die folgenden Informationen bekannt sind: Der DDE ‘server name’ (normalerweise nur der Name der Anwendung, wie „Excel"), der DDE ‘topic name’ (normalerweise nur der aktuelle Dokument- oder Datei-Name in dieser Anwendung, wie "Tab1") und die Art, wie diese Anwendung DDE Zielangaben (item names) benutzt. In Excel zum Beispiel, sind Zellen gekennzeichnet durch Reihen und Spalten, wie etwa ‘R1C1’ (**In der deutschen Version von Excel heißt es allerdings ‘Z1S1’ für Zeile und Spalte).** Lotus 123 benutzt Zellenbenennungen wie ‘A5’.

Ein DDE Beispiel mit Microsoft Excel

Als ein Beispiel sei hier eine Anwendung angeführt, die von einem GPIB Gerät Werte liest und den empfangenen Zahlenwert in die Tabellenkalkulation Microsoft Excel einträgt:



Das DDE Objekt teilt TestPoint mit, daß es mit ‘Excel’, ‘Sheet1’ kommunizieren wird (**Tab1 in der deutschen Version von Excel**). Die Einstellung ‘Auto-Run’ startet Excel, wenn es noch nicht geladen ist:



Die nachfolgende Aktions Liste des Schalters liest ein GPIB Gerät aus und benutzt dann die Aktion ‘Set remote’ des DDE Objekts, um Werte in die Tabellenkalkulation zu übertragen:

Enter from	GPIB1	up to 256 bytes
2) Set remote	DDE1	item "R1C1" to GPIB1
3) Set	Display1	to GPIB1

Beachten Sie, daß die deutsche Excel Version anstelle „R1C1“ in der Action Liste „Z1S1“ erwartet.

Listen, Vektoren und DDE

Durch die Verwendung der Aktionen ‘Set remote’ oder ‘Get remote’ kann mehr als eine einzelne Zahl zwischen Programmen übertragen werden:

Sendet man Vektoren oder Listen von Vektoren, so sind die Daten wie beim Schreiben einer Datei durch das File Objekt formatiert. In den meisten Fällen ist diese Voreinstellung ausreichend:

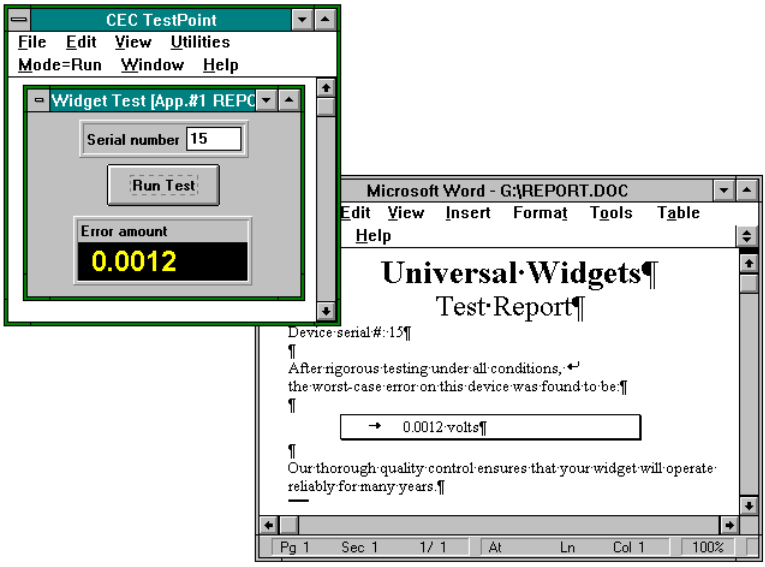
1)	Acquire A/D	A/D1	#samples=50, rate=1000 Hz,channel(s)=0,1
2)	Set remote	QPW	item "A1..B50" to A/D1

Diese Action Liste nimmt 50 Werte von zwei analogen Eingangskanälen auf, das bedeutet, die Daten sind in einer Liste von zwei Vektoren aufgeteilt. Anschließend sendet sie die Daten zu einem Quattro Pro Tabellenblatt. Der Ziel Name ist in diesem Fall im Tabellenblatt der Zellenbereich von zwei Spalten und 50 Reihen.

Muß das Format der Daten doch einmal geändert werden, so kann dies durch einen Doppelklick auf den Parameter (im oberen Beispiel A/D1 in Zeile 2) in der ‘Set Remote’ Aktion geschehen.

Den Empfang erledigt die ‘Get remote’ Aktion, für die das gleiche gilt. Andere als die voreingestellte Formatierung erfolgt über einen Doppelklick auf den Namen des DDE Objekts und der Auswahl des gewünschten Datentyps und Formats.

Ein DDE Beispiel: Protokoll Erstellung



Diese Anwendung arbeitet einen Test ab und schreibt die Ergebnisinformationen in ein Microsoft Word Dokument.

Die Einstellungen des DDE Objekts: Objekt Name=Report, Application name=WINWORD und Topic=G:\REPORT.DOC.

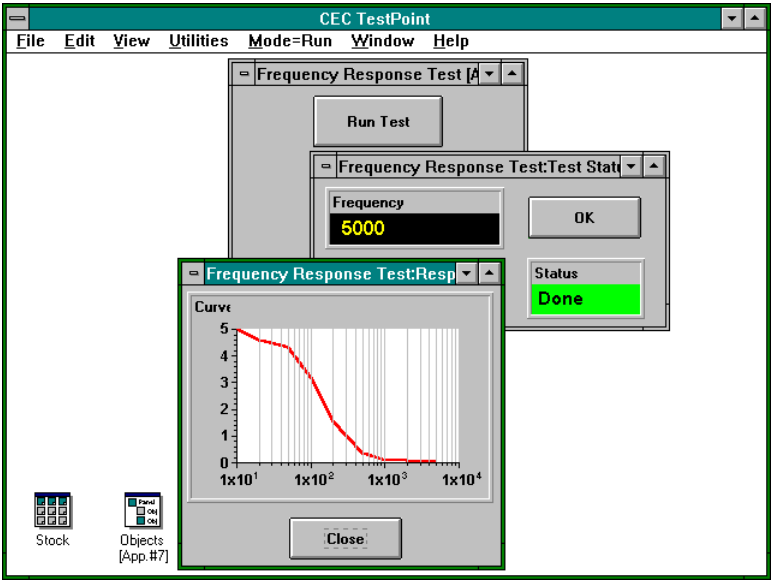
Im Word Dokument sind Textmarken, ‘bookmarks’ definiert, um die DDE-Datei zu plazieren. Word benutzt Textmarken als DDE ‘item names’. Zwei Textmarken wurden definiert: ‘Serial’ und ‘Result’.

Die Action-List Zeilen:

3)	Set remote	Report	item Serial to Serial number
4)	Set remote	Report	item Result to Error amount

Protokoll Erzeugung mit Quattro Pro

Diese TestPoint Anwendung gibt über einen programmierbaren GPIB Funktionsgenerator eine Serie von Frequenzen aus, liest die Spannung zurück und zeichnet die Ergebniskurve in ein Diagramm:



1)	Clear	Container1	
2)	Decade series	Frequency	from 10^1 to 10^3 in 1,2,5 steps
3)	Output to	Func. Gen.	from Frequency, term.=LF
4)	Enter from	Voltmeter	up to 256 bytes, stop on LF
5)	Append to	Container1	from Frequency, Voltmeter
6)	End	Frequency	
7)	Draw graph	Curve	with Container1

Um ein Protokoll zu erzeugen, erzeugt man zuerst ein Tabellenblatt in Quattro Pro für Windows. Man definiert darin einen Bereich von Zellen und benennt ihn als ‘Result’ : 9 Zeilen und 2 Spalten, in welche die Frequenz- und die Spannungswerte eingefügt werden.

Anschließend nimmt man ein DDE Objekt in die TestPoint Anwendung auf. Der Application Name ist ‘QPW’ (die *.EXE Datei für Quattro Pro), der ‘topic name’ ist der Name des Tabellenblatts, z. B. ‘FREQ. WB1’, und ‘auto-run’ wird eingeschaltet. (Freq. WB1 vorher einladen)

Dann noch diese zwei Zeilen in die Actions Liste hinzufügen:

```
8)  Set remote      Quattro     item "Results" to Container1
9)  Execute remote  Quattro     command "{Print.DoPrint}"
```

Das ist alles!

Das Tabellenkalkulations Programm druckt nun automatisch. Wollte man ein Diagramm in Quattro Pro erstellen, so würde dieses ebenfalls ausgedruckt. Man kann die Möglichkeiten von Quattro Pro ausschöpfen, um Schriftart, Schattierung, Farben und so weiter zu verändern, damit ein Ausdruck hoher Qualität erfolgt.

Der Name ‘Results’ ist einfach der vereinbarte Name eines Zellenbereichs in Quattro Pro (Alternativ wäre auch ‘A:A1..B9’ möglich). Das ‘Execute remote’ Kommando ist einfach ein Quattro Pro Makro Aufruf, um ein Tabellenblatt zu drucken.

Kommandos an andere Programme

DDE Kommandos

Die meisten populären Programme akzeptieren Kommandos über die DDE Verbindung. Mit TestPoint können Sie ein DDE Kommando mit der „**Execute remote**“ Aktion übertragen:

1) Execute remote Excel command "[RUN("Macro1",false)]"

Das Format für gültige DDE Kommandos hängt vom verwendeten Programm ab. Für Microsoft´s Excel, wie im obigen Beispiel, kann jedes gültige Makro in eckigen Klammern gesendet werden.

Tastenanschläge übertragen

Einige Programme besitzen keine DDE Kommando Sprache, andere haben Funktionen, die nur über Menüs erreichbar sind. Für diese Fälle werden von TestPoint aus simulierte Tastaturkommandos an diese Programme übertragen. Das DDE Objekt stellt die „**Send keys to**“ Aktion zur Verfügung:

1) Send keys to Excel keys="@DS"

Das Zeichen „@“ in der Action Liste bedeutet, daß die „Alt“-Taste, gefolgt von einem „D“ und „S“ übertragen wird. Für Excel bedeutet dies, daß das Menükommando DATEI/SPEICHERN ausgeführt wird.

Die Details für die „Send keys“ Aktion finden Sie im Referenzteil dieses Handbuchs unter dem DDE Objekt.

Kurzreferenz zu DDE mit bekannten Programmen

In allen Fällen, die folgen, sollte der 'Application name' und der 'topic name' den vollen Laufwerks- und Pfadnamen enthalten. Zum Beispiel: 'C:\EXCEL\EXCEL'

Excel

Application name:		EXCEL
Topic name:		Tabellendateiname
Item name:		RrowCcolumn (z.B. Z1S1)
	or:	RrCc:Rr2Cc2 (z.B. Z1S1:Z3S5)
Remote Kommando Format:		[Excel Macro] (z. B. "[COPY()]")

Beachten Sie: Für Excel 5.0 „Arbeitsmappen" verwenden Sie bitte neben dem Mappennamen den Dateinamen in eckigen Klammern:

```
c:\excel\[tab1.xls]Mappe1
```

Lotus 123

Application name:		123W
Topic name:		*Tabellendateiname*
Item name:		*Zellenname* (z.B. "A1")
	oder	*Benannter Bereich* (z.B. "Results")
Remote Kommando Format:		[run({*Lotus Macro*})] (z. B. "[run({select B3..C4})]")

Quattro Pro for Windows

Application name:		QPW
Topic name:		*Tabellendateinamen*
Item name:		*Zellenname* (z. B. "A1")
	oder	*Benannter Bereich* (z. B. "Results")
Remote Kommando Format:		{*Quattro pro Macro*} (z. B. "{Print.DoPrint}")

Word for Windows

Application name:	WINWORD
Topic name:	Dokumentenname
Item name:	Textmarke (z. B. "Result")

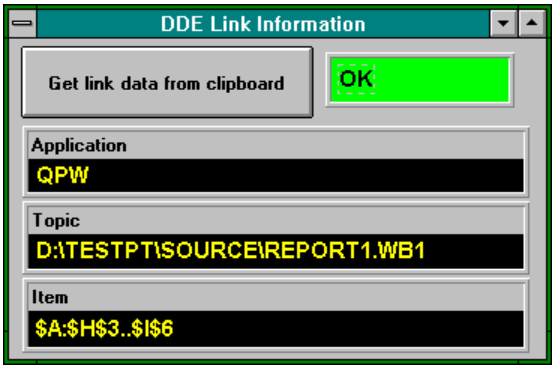
TestPoint

TestPoint Objekte, sogar solche, die sich auf dem Panel nicht zeigen (z. B. Math-Objekt), stellen Ihre Daten zur Übertragung per DDE zur Verfügung. Auch TestPoint Schalter, Pushbuttons, Data-Entry Objekte, usw. können verwendet werden.

Application name	TPRUN
Topic name:	.TST Dateiname
Item name:	Object Name

Andere Programme

TestPoint beinhaltet ein Hilfsprogramm namens **‘LINK.TST’.** Dieses kann in TestPoint geladen werden. Wählt man ‘EDIT’ ‘Copy’ im gewünschten Programm, schaltet TestPoint auf RUN-Mode, so ermittelt TestPoint die Link-Informationen des Link-Partners und bringt sie zur Anzeige.



DDE in Netzwerken

Arbeitet man mit Windows for Workgroups, so kann NetDDE verwendet werden, es läßt DDE Verbindungen über das Netzwerk zu.

Das Handbuch zu Windows for Workgroups enthält eine komplette Beschreibung, hier folgt nur ein kurzer Auszug von Befehlen, die benötigt werden, um eine Verbindung zwischen zwei Computern herzustellen:

- Am Ausgangscomputer kopieren der gewünschten Daten mit 'EDIT' 'Copy'.
- Nun wird die Zwischenablage geöffnet.
- Benutzen Sie das Menü, um Ihr clipbook zu öffnen oder zu erstellen.
- Einfügen der Daten in das clipbook. Zuweisung eines Namens. Markieren Sie es als „shared" im Netzwerk.
- Am Zielcomputer die Zwischenablage öffnen.
- Benutzen Sie das Menü, um auf das Clipbook des anderen Computers zuzugreifen.
- Kopieren Sie den shared Eintrag mit 'EDIT' 'Copy'.
- Zum Schluß Paste Link im Zielprogramm benutzen.

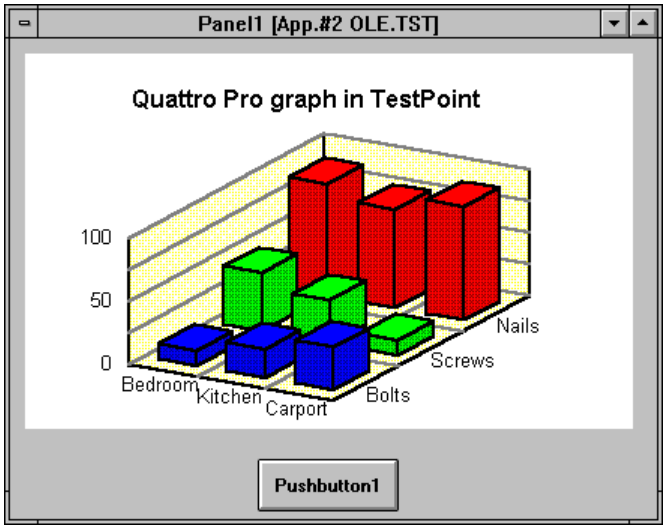
Dieser Vorgang ist identisch mit zwei aufeinanderfolgenden kopieren/einfügen Handlungen und erzeugt eine Verbindung zwischen Programmen, die auf verschiedenen Computer laufen.

Was ist OLE?

OLE (Objekct Linking und Embedding) ist eine Funktion von Windows und ebenfalls von TestPoint, die es erlaubt, Informationen von einem Programm verarbeiten zu lassen und in einem anderen anzuzeigen.

Das TestPoint OLE Objekt 

erlaubt es Ihnen, Daten eines anderen Programms in TestPoint anzuzeigen:



Das OLE Objekt kann mit eingebetteten (Embedded) oder verbundenen (Linked) Daten arbeiten.

Eingebettete OLE Daten

Eingebettete Daten werden in einer TestPoint Anwendung gespeichert (.TST Datei). Obwohl die Daten ursprünglich aus einem anderen Programm, wie z. B. einem Tabellenkalkulationsprogramm kommen, sind die Daten in TestPoint gespeichert.

Beispiel: Nehmen Sie ein neues OLE Objekt. Wählen Sie ein „Paintbrush Bild“ aus und klicken Sie den „New“ Schalter. Das Paintbrush Bild, das Sie erzeugen, wird in der .TST Datei mitgespeichert und Sie benötigen keine weitere Bitmap Datei (es gibt im Paintbrush auch kein DATEI/SPEICHERN Menükommando). Das ist der Vorteil von OLE Daten: Sie benötigen keine weitere Datei neben Ihrer TestPoint Anwendung.

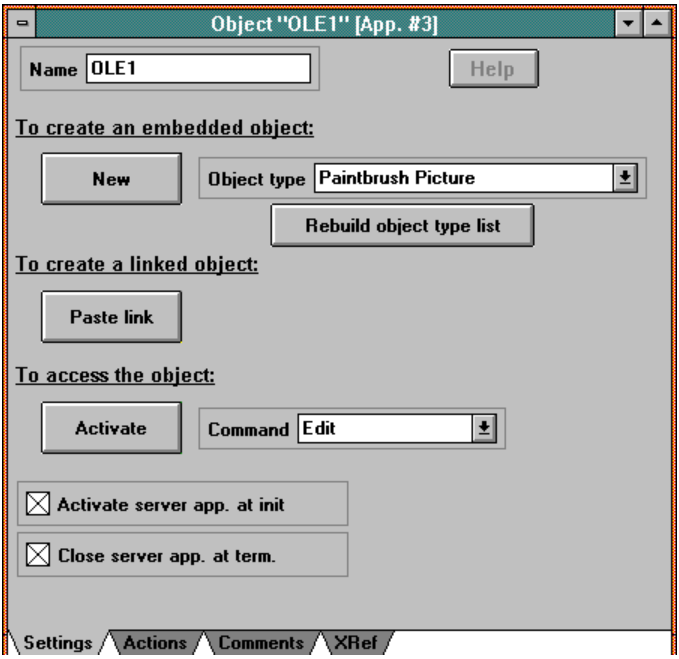
Der Nachteil dieser eingebetteten Daten ist, daß sie nicht automatisch aufgefrischt werden, wenn sich eine Änderung ergeben hat. Das ist der Grund, warum Sie keine Excel Grafik von den Daten des Grid Objektes im TestPoint erzeugen können.

Verbundene OLE Daten

Verbundene (Linked) Daten kommen aus einer separaten Datei. Sie können beispielsweise eine Excel Tabelle speichern und diese mit TestPoint verbinden. Um ein verbundenes OLE Objekt zu erstellen, verwenden Sie das Menükommando BEARBEITEN/KOPIEREN im Quellprogramm (z. B. Excel) und den „**Paste link**“ Schalter im Settings Fenster des OLE Objektes im TestPoint.

OLE: Darstellung von Daten anderer Programme

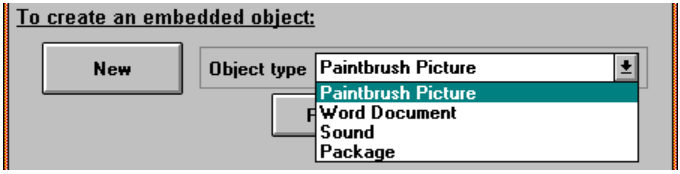
Wenn Sie ein OLE Objekt in Ihrer Anwendung verwenden, erhalten Sie folgendes Settings Fenster:



Als erstes müssen Sie sich entscheiden, ob Sie eingebettete (embedded) oder verbundene (linked) Daten verwenden wollen. Eingebettete Daten sind sinnvoll bei statischer Information. Diese muß manuell verändert werden (und macht Ihre .TST Datei größer). Verbundene Daten werden die häufigste Auswahl sein.

Erzeugung eingebetteter Daten:

Wählen Sie als erstes den Objekt Typ. Wenn dieser nicht in der vorgegebenen Liste vorhanden ist, verwenden Sie den „**Rebuild object type list**“ Schalter, durch den das gesamte System nach verfügbaren OLE Datenquellen durchsucht wird.



Klicken Sie dann den „**New**“ Schalter an. Dieser startet das passende Programm und läßt Sie die OLE Daten bearbeiten. Beispielsweise wird Paintbrush verwendet, um Bitmaps zu bearbeiten, der Waveform Recorder für Sounds, usw.

Anstelle des Menükommandos DATEI/SPEICHERN wird das Programm das Kommando DATEI/AKTUALISIEREN (Update) anbieten, das die OLE Daten im TestPoint aktualisiert.

Einige Daten, wie z. B. ein Paintbrush Bild, werden direkt dargestellt. Andere, wie Sounds, werden als Symbol dargestellt:

Bearbeiten eingebetteter Daten:

Wählen Sie die „Command“ Einstellung „Bearbeiten“ im Settings Fenster des OLE Objektes und klicken Sie den „Activate“ Schalter.

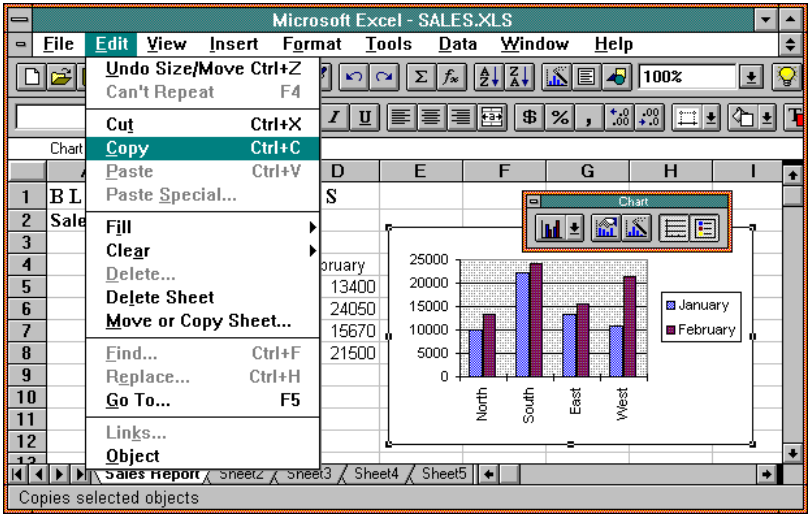
Die meisten OLE Daten werden eine Bearbeitung zulassen. Einige werden auch andere Befehle wie z. B. „Wiedergabe“ oder ähnliche Befehle haben.

Verbundene Daten zu erzeugen:

Erzeugen Sie als erstes die Daten im entsprechenden Programm (wie Excel, 123, Word, ...).

Speichern Sie die Daten im Quellprogramm.

Wählen Sie das Menükommando BEARBEITEN/KOPIEREN:



Verwenden Sie anschließend den „Paste link“ Schalter im Settings Fenster des OLE Objektes von TestPoint:

2-Wege Verbindungen: Bilder aktualisieren

Eine der nützlichsten Anwendungen des OLE ist die Erzeugung von 2-Wege Verbindungen. Das bedeutet die Anzeige eines Bildes, das von einem anderen Programm erzeugt wurde. Beispielsweise kann Excel verwendet werden, um Daten, die von TestPoint aus übertragen werden, darzustellen. Immer wenn TestPoint neue Daten zur Verfügung stellt, wird die Excel Darstellung aktualisiert. Die Darstellung, die im Excel gewählt wurde, kann nun wieder für die Darstellung in einem TestPoint Panel verwendet werden und wird automatisch auf den neuesten Stand gebracht.

Dies erfordert ein laufendes OLE Programm (z. B. Excel), um den Prozeß zu automatisieren. Ebenfalls erforderlich sind Daten, die über eine „Paste Link“ Funktion vom TestPoint aus übertragen werden.

Hier ist eine Schritt-für-Schritt Anweisung, wie man eine solche Verbindung aufbaut (unter Verwendung von EXCEL als Beispiel):

1. Öffnen Sie TestPoint und Excel.
2. Stellen Sie sicher, daß beide Dateien, die TestPoint- und die Excel- Datei, gespeichert sind (also keines „Unbenannt“ ist), weil die Verbindung einen Namen erfordert.
3. Erzeugen Sie ein TestPoint Objekt, das die Daten beinhalten soll. Dies kann ein Math Objekt, ein File Objekt usw. sein. Nehmen Sie z. B. ein Container Objekt und benennen es „OLE-QUELLE“. Immer, wenn Sie die OLE Daten nun aktualisieren wollen, erzeugen Sie eine Action Zeile, die Daten in diesem Container speichert.
4. Selektieren Sie das Quelldaten Objekt (OLE-QUELLE) dadurch, daß Sie mit der Maus im Objekt Fenster darauf klicken.
5. Benutzen Sie das EDIT/COPY Menükommando, um es zu

kopieren.

6. Gehen Sie zu Excel und markieren Sie eine Zelle. Verwenden Sie das BEARBEITEN/INHALTE EINFÜGEN Menükommando und wählen Sie dort „Verknüpfen“. Dies erzeugt die Verbindung zu TestPoint.
7. Erzeugen Sie nun die gewünschte OLE Information in Excel. Verwenden Sie beispielsweise eine Grafik, in der Sie die gerade eingelesenen Daten darstellen.
8. Markieren Sie die OLE Information in Excel (Klicken Sie auf die Grafik) und verwenden Sie das Menükommando BEARBEITEN/ KOPIEREN.
9. Schalten Sie zurück zu TestPoint und öffnen Sie das Settings Fenster des OLE Objektes. Klicken Sie den „**Paste link**“ Schalter an.
10. Stellen Sie sicher, daß **beide** Dateien gespeichert sind.

Das war´s! Starten Sie die Anwendung. Immer wenn sich die Daten im TestPoint ändern, wird das OLE Bild aktualisiert.

Automatischer Start des OLE Quellprogramms

Wie bereits oben erwähnt, muß das OLE Quellprogramm bereits laufen, wenn das automatische Aktualisieren funktionieren soll. Das OLE Objekt stellt eine Möglichkeit zur Verfügung, das OLE Quellprogramm (oder Server) automatisch zu starten, wenn das TestPoint Programm initialisiert wird. TestPoint kann die Anwendung ebenfalls schließen, wenn die TestPoint Anwendung beendet wird.

Schnell Referenz zu OLE mit populären Programmen

Microsoft Excel

Komplette OLE Unterstützung.

Microsoft Word 2.0

Unterstützt OLE - zeigt sich aber nur als Symbol, nicht als Text.

Microsoft Draw (gehört zu Word und anderen Programmen)

Unterstützt OLE. 2-Wege Verbindungen werden nicht unterstützt.

Lotus 123

Unterstützt OLE nur für ganze Tabellen, nicht für Grafiken.

Quattro Pro

Baut bei 2-Wege Verbindungen die Verbindung nicht automatisch auf. Benötigt ein Tools/Links/Refresh Menükommando (Makro), nachdem Quattro gestartet ist.

Microsoft PowerPoint

Unterstützt OLE, einschließlich „Slide Show“ Kommando, um eine Präsentation abzuspielen. Slide Show läuft nur bei verbundenen Dokumenten, nicht bei eingebetteten.

Lotus Freelance

Unterstützt OLE

Sound Recorder

Unterstützt OLE - zeigt sich aber nur als Symbol. Beinhaltet ein „Abspielen“ Kommando. 2-Wege Verbindungen werden nicht unterstützt.

Media Player

Unterstützt OLE. Für Videos ist es möglich, ein Bild zu zeigen oder auch, den Clip per „Abspielen“ Kommando zu zeigen. 2-Wege Verbindungen werden nicht unterstützt.

Kapitel Rückblick:

- **DDE:** Dynamic Data Exchange, der dynamische Datenaustausch, erlaubt, andere Windows Programme innerhalb von TestPoint Anwendungen zu gebrauchen. Daten können zu solchen Programmen gesendet und zurückgelesen werden.
- **OLE:** Object Linking und Embedding erlaubt es, Daten und Bilder aus anderen Anwendungen direkt in ein TestPoint Panel zu übernehmen. Dies beinhaltet Grafiken, Klänge, Videos oder andere Informationen.
- **Paste link:** DDE-Verbindungen können manuell erstellt werden, durch Kopieren von Daten von einem Programm und Einfügen dieser durch den Gebrauch von 'Paste Link' in einem anderen Programm. Diese Verbindung kann von TestPoint zu einem anderen Programm oder von einem anderen Programm zu TestPoint verlaufen.
- **Topics:** Alle Programme, die DDE unterstützen, stellen 'topic' Namen für die DDE-Verbindung zur Verfügung. Der 'topic' Name ist normalerweise der Name des zu editierenden Dokuments.
- **Items**: Ebenso stellen die DDE unterstützenden Programme 'item' Namen bereit, um die ansprechbaren Informationen zu kennzeichnen. Diese Bezeichner variieren von einem Programm zum anderen. Excel benutzt Reihen und Spalten Numerierung (z. +B. Z1S1), Word Textmarkennamen.

Kapitel 20. Fehlerbehandlung

Was in diesem Kapitel behandelt wird:

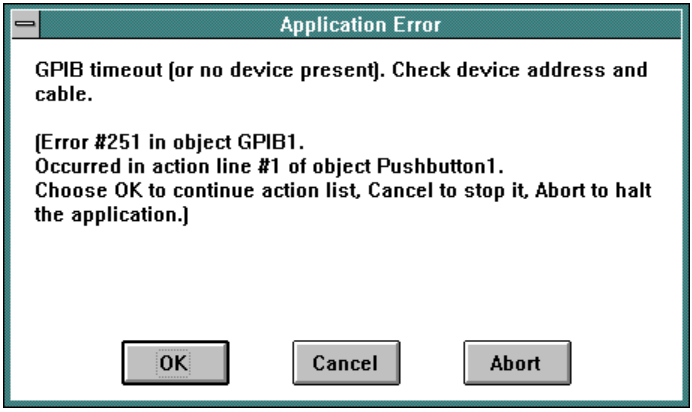
- TestPoint Fehler und eingebaute Fehlermeldungen.
- Wie können eigene Fehlerbehandlungen implementiert werden.
- Beispiele für gebräuchliche Fehlerfälle.
- Definition eigener Fehlermeldungen.

Wie TestPoint Fehler darstellt

Fehler kommen in TestPoint Anwendungen vor, wenn ein Hardware Schnittstellen Problem festgestellt wurde, ein falscher Parameterwert in eine Aktion eingegeben wurde, eine benötigte Datei nicht gefunden wurde, usw.

Jedem Fehlertyp ist ein Fehlercode zugeordnet. Diese Fehlercodes sind in Gruppen zusammengehöriger Fehler geordnet. Zum Beispiel, GPIB Fehler liegen alle in dem Bereich 250 bis 259. Fehlernummern sind in der Datei ERRORNUM.DOC im TestPoint Verzeichnis aufgelistet.

Wenn ein Fehler vorkommt, zeigt TestPoint ein Fehlermeldungs-Fenster, in dem die Fehlernummer, der Objektname, die Nummer der Befehlszeile und die Auswahl 'OK', Abbrechen (Cancel) oder Programmabbruch (Abort) aufgeführt sind.



Bei Auswahl von 'OK' wird mit der Action Liste, die den Fehler enthält, weitergearbeitet, als wäre kein Fehler vorgekommen. Natürlich werden in diesem Fall eventuell weitere Aktionen aufgrund des vorherigen Fehlers falsch ausgeführt. Bei Auswahl von Abbrechen wird diese Action list abgebrochen. TestPoint wartet danach auf weitere neue Ereignisse von anderen Action Listen.

Das Error Handler Object

Wenn eine andere Fehlerbehandlung als TestPoints Voreinstellung gewünscht wird, können Sie hierfür das Error Handler Objekt verwenden. Das Objekt kann auf jeden beliebigen Fehler oder Bereich von Fehlern reagieren und eine Aktion ausführen, die diesen speziellen Fehler behandelt.

Jedes 'Error Handler' Objekt hat Einstellungen für die minimale und maximale Fehlernummer, die extra behandelt werden soll. Sie können eine beliebige Anzahl von 'Error Handler' Objekten hinzufügen, wobei jedes dieser Objekte für einen anderen Fehlertyp eingestellt werden kann.

Tritt ein Fehler auf, und es existiert ein 'Error Handler' Objekt für diesen Fehler, erscheint nicht die von TestPoint voreingestellte Fehlermeldung, sondern die Action Liste dieses 'Error Handler' Objekts wird ausgeführt. Diese Action Liste kann jede beliebige Aktion enthalten und so die Fehlerbedingung anzeigen, abfangen, usw.

Am Ende der Action Liste des 'Error Handlers' sollte eine der folgenden Aktionen hinzugefügt werden, indem das 'Error Handler' Objekt in die Action Liste gezogen wird. (Diese Aktionen treten am Ende der Action Liste des 'Error Handler' Objekts in Kraft, sogar dann, wenn diese schon früher in der Action Liste ausgeführt wurden).

Continue after error

4) Continue after Error1 with data=1

Die 'Continue after' Aktion bringt TestPoint dazu, nach der Original-Aktionszeile, in der der Fehler aufgetreten ist, weiterzumachen. Das Objekt, das den Fehler verursacht hat, hat anschließend den Wert des "data=" Parameters von dieser Befehlszeile. Wenn dieser Parameter

leer gelassen wurde, werden die Daten des Objekts nicht verändert. Es wird keine Meldung angezeigt.

Stop after error

4) Stop after Error1

Die ‘Stop after’ Aktion bringt TestPoint dazu, die Action Liste, in der der Fehler aufgetreten ist, abzubrechen, so, als wäre eine normale Fehlermeldung erschienen und Sie ‘Abbrechen’ angeklickt hätten. Es wird keine Meldung ausgegeben.

Retry after error

4) Retry after Error1

Die ‘Retry after’ Aktion bringt TestPoint dazu, die Original Befehlszeile, in der der Fehler aufgetreten ist, noch einmal auszuführen. Diese Aktion macht dann Sinn, wenn in der Action Liste des ‘Error Handler’ Objekts bereits Schritte zur Beseitigung des Fehlers unternommen wurden, die Originalaktion aber nochmals ausgeführt werden soll. Es wird keine Meldung angezeigt.

Error message

4) Error message Error1 message="Uh Oh", button="retry"

Die ‘Error message’ Aktion zeigt eine Fehlermeldung an, so wie es TestPoint auch standardmäßig schon durchführt, jedoch können Sie Ihre eigene Fehlermeldung hierbei ausgeben. Die Schalter in dem Meldungsfenster können zusätzlich einen ‘Retry’ oder ‘Continue’ Schalter zum standardmäßigen ‘Cancel’ Schalter enthalten

Beispiel: Handhabung von GPIB Timeouts

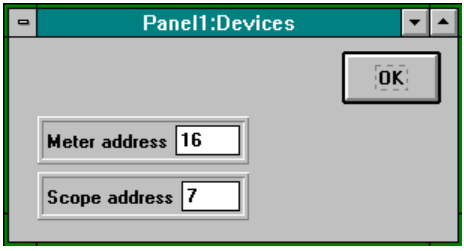
Der Timeout Fehler kann bei externen Geräten aufgrund verschiedener Tatsachen vorkommen. Die häufigsten Fehler sind nicht angeschaltete Geräte oder keine Kabelverbindungen. Eine Meldung für den Anwender, was er überprüfen soll und danach weitermachen kann, ist hierbei sehr sinnvoll.

Das einzige, was hierbei benötigt wird, ist ein ‘Error Handler’ Objekt genannt ‘Timeouts’, eingestellt, den Fehler mit der Nummer 251 abzufangen.

Seine Action Liste würde so aussehen:

1)	Error message	Timeouts	message="GPIB timeout. Check the cables and power switches on the devices." button="retry".

Falls gewünscht, kann der String auch noch eine Meldung beinhalten, die Geräteadresse zu überprüfen. Ein Gerät mit der falschen Adresse anzusprechen würde ebenfalls einen Timeout verursachen. Wenn die GPIB Geräteadresseingabe in einem Unterfenster genannt ‘Devices’ stattfinden soll, könnte es so aussehen :



Somit könnte folgende Action List des Error Handlers benutzt werden:

1) Error message	Timeouts	message="GPIB timeout. Check cables, power switches, and device addresses in the next screen displayed." button="retry".
2) Show & Wait	Devices	

Das ‘Devices’ Fenster enthält einen Schalter genannt ‘OK’, dessen Action Liste den Befehl ‘Hide Devices’ ausführt und damit das Einstellungsfenster wieder unsichtbar schaltet.

Wenn der Anwender „Retry“ wählt und andere Adressen einstellt, wird die Anwendung die GPIB Aktion wiederholen.

Beispiel: Die Unterstützung von zwei Tabellenkalkulationen über DDE

In diesem Beispiel wird ein DDE Objekt verwendet, um Testergebnisse in eine Tabellenkalkulation zu transferieren. Es wäre wünschenswert, daß TestPoint mit beiden Anwendungen, Excel und Quattro Pro, welche verschiedene DDE Namen haben, funktionieren würde, ohne den Anwender zu fragen, welches zur Verfügung steht.

Das DDE Objekt erzeugt einen Fehler, wenn es keine Verbindung zu der spezifizierten Tabellenkalkulation herstellen kann. Wir stellen das DDE Objekt einfach auf Quattro Pro ein und benutzen dies.

Das Error Handler Objekt stellen wir ein, DDE Fehler abzufangen und nachstehende Action Liste auszuführen:

1)	If/Then/Else	AlreadyExcel	with x=DDE(Application name)
2)	Error message	DDE Error	message="Can't open spreadsheet button="retry"
3)	Else if not	AlreadyExcel	
4)	Set	DDE(Application name)	to "Excel"
5)	Set	DDE(Topic name)	to "Sheet1"
6)	Retry after	DDE Error	
7)	End if	AlreadyExcel	

Das ‘Conditional’ Objekt ‘Already Excel’ prüft, ob diese Action Liste schon einmal ausgeführt wurde, indem es das Setting ‘Application name’ des DDE Objekts abfragt. In diesem Fall gibt es keinen weiteren Grund mehr, es weiter zu versuchen, im anderen Fall wird das DDE Objekt geändert, um die Verbindung mit Excel herzustellen. (Wie bereits beschrieben, lassen sich die Einstellungen der Objekte editieren und in Action Listen einfügen).

Eigene Fehlermeldungen

Wenn Sie Bedingungen haben, die in Ihrer Anwendung auftreten können, können Sie Fehlerbedingungen dafür definieren. Verwenden Sie dann die „Cause Error“ Aktion, um das Eintreten eines solchen Fehlers anzuzeigen:

```
1) Cause Error          Error-Handler1    code=20000
```

Benutzerdefinierte Fehlercodes können im Bereich 20000 bis 29999 liegen.

Dadurch sind Sie in der Lage eine Behandlung Ihrer eigenen Fehler zu erzeugen und diese anzeigen zu lassen, wann immer diese eintreten.

Chapter Review:

- Fehlermeldungen: TestPoint zeigt Fehlermeldungen, wenn ein Fehler vorkommt, mit näheren Details zum Fehlertyp, wo dieser passierte und eine Auswahl von Aktionen, um fortzufahren oder abzubrechen.
- Fehlernummern: Jede Fehlerbedingung, die von TestPoint erkannt wird, hat eine Fehlernummer. Diese Nummer ist gruppiert in zusammenhängende Fehlertypen.
- Benutzerdefinierte Fehlerbehandlung: Das Error Handler Objekt erlaubt benutzerspezifische Fehlerbehandlung für jeden beliebigen Fehler. Sobald ein Fehler auftritt, bekommt das Error Handler Objekt dieses Ereignis mitgeteilt und kann die entsprechende Aktion auslösen.

Kapitel 21. Multitasking

Was in diesem Kapitel behandelt wird:

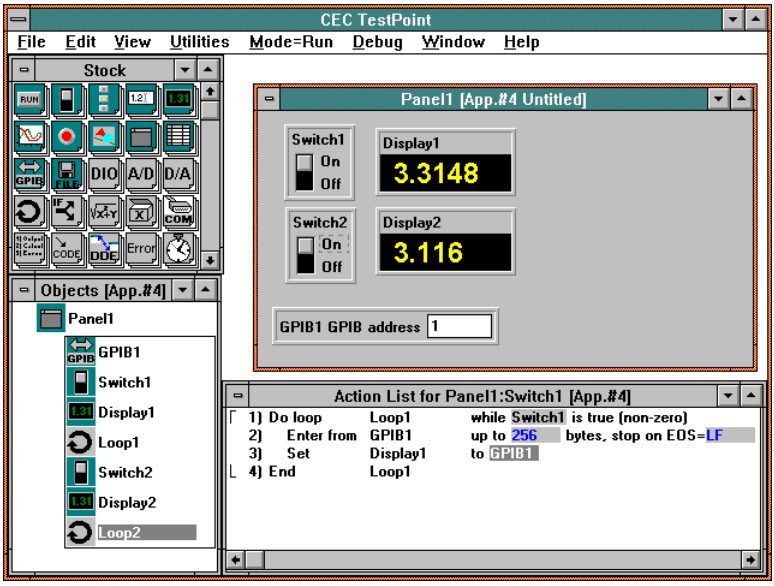
- Was Multitasking ist.
- Wie man Hintergrundbearbeitung benutzt.
- Wie man Multitasking in TestPoint kontrolliert.

Multitasking Definition

Multitasking bezieht sich auf Computer, die simulieren, viele Dinge gleichzeitig zu tun. Während ein Computer aktuell nur eine Instruktion zur gleichen Zeit bearbeiten kann, kann es doch so awussehen, als würde er mehrere Aufgaben gleichzeitig erledigen, indem er sehr schnell zwischen den einzelnen Aufgaben hin und her schaltet.

TestPoint erlaubt das Umschalten zwischen den einzelnen Aufgaben, wenn keine Ereignisse anliegen oder das Ende einer Schleife in einer Action Liste erreicht ist. Das bedeutet, daß eine einzelne TestPoint Anwendung mehrere Action Listen mit Schleifen besitzen darf, die zur gleichen Zeit ausgeführt werden.

Probieren Sie dieses Beispiel:



Beide Schalter haben ähnliche Action Listen mit Schleifen, die laufen, sobald der Schalter auf ‘On’ ist.

1)	Do loop	Loop1	while Switch1 is true
2)	Enter from	GPIB1	up to 256 bytes
3)	Set	Display1	to GPIB1
4)	loop	Loop1	

(Die Action Liste von Switch2 ist identisch, nutzt nur Loop2 und Display2).

Wenn Sie einen der Schalter auf ‘On’ stellen, beginnt am Display die Ausgabe. Schalten Sie den zweiten Schalter ein, so beginnt auch dieses Display mit der Anzeige. Beide Schleifen scheinen zur selben Zeit zu laufen. Aktuell läuft jedoch nur eine Schleife, gibt die Kontrolle über den Computer wieder frei, so daß die andere Schleife auch ausgeführt werden kann.

Hintergrund Action Listen

Ein Weg, Multitasking in TestPoint zu realisieren, ist, eine oder mehrere Hintergrund Action Listen zu erstellen, welche immer laufen, wenn Ihre Anwendung gestartet wurde. Diese können den Status eines Gerätes überprüfen oder andere Operationen durchführen, während es dem Anwender immer noch erlaubt ist, Schalter im Arbeitsfenster zu betätigen und damit Aktionen zu starten.

Um Hintergrund Action Listen zu erstellen, nutzen Sie eine Schleife, wie in dem vorherigen Beispiel gesehen, mit einem Schalter.

1) Do loop Loop1 while Switch1 is true
2) ...other actions here as desired...
3) End loop Loop1

Benutzen Sie die Einstellungen des Schalters, um die Action Liste bereits bei der Initialisierung zu starten (Execute at initialize). Sie können den Schalter auch unsichtbar (Invisible) machen.

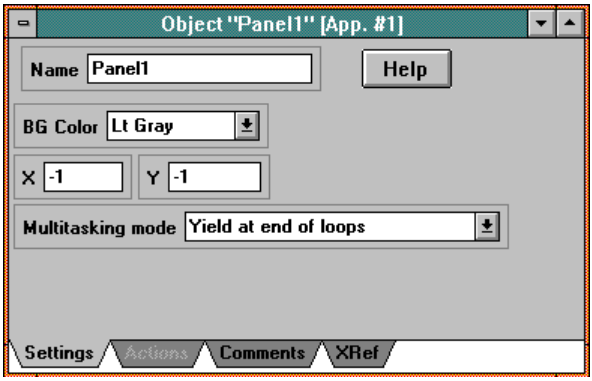
Multitasking Modi

Normalerweise, wie vorher beschrieben, führt TestPoint Action Listen aus, bis diese beendet sind oder das Ende einer Schleife erreicht wird und erlaubt dann anderen Prozessen, die Kontrolle zu übernehmen.

Sie können andere Modis des Multitasking auswählen:

- Kein Multitasking (ausgeschaltet)
- Freigabe an andere ‘Action Listen’ am Ende von Schleifen
- Freigabe an andere ‘Action Listen’ nach jeder Action Zeile

Diese Einstellung erfolgt in den ’Settings’ des Hauptarbeitsfensters.



Wenn Sie Probleme haben, die vielleicht im Multitasking begründet sind, wie z. B. zeitweilige Aussetzfehler, versuchen Sie, das Multitasking auszuschalten, um zu sehen, ob das Problem damit beseitigt ist.

Der „Yield on every Line“ Modus sollte möglichst nicht verwendet werden. Bei dessen Benutzung kann es zu Problemen kommen, da dann verschiedene Action Listen, die u. U. die gleichen Objekte verwenden, in nicht vorgesehener Reihenfolge aufgerufen werden können.

Starten und Stoppen von Tasks

Neue Tasks werden von TestPoint gestartet, wenn Ereignisse auftreten. Zum Beispiel: Wenn der Benutzer mit der Maus auf ein Pushbutton Objekt klickt, wird eine Task gestartet, die die Action List dieses Pushbuttons ausführt. Oder: Wenn A/D Meßwerte aufgenommen worden sind, startet die A/D Action List eine neue Task.

Falls eine Action List eine andere zur Ausführung bringt, z. B. indem eine Set Action eines Data-Entry Objekts ausgeführt wird, wird **keine** neue Task gestartet. Die neue Action List wird ähnlich eines Unterprogramms ausgeführt und ist somit Teil der aufrufenden Task.

Start

Um neue Tasks innerhalb einer Action List zu erzeugen, müssen Sie ein **Task** Objekt benutzen. Eine der möglichen Actions dieses Objekts ist "Start", damit wird die Action Liste des Task Objekts als neue Task aufgerufen, während die originale Action List weiterhin eigenständig läuft.

Stop

Wenn Sie ein Task Objekt benutzen, eine Task zu starten, haben Sie ebenfalls die Möglichkeit, diese Task mit der "Stop" Aktion zu stoppen, auch wenn noch nicht das Ende seiner Action List erreicht wurde oder es eine Endlosschleife ist.

Kritische Bereiche

Obwohl Multitasking in der Regel nützlich sein kann, können Sie damit auch leicht in Schwierigkeiten kommen. Normalerweise kann das nur passieren, wenn Sie den Multitasking Modus auf "yield after every line" setzen. Das ist auch der Grund, warum dieser Modus für die meisten Anwendungen nicht zu empfehlen ist.

Der Grund für die Probleme liegt meistens darin begründet, daß die selben Daten (Objekte) von zwei Action Listen gleichzeitig verwendet werden. Wenn beide Action Listen die Daten lesen, eine Berechnung durchführen und die Daten auf den neuesten Stand bringen, können Sie falsche Ergebnisse erhalten.

Zum Beispiel, sehen Sie sich diese beiden Action Listen an:

Pushbutton1:

1) Calculate	Next value	with x=Display1
2) Delay	Timer1	for 3 seconds
3) Set	Display1	to Next value

Pushbutton2:

1) Calculate	Next value	with x=Display1
2) Delay	Timer1	for 3 seconds
3) Set	Display1	to Next value

wobei "Next value" berechnet "x+1".

Wenn Sie den Schalter 2 drücken, wird das Display auf 3 gesetzt. Betätigen Sie Schalter 1 , wird ein neuer Wert berechnet, welcher 3 Sekunden verzögert in der Anzeige erscheint. Der Anzeigewert ist jetzt 4. Führen wir ein Experiment durch. Schalten Sie Schalter aktiv und drücken Schalter 2 innerhalb der drei Sekunden. Die Berechnung bezieht sich auf den Anzeigewert 4, auch wenn Schalter 2 zwischenzeitlich eine 3 erscheinen ließ, so daß das Display letztlich eine 5 anzeigt.

Das Endergebnis hängt somit vom exakten Zeitverhalten der Ausführung von Aktionen ab. Dieses Verhalten kann in einigen Fällen zu ungewollten Ergebnissen führen.

Es ist nicht spezifisch für TestPoint, sondern kann immer dann passieren, wenn Multitasking eingeschaltet ist.

TestPoint beinhaltet allerdings eine Lösung für derartige Fälle (critical regions). 'Critical regions' ist ein Standard-Programmierwerkzeug, um Multitasking zu kontrollieren. Es ist eine Sperrvorrichtung, damit nur ein Prozeß zur gleichen Zeit zugreifen kann. Dies erlaubt Ihnen, einen sequentiellen Zugriff auf gemeinsame Daten zu erzwingen.

```
Pushbutton1:
1) Enter critical region Task1
2) Calculate        Next value      with x=Display1
3) Delay            Timer1          for 3 seconds
4) Set              Display1        to Next value
5) Exit critical region Task1

Pushbutton2:
1) Enter critical region Task1
2) Calculate        Next value      with x=Display1
3) Delay            Timer1          for 3 seconds
4) Set              Display1        to Next value
5) Exit critical region Task1
```

Ändern Sie das vorangegangene Programmbeispiel in der oben gezeigten Art und Weise und starten Sie erneut. Das Endergebnis beträgt immer 3, da der Zugriff für Schalter 2 solange gesperrt ist, bis Schalter 1 die 'critical region' verlassen hat. 'Critical regions' werden von dem Task Objekt zur Verfügung gestellt.

Erweiterte Programmierung

Kapitel 22.
Benutzerdefinierte Objekte

Was in diesem Kapitel behandelt wird:

- Erzeugung benutzerdefinierter Objekte.
- Wie benutzerdefinierte Objekte die Möglichkeiten von TestPoint erweitern.
- Weitergabe von benutzerdefinierten Objekten als Bestandteil von TestPoint in Form von Libraries.

Benutzerdefinierte Objekte sind ein Weg, die eingebauten Möglichkeiten von TestPoint zu erweitern. Ein benutzerdefiniertes Objekt besitzt alle Eigenschaften eines mitgelieferten Objektes. Der innere Aufbau besteht aus TestPoint Objekten und Action Listen.

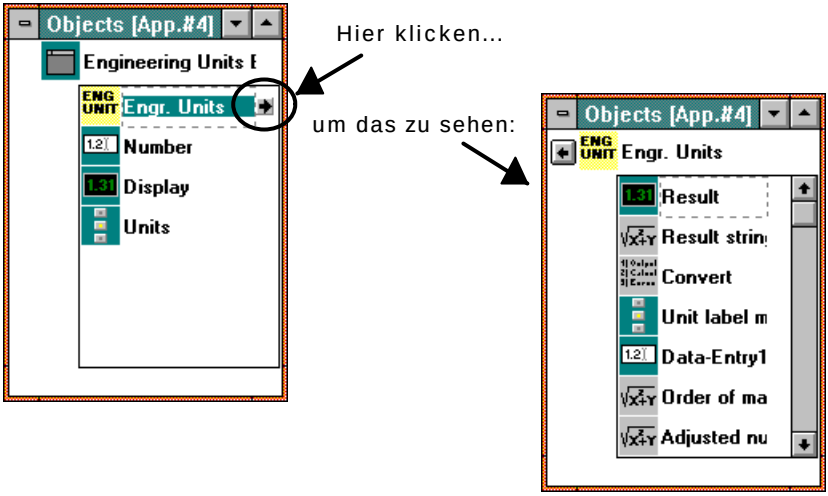
Benutzerdefinierte Objekte können beispielsweise als wiederverwendbare Elemente, wie eine Library für Geräte, spezifische Kommandos, oder als Objekt, welches ein spezielles Protokoll erstellt, ausgelegt werden.

Benutzerdefinierte Objekte sind vor allem in Einzelanwendungen nützlich, wenn diese sehr groß werden. Das Programm nutzt diese benutzerdefinierten Objekte ähnlich einem Unterprogramm. Die Übersichtlichkeit der Anwendung wird dadurch erhöht, daß die Details der Unterfunktionen innerhalb der benutzerdefinierten Objekte versteckt sind.

Damit Sie sehen, was mit benutzerdefinierten Objekten gemacht werden kann, sind im Lieferumfang von TestPoint einige Beispiele enthalten: FKEY.TST (Aktionen auf Funktionstasten), ENGR.TST (Umwandeln auf physikalische Einheiten), PID.TST (PID-Regler Beispiel).

Wie benutzerdefinierte Objekte arbeiten

Ein benutzerdefiniertes Objekt im TestPoint enthält andere Objekte, wie es ebenfalls bei Panels der Fall ist:

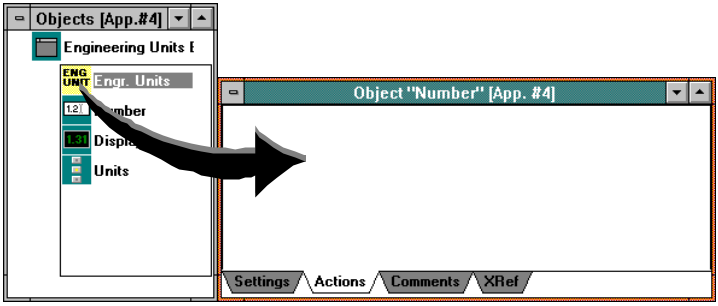


(Hinweis: Benutzerdefinierte Objekte können **gesperrt** sein. In diesem Fall gibt es keinen Pfeil, auf den man klicken kann, um den Inhalt zu sehen.)

Außerdem besitzt ein benutzerdefiniertes Objekt Aktionen, Daten, Einstellungen und eine Action Liste, die komplett vom Programmierer definiert werden.

+ **Ein benutzerdefiniertes Objekt ist definiert durch die Beziehungen der Objekte, die es beinhaltet.**

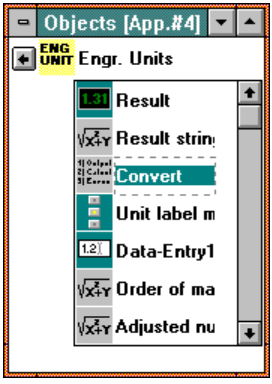
Wenn Sie das Objekt „Engr. Units“ aus der Anwendung ENGR.TST in einer Action Liste verwenden:



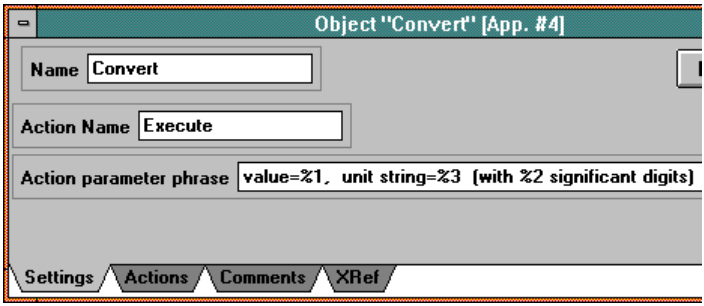
erhalten Sie diese Action Zeile:

1) Convert to Engr. Units value=____, unit string=____, (with ____ significant digits)

Woher kommt diese Zeile? Sie wird durch ein Action Objekt definiert, das „Convert“ heißt und sich in dem benutzerdefinierten Objekt befindet:

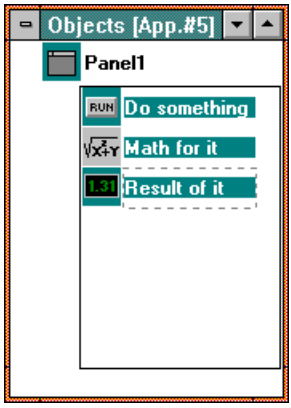


Dessen Einstellungen sind:



Erstellen von benutzerdefinierten Objekten

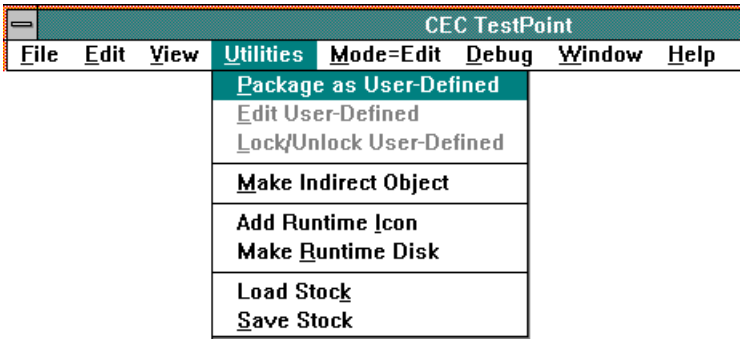
Packen (Packaging) Sie beginnen mit der Erstellung durch das Selektieren (Markieren) der Objekte, die verwendet werden sollen:



Sie können mit einem Schalter oder einem Action Objekt beginnen, das eine der Aktionen des neuen Objektes liefert. Es kann aber auch eine Gruppe bestehender Objekte ausgewählt werden, die die gewünschten Objekte beinhalten.

Es ist sinnvoll, nur Objekte zu verwenden, die Action Listen der Objekte innerhalb des neuen Objektes benutzen. Sollten die Action Listen auf Aktionen außerhalb des Objektes zugreifen, werden die benutzer-definierten Objekte in anderen Anwendungen nicht funktionieren.

Als nächstes wird das Menükommando UTILITIES/PACKAGE AS USER-DEFINED verwendet, um diese Auswahl zu einem neuen Objekt zu „packen“.

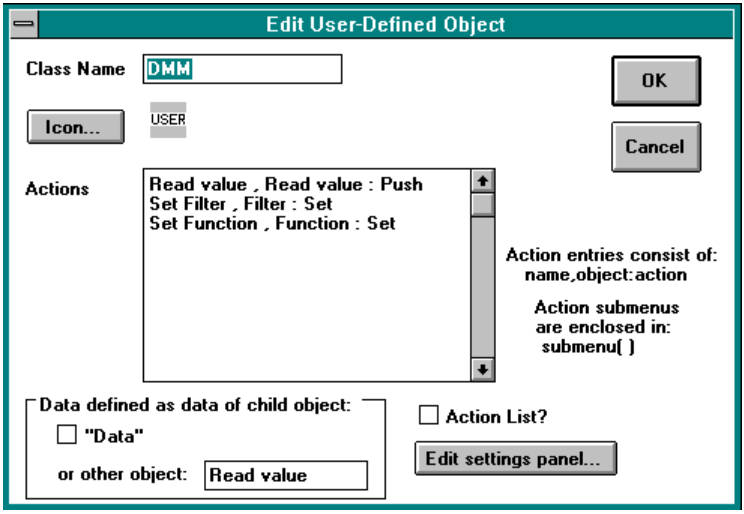


Beachten Sie, daß die Originalobjekte sich nun innerhalb des benutzerdefinierten Objektes und nicht mehr auf dem Panel befinden!

Plazieren Sie keine Objekte, die Sie auf dem Panel behalten wollen, in das benutzerdefinierte Objekt. Wenn es sich nicht vermeiden läßt, fügen Sie neue auf dem Panel ein, die deren Platz für diese Aufgaben einnehmen.

Jedes Eingabe- (wie Data-Entry, Pushbutton, Slider,...) und Action Objekt, das gepackt wird, wird zur Aktion im neuen benutzerdefinierten Objekt.

Ein Fenster erscheint, mit dem verschiedene Aspekte des neuen Objekts angepaßt werden können:



Zu diesem Fenster kommen wir in den folgenden Abschnitten.

Spezielle Objekte im benutzerdefinierten Objekt

Wenn Sie das Pack-Kommando benutzt haben, erzeugt TestPoint automatisch drei spezielle Data-Entry Objekte innerhalb des neuen benutzerdefinierten Objektes: **Action Names, Class Name,** und **UI Name**.

Diese Objekte beinhalten Informationen über die Definition des benutzerdefinierten Objektes (die gleiche Information, die in dem Fenster „Edit User-Defined Object“ vorher dargestellt wurde).

Weitere spezielle Objekte können erzeugt werden, wenn Sie das Bearbeitungsfenster verwenden. Wenn Sie den Schalter „Action List?“ anklicken, wird ein Objekt mit dem Namen „**Action List**“ angefügt.

Die folgenden Namen für spezielle Objekte sind reserviert und haben innerhalb eines benutzerdefinierten Objektes einer spezielle Bedeutung:

Action List
Action Names
Class Name
Data
Icon
Initialize
Lock
Settings
UI Name

Bearbeiten oder löschen Sie diese Objekte nie direkt, sondern verwenden Sie stattdessen das Menükommando „EDIT USER-DEFINED“ dazu.

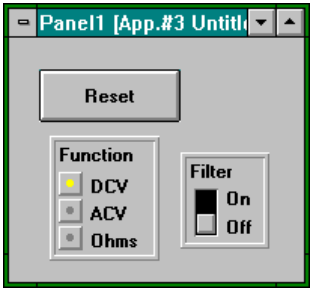
Definieren von Aktionen

Wie schon erwähnt sind die Aktionen, die ein benutzerdefiniertes Objekt beim Ziehen in eine Aktionsliste zur Verfügung stellt, alles Aktionen von Objekten innerhalb des benutzerdefinierten Objekts. Sie werden im ‘Actions’ Feld im ‘User-defined Object Edit’-Fenster definiert.

Aktionen, die beim Packen des Objekts entstehen

Wenn eine Gruppe von Objekten ausgewählt und mit dem Menükommando ‘Package as User-Defined’ gepackt wurde, wird aus jedem Objekt im Panel, das eine Eingabe erlaubt, eine Aktion im neuen Objekt. Das schließt Druck-, Schiebe-, Kippschalter, Selectoren und Dateneingabefelder ein.

Zum Beispiel, dieses Fenster :



nach dem Packen wird es ein neues Objekt mit diesem Aktionsmenü:

Hinzufügen eines Objekts zum benutzerdefinierten Objekt

Wird ein Objekt innerhalb des benutzerdefinierten hinzugefügt, so gelten zu jeder Zeit die gleichen Regeln wie bei dem ersten Packen: aus Eingabeobjekten wird eine neue Aktion.

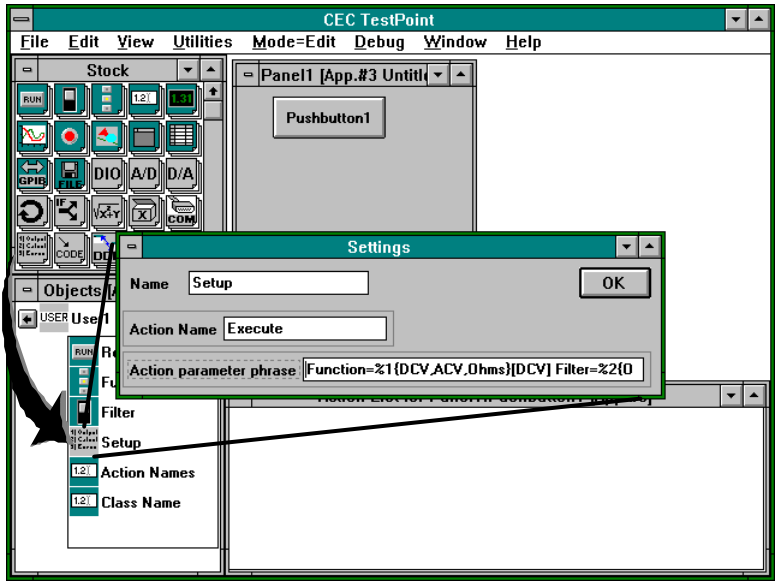
Also, wenn mit dem rechten Pfeil im Objekt Fenster in das benutzerdefinierte Objekt geschaltet und dann ein neues Objekt, wie etwa ein Dateneingabefeld, eingefügt wird, so hat das benutzerdefinierte Objekt nun eine neue Aktion.

Eine ganz neue Aktion definieren

Die auf die enthaltenen Objekte basierende automatische Aktions-Definierung reicht möglicherweise nicht für alle Zwecke aus.

Angenommen, man möchte eine einzelne Aktion mit verschiedenen Parametern ausstatten, anstelle gesonderter Aktionen für jedes Eingabefeld und jeden Selector. Um dies zu tun, wird nur ein ‘Action’ Objekt innerhalb des benutzerdefinierten Objekts benötigt.

Zum Beispiel : Das benutzerdefinierte Objekt aus dem Beispiel vorher, mit ‘Set Function’ und ‘Set Filter’ Aktionen. Um daraus eine neue Aktion zu gestalten, die beide Parameter einstellen kann, zieht man ein ‘Action’ Objekt hinzu und stellt es so ein:

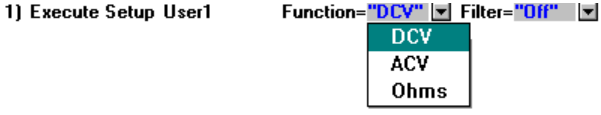


Die „Action parameter phrase“ Einstellung ist:

```
Function=%1{DCV,ACV,Ohms}[DCV] Filter=%2{On,Off}[Off]
```

Damit hat die neue Aktion zwei Parameter mit ‘drop down’ Auswahllisten und Default Werten (siehe im Referenz Teil unter ‘Action’ Objekt)

Die neue Aktionszeile des benutzerdefinierten Objekt stellt sich wie folgt dar:

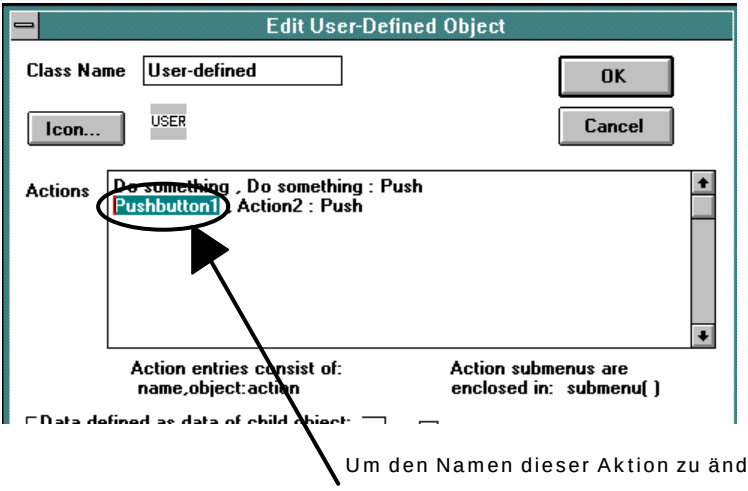


Löschen einer Aktion

Ist eine automatisch erzeugte Aktion nicht gewünscht, oder möchte man eine vorher definierte Aktion loswerden, so geschieht dies wieder im ‘Object Edit Fenster’ (‘Utilities’ ‘Edit User-Defined’). Es werden einfach die zu löschenden Aktionen im „Edit User-Defined“-Fenster markiert und mit der Entf- (DEL) Taste gelöscht.

Umbenennen einer Aktion

Standard Namen werden bei der automatischen Erzeugung der Aktionen verwendet. Diese sind ebenfalls im ‘Object Edit Fenster’ veränderbar. Dazu werden die Namen der Aktionen im ‘Action’ Feld verändert. Der Aktionsname ist der erste Teil in der Aktionszeile vor dem Komma.



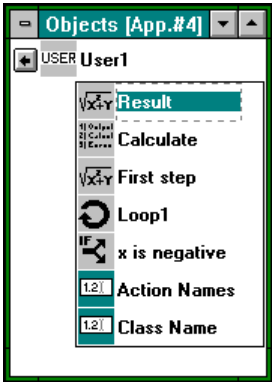
Um den Namen dieser Aktion zu ändern, ersetzen Sie nur den Text (vor dem Komma).

Rückgabe von Datenwerten

Die Aktionen benutzerdefinierter Objekte bekommen Daten von den Aktionsparametern beim Ausführen der Aktionen. Doch wie bekommt man Daten in die aufrufende Aktionsliste zurück?

Genau so, wie die vorhandenen TestPoint Objekte die sie beinhaltenden Werte nach Ausführung einer Aktion verändern, trifft das auch für die benutzerdefinierten Objekte zu.

Der Datenwert eines benutzerdefinierten Objekts ist definiert entweder durch irgendein Objekt innerhalb mit dem Namen ‘Data’ oder dem an der ersten Stelle stehenden Objekt im Objekt Fenster. Beispiel: Wenn ein benutzerdefiniertes Objekt eine komplexe mathematische Berechnung durchführt, einschließlich Schleifen und Bedingungen, das Ergebnis aber von einem Mathematik Objekt kommt, kann dies an die erste Stelle gerückt werden:



Nun, nachdem die Aktion abschließt (der letzte Schritt in der ‘Calculate’ Action Liste ist die Berechnung von ‘Result’), ist der Datenwert des benutzerdefinierten Objektes in der aufrufenden Aktionsliste wie folgt verfügbar:

1)	Calculate	User1	with x=3, y=5
2)	Set	Display1	to User1

Settings

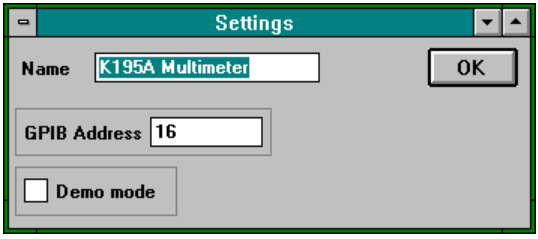
Benutzerdefinierte Objekte können, wie jedes andere TestPoint Objekt auch, Einstellungen haben.

Um diese für ein benutzerdefiniertes Objekt zu definieren, benötigt man ein Fenster mit dem Namen ‘Settings’ innerhalb des benutzerdefinierten Objekts (das Hinzufügen kann manuell oder im „Edit User-Defined“ Fenster’ geschehen).

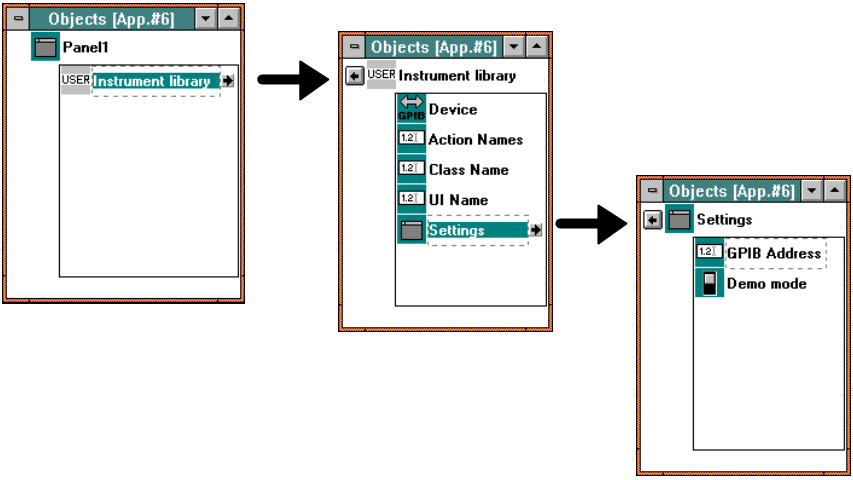
Hier können nun alle gewünschten Objekte plaziert werden. Die Werte dieser Objekteinstellungen können in den Action Listen innerhalb des benutzerdefinierten Objekts weitere Verwendung finden.

Wenn Sie die Bearbeitung der Einstellungen beendet haben, doppelklicken Sie auf die Systembox, oben links am Fenster.

Eine gewöhnliche Anwendung der Einstellungen eines benutzer-definierten Objekts kann in den GPIB Geräte Bibliotheken gefunden werden. Jede Gerätebibliothek ist ein benutzerdefiniertes Objekt mit vielen Aktionen für ein spezielles Meßgerät. Ein typisches Bibliothek-Objekt hat Einstellungen für die GPIB Adresse und für Demo Mode:



Und dies ist die Objektliste innerhalb des benutzerdefinierten Objekts:



dDie Action Liste für die ‘GPIB address’ Einstellung, welche dem GPIB Objekt den Wert der ‘GPIB address’ Einstellung zuweist, wenn diese verändert wird:

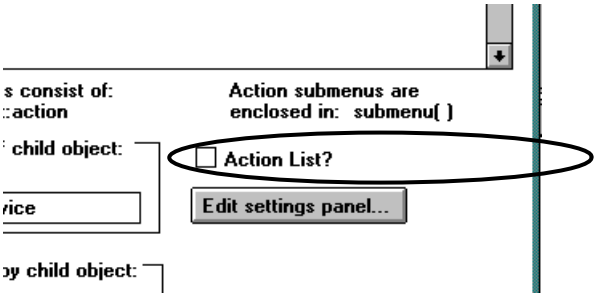
1) Set Device(GPIB Address) to Instrument library(GPIB Address)

(Das „Device“ GPIB Objekt ist innerhalb der Bibliothek verwendet, um das Gerät anzusprechen. "K195A Multimeter" ist der Name des Bibliotheks Objektes selbst.)

Action Listen

Hinzufügen einer Action-Liste

Ein benutzerdefiniertes Objekt kann auch eine Action Liste haben. Hinzugefügt wird diese durch ein Objekt mit dem Namen „Action List“. Das wird automatisch angefügt, wenn Sie in dem „Edit User-Defined“ Fenster den Schalter „Action List“ anklicken:



Dieses Objekt muß kein ‘Action Phrase Setting’ besitzen, auch sollte es keine Aktionszeilen enthalten. Es ist nur ein Platzhalter mit einem speziellen Namen, um TestPoint mitzuteilen, daß das benutzerdefinierte Objekt eine Action Liste haben soll.

Wann wird die Action-Liste ausgeführt?

Alle TestPoint Objekte mit einer Action Liste haben ein Ereignis definiert, bei dessen Eintreten die Action Liste ausgeführt wird. Pushbutton Objekte führen ihre Action Liste aus, wenn der Anwender auf den Schalter klickt. A/D Objekte führen die Action Liste aus, wenn die Erfassung beendet ist.

+ Die Action Liste eines benutzerdefinierten Objektes wird immer dann ausgeführt, wenn der Programmierer es in einer Action Liste definiert.

Damit die Action Liste des benutzerdefinierten Objekts ausgeführt wird, muß diese von einem anderen, in ihm enthaltenen Objekt aufgerufen werden.

Beispiel: Ein benutzerdefiniertes Objekt enthält einen Schalter mit dem Namen „Read“. Dieser soll einen neuen Wert von einem Gerät lesen. Anschließend soll die Action Liste des benutzerdefinierten Objektes ausgeführt werden:

Die Action Liste des "Read" Schalters innerhalb des benutzerdefinier ten Objekts:

1)	Output to	GPIB1	":MEAS:VOLT:DC?"
2)	Enter from	GPIB1	up to 256 bytes
3)	Set	Data	to GPIB1
4)	Execute my	Action List	

Zeile 1 sendet den Befehl zum Messen und Zeile 2 liest den gemessenen Wert vom GPIB-Gerät ein. Zeile 3 setzt das ‘Data’ Objekt zum gelesenen Wert, so daß das benutzerdefinierte Objekt nun mit dem aktuellen Wert arbeitet. Zeile 4 startet die Action Liste.

Wenn also die Hauptanwendung ein ‘Timer’ Objekt mit dieser Action Liste beinhaltet:

```
1) Read                 User1
```

und die Action Liste des benutzerdefinierten Objekts:

```
1) Set                  Display1        to User1
```

enthält, dann wird die Anzeige periodisch mit neuen Meßwerten aufgefrischt.

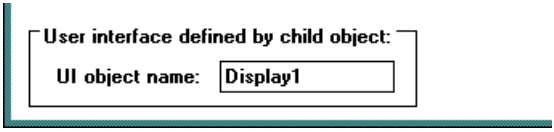
Hervorzuheben ist, daß das benutzerdefinierte Objekt wie ein vorhandenes Objekt mit einer neuen Aktion namens ‘Read’ aussieht und die Action Liste des Objekts ausgeführt wird, wenn ‘Read’ aufgerufen wird.

Benutzerschnittstellen

Einige Objekte im TestPoint, wie Data-Entry- und Graph-Objekte haben Benutzerschnittstellen, andere, wie das Math-Objekt, haben keine.

+ Sie können einem benutzerdefinierten Objekt eine Benutzerschnittstelle geben, indem Sie einfach die jeweilige eines Objektes innerhalb des benutzerdefinierten Objekts auswählen.

Wenn Sie beispielsweise ein Objekt mit dem Namen „Display1“ in Ihrem benutzerdefinierten Objekt haben, geben Sie diesen Namen einfach im „Edit User-Defined“ Fenster als UI-Objektnamen an.

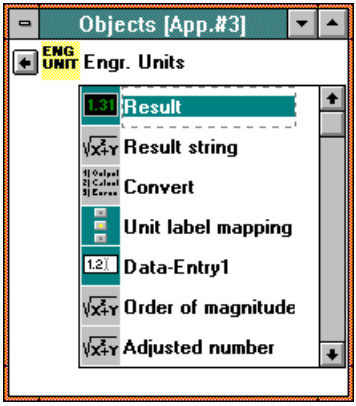


Das benutzerdefinierte Objekt sieht dann aus, wie ein Display-Objekt, wird aber Daten und Aktionen Ihrer Definition besitzen.

Beispiel: Anzeigen von pyhsikalischen Einheiten

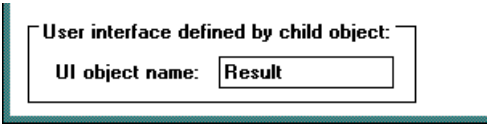
Als Beispiel finden Sie im Unterverzeichnis EXAMPLES die Datei ENGR.TST, die Zahlen in physikalische Einheiten mit metrischen Vorzeichen erzeugt. Das Objekt gibt eine Zeichenkette als Datenwert zurück.

Um eine neue Art von Display Objekt zu erzeugen, das wie ein reguläres Objekt aussieht, jedoch Zahlen in physikalischen Einheiten anzeigt, fügen Sie innerhalb des benutzerdefinierten Objektes „Engr. Units“ ein Display Objekt mit dem Namen „Result“ ein (Beachten Sie, daß das Objekt in ENGR.TST gesperrt ist, Sie es aber mit dem Paßwort „library“ und dem Menükommando LOCK/UNLOCK USER-DEFINED „entsperren“ können).

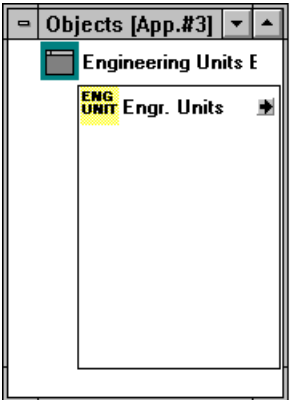


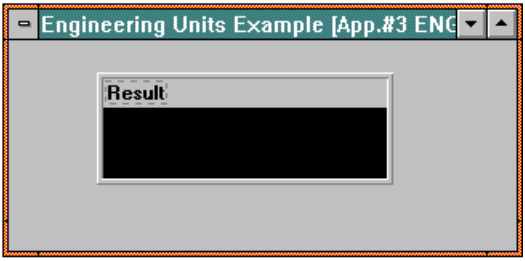
Die Action Liste von „Convert“ muß das Ergebnis im Display darstellen.

Selektieren Sie das Objekt „Engr. Units“ und wählen Sie das Menükommando EDIT USER-DEFINED. Geben Sie „Result“ als „UI Objectname“ an:



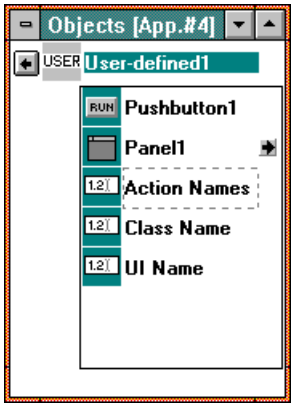
Nun sieht das Objekt so aus:





Panels in benutzerdefinierten Objekten

Die zusammengepackten Objekte erscheinen nicht mehr auf dem Hauptpanel. Trotzdem können Sie die in ein benutzerdefiniertes Objekt eingebetteten Objekte durch die Verwendung von Unterpanels dem Anwender zur Verfügung stellen:



Das Panel Objekt mit dem Namen „Panel1“ kann von einer Action Liste innerhalb des benutzerdefinierten Objektes angezeigt werden, wie hier:

1) Show & Wait Panel1

Details der Bearbeitung von benutzer-definierten Objekten

Das beim Packen oder später durch das Menükommando ‘Edit User-defined’ erscheinende Fenster, läßt die Eigenschaften des Objekts verändern:

Class Name - Dieses Feld deklariert den Oberbegriff für die neue Objekt-Klasse. Diese wird benutzt, wenn das Objekt in den ‘stock’ gebracht wird oder neue Kopien davon erstellt werden.

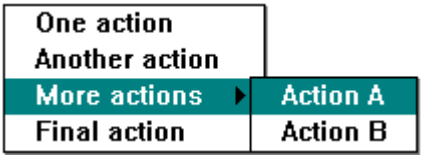
Icon - soll das Objekt ein eigenes Bildsymbol erhalten, so benötigt man dafür ein 24 mal 24 Bitmap (z. B. mit Windows Paintbrush erzeugte Datei). Wenn erwünscht, einfach die Option ‘Icon’ wählen.

Actions - dieses Feld definiert die Aktionen für das neue benutzerdefinierte Objekt, angezeigt wird eine Aktion pro Zeile. Jede Aktionsdefinition besteht aus dem Namen der Aktion (welcher ein oder mehr Wörter lang sein kann), dann ein Komma, der Name des Objekts, welches die Aktion tatsächlich ausführen wird, ein Doppelpunkt und die Aktion, die auszuführen ist. In dieser Weise werden die Aktionen des benutzerdefinierten Objekts in Bezug auf die ausgeführten Aktionen der Objekte, die innerhalb des benutzerdefinierten Objekts gepackt sind, definiert. Das Menükommando zum Packen wählt automatisch die Aktionen auf Basis der Benutzerschnittstelle des Originalfensters (dessen Schalter, etc.) aus. Wenn neue Objekte innerhalb des benutzerdefinierten Objekts hinzugefügt werden, so werden diese automatisch dieser Liste hinzugefügt. Wenn Aktionen umbenannt, gelöscht oder hinzugefügt werden sollen, so kann das Aktionsfeld in diesem Fenster editiert werden (wichtig obige Konvention beachten).

Wird ein benutzerdefiniertes Objekt in eine Action Liste gezogen, so erscheint das ‘pop up’ aller ‘Actions’ mit der Liste der Aktionen, die in diesem Feld definiert wurden. Bei Objekten mit einer großen Anzahl von Aktionen empfiehlt es sich, ein hierarchisches ‘Pop up’ Aktions-Menü zu erstellen. Um eine Gruppe von Aktionen in ein Untermenü zu verschachteln, schließt man diese nur durch Klammern ein und setzt ein Untermenü Titel davor, wie hier zum Beispiel:

```
One action, Object1:Action1
Another action, Object2:Action2
More actions (
                    Action A, Object3:Action3
                    Action B, Object4:Action4
)
Final action, Object5:Action5
```

Das erzeugt ein kaskadiertes Popup Menü wie folgt:



Data - Die Daten des benutzerdefinierten Objekts basieren ähnlich wie die Aktionen auf Daten eines Objekts, welches im benutzerdefinierten Objekt enthalten ist. Ein Kreuz in der Box vor Data erzeugt einen neuen Container mit Namen ‘Data’ innerhalb des benutzerdefinierten Objekts. Dieser Container wird dann benutzt, wenn Daten vom diesem benutzerdefinierten Objekt angefordert werden. Jede beliebige Action Liste innerhalb des benutzerdefinierten Objekts kann jede gewünschte Information in diesen Container legen.

Alternativ dazu kann ein beliebiges Objekt innerhalb des benutzerdefinierten als Datenangabe dienen, wenn anstatt die Check-Box vor Data anzukreuzen, der Name des Objekts in das Eingabefeld eingetragen wird.

Action List - Die meisten benutzerdefinierten Objekte enthalten keine Action Liste. Sie können jedoch eine haben, wenn es gewünscht wird. Kreuzt man diese Box an, so wird durch das automatische Hinzufügen eines ‘Action’ Objekts mit dem Namen ‘Action List’ innerhalb des benutzerdefinierten eine Befehlsliste erzeugt.. Die Action Liste dazu ist wie die Action Liste eines jeden anderen Objekts zu editieren: selektieren des benutzerdefinierten Objekts in der Objektliste, dann das Menü ‘View’ ‘Action List’ auswählen und Befehle hinzufügen.

Vorhandene TestPoint Objekte, die Action Listen besitzen, führen diese basierend auf verschiedene Ereignisse aus. Beispielsweise vollzieht ein Dateneingabe-Objekt seine Action Liste, nachdem ein neuer Wert in dessen Feld eingegeben wurde. Das benutzerdefinierte durchläuft seine Action Liste, wenn eines der in ihm enthaltenen Objekte das ‘Action’-Objekt ‘Action List’ aufruft.

Settings - Es kann ein Einstellungsfenster für das benutzerdefinierte Objekt erstellt werden, indem man den entsprechend benannten Schalter drückt. So wird dem benutzerdefinierten Objekt ein neues Fenster namens "Settings" hinzugefügt. Darauf kann jedes gewünschte Objekt plaziert werden, wie Dateneingabefelder, Selectoren, etc. Die Werte dieser Objekte sind in den Action Listen innerhalb des benutzerdefinierten Objekts weiterverwendbar. Wird ein Doppelklick auf das benutzerdefinierte Objekt getätigt, so erscheint ein Einstellungsfenster, welches wie die der vorhandenen Objekte arbeitet.

Bearbeitung - Das gleiche Fenster erscheint wieder, wenn das benutzerdefinerte Objekt zuerst gepackt und dann später editiert werden soll. Dazu selektiert man es einfach in der Objekt Liste und wählt in der Menüzeile ‘Utilities’ ‘Edit User-Defined’ aus.

Gebrauch von benutzerdefinierten Objekten

Lagern des neuen Objekts Wenn das neue Objekt in vielen weiteren Anwendungen genutzt wird oder an andere TestPoint Anwender weitergegeben werden soll, so kann es in das Stock-Fenster gebracht werden. Dazu wird das Objekt in den Stock gezogen und bei gleichzeitigem Drücken der STRG-Taste abgelegt. Jetzt ist es Bestandteil des Stock und kann ab jetzt wie alle anderen Objekte verwendet werden.

Weitergeben von Objekten Der einfachste Weg, benutzerdefinierte Objekte weiterzugeben, ist, eine Anwendung, die dieses Objekt enthält, abzuspeichern. Die Ergebnisdatei *.TST kann nun an andere weitergegeben werden, die dieses Objekt einfach in ihre Anwendung kopieren oder das Objekt in ihren Stock schieben.

Sperren Wenn benutzerdefinierte Objekte an andere weitergegeben werden, so ist es manchmal wünschenswert, daß die internen Details nicht sichtbar gemacht oder verändert werden können. Dazu dient das Menükommando 'Lock/Unlock User-Defined'. Diese Kommando sperrt ein benutzer-definiertes Objekt, wenn es offen ist und umgekehrt. Beim Schließen wird man nach einem Schlüssel oder einem Paßwort gefragt, welches ein beliebiger String von vier bis vierzig Zeichen sein kann. Dieser Schlüssel wird in die TESTPT.INI Datei der eigenen TestPoint Installation hinzugefügt. Damit kann das Objekt wieder geöffnet werden, ohne das Paßwort eingeben zu müssen. Wenn jedoch eine Kopie des Objekts an andere weitergegeben wird, so können diese das Objekt ohne Kenntnis des Schlüssels nicht öffnen.

Geschlossene benutzerdefinierte Objekte erscheinen wie normale Objekte, die Pfeile sind dann nicht vorhanden.

Objekte als Bibliotheken Benutzerdefinierte Objekte können als wiederverwendbare, aus Anwendungsteilen bestehende Bibliothekelemente, fungieren. Jeder Teil einer TestPoint Anwendung, die in anderen Programmen brauchbar ist, kann gepackt und wiederverwandt werden. Ein ganzer Satz ähnlich gearteter benutzerdefinierter Objekte kann so in Form einer Programmbibliothek zur eigenen Verwendung oder Weitergabe angelegt werden.

Solche Bibliotheken können sein: geräteabhängige Bibliotheken, die dann die Steuerung auf hoher Ebene ermöglichen; Mathematikbibliotheken, bestehend aus komplexen Routinen, erzeugt mit Schleifen, Speichern und Bedingungen; Programm Interfaces, die DDE-Transfer oder Dateikonvertierungsroutinen enthalten.

Modularisierung großer Anwendungen

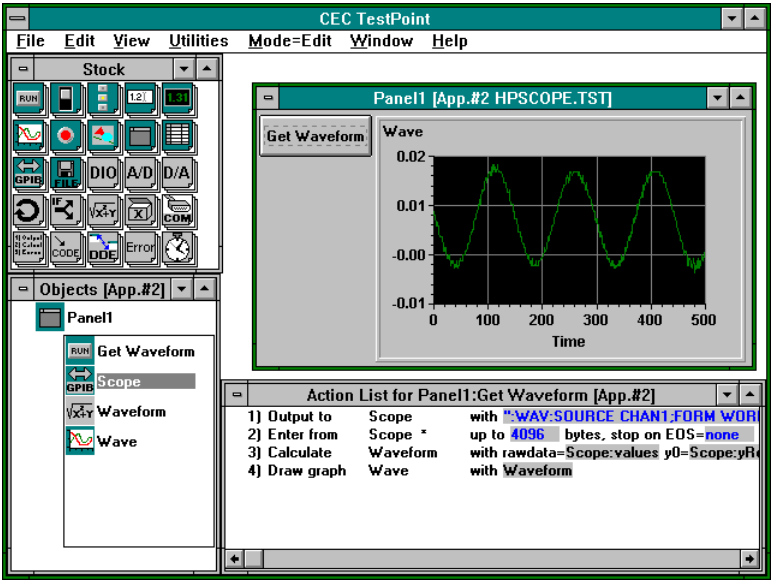
Je mehr Funktionen Sie in Ihr TestPoint Programm einbauen, desto größer werden die Befehlslisten und die Anzahl der verwendeten Objekte, so daß die Anwendung beginnt, unübersichtlich zu werden. Sie tun sich dann schwer, die Anwendung noch zu verstehen oder bestimmte Objekte zu finden.

Sie können Gruppen und Panels verwenden, die bei der Organisation von Objekt Listen helfen.

Eine weitere Lösung lautet: Verwendung von benutzerdefinierten Objekten. Diese können andere Objekte enthalten, stellen jedoch nur ein Einzelobjekt dar, welches Ihre Auswahl an Daten und Funktionen enthält.

Fallstudie

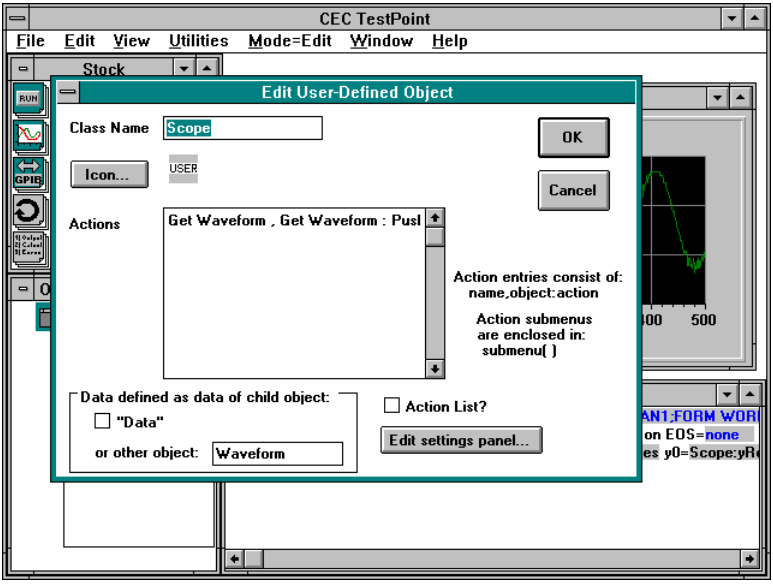
Hier ist eine Applikation, die eine digitalisierte Kurve vom Oszilloskop einliest (ähnlich dem mitgelieferten Beispielprogramm HPSCOPE.TST):



Dieses Programm ist noch nicht groß (nur 4 Objekte und 4 Befehlszeilen), aber es enthält auch noch keine Funktionen, um einen kompletten Test zu realisieren. Das Ziel in diesem Fall ist, eine Anwendung für die Produktion zu erstellen, die eine Kurvenform einliest, mit einer minimalen und maximalen Kurve vergleicht und eine ‘gut - schlecht’ Auswertung anzeigt. Dies verlangt nach mehr Objekten : Dateizugriff, Mathematik, Anzeige, usw.

Als ersten Schritt packen Sie den ‘get waveform’ Teil der Anwendung in ein benutzerdefiniertes Objekt, da wir keine Details sehen wollen, während wir am Rest der Anwendung arbeiten. Wir packen den Schalter, das GPIB und das Mathematik Objekt, nicht aber das Grafik Objekt.

Beginnen Sie, den Schalter, das GPIB Objekt, dann das Mathematik Objekt auszuwählen, indem Sie den Schalter anklicken und die Maus innerhalb der Objektliste über die restlichen Objekte ziehen. Konvertieren Sie jetzt in ein benutzerdefiniertes Objekt mit dem Menükommando ‘Utilities’ ‘Package as User-Defined’. Wenn das ‘User-Defined Dialog’-Fenster erscheint, setzen Sie den Klassennamen als ‘Scope’ ein und geben Sie ‘Waveform’ in das ‘data defined as’ Eingabefeld ein. Damit hat das neue Objekt die Daten des Mathematik-Objekts genannt ‘Waveform’ (Dieses Objekt beinhaltet den Spannungsvektor der Kurvenform):



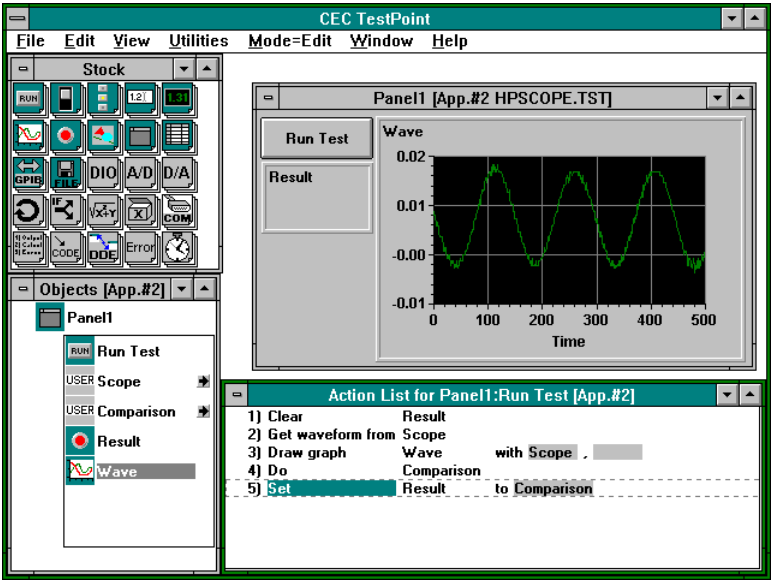
Nun löschen Sie die 4. Befehlszeile in der Action Liste für den Schalter, da wir das neue Objekt (get waveform from Scope) zum Einlesen, nicht aber zum Darstellen der Kurve nutzen wollen. Wir definieren ein Modul oder Unterprogramm, welches vom GPIB einliest und die Daten-Formatierungs, -Skalierungs und Offsetoperationen vornimmt.

Das ist alles fürs Packen des benutzerdefinierten Objekts.

Nun ziehen Sie einen Schalter vom ‘stock’ in Ihr Hauptarbeitsfenster und geben ihm den Namen ‘Run Test’. Öffnen Sie nun seine Action Liste und ziehen das ‘Scope User-Defined’ Objekt hinein. Sie bekommen eine Befehlszeile "Get Waveform Scope", was ein Aufruf des benutzerdefinierten Objekts ist, welches alle Details zum Einlesen vom ‘Scope’ enthält (Wenn Sie wollen, können Sie das benutzerdefinierte Objekt editieren und den Namen des Befehls in "Get Waveform from" ändern, was vielleicht ein wenig verständlicher in der Action Liste klingt, aber nur Kosmetik ist).

In ähnlicher Weise können Sie auch noch das ‘File’ Objekt und das Mathematik Objekt, welches den Vergleich ermöglicht, nach dem Hinzufügen in ein benutzerdefiniertes Objekt packen.

Die endgültige Anwendung sieht dann so aus:



Für Fortgeschrittene: Action Objekte in benutzerdefinierten Objekten

In dem letzten Beispiel, dem ‘Comparison User-Defined’ Objekt, wurde von einer Datei eingelesen und die eingelesene Kurve mit ‘min’ und ‘max’ Werten verglichen.

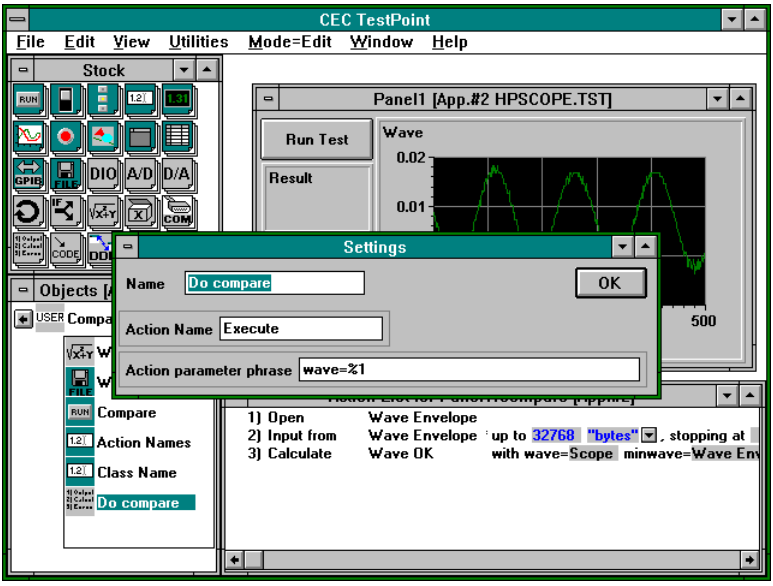
Die Action Liste für den "Compare" Schalter innerhalb des benutzerdefinierten Objekts sieht etwa so aus:

```
1) Open          Wave Envelope
2) Input from    Wave Envelope    up to 32768 bytes
3) Calculate     Wave OK          with wave=Scope
                                  minwave=Wave Envelope:Min
                                  maxwave=Wave Envelope:Max
```

Zeile 3 bezieht sich direkt auf das ‘Scope’ Objekt (Das andere benutzerdefinierte Objekt in der Anwendung), welches die neu gemessene Kurve enthält.

Es würde sinnvoll sein, wenn das ‘Comparison’ Objekt alle Kurven vergleichen könnte, nicht nur das eines ‘Scope’ Objekts. Dies würde erlauben, das ‘Comparison’ Objekt auch in andere Anwendungen zu kopieren, oder in der gleichen Anwendung auch andere Kurven auf bestimmte Grenzen hin zu vergleichen und so das Objekt mehrmals verwendbar machen. Um das zu erreichen, müssen wir unsere eigene Aktion für das "Comparison" Objekt, mit einem Parameter für die zu vergleichende Kurve, definieren.

Als erstes schauen wir in das Innere des ‘Comparison’ Objektes, indem Sie den rechten Pfeil neben dem Objekt in der Objektliste anklicken. Fügen Sie nun ein ‘Action’ Objekt in Ihre Anwendung ein. ‘Action’ Objekte werden verwendet, um neue Befehlszeilen zu definieren, welche jeden beliebigen Ausdruck und eine Parameterliste enthalten können. Geben Sie dem neuen Objekt den Namen "Do Compare" und geben Sie den Ausdruck "wave=%1" in das Settings Fenster ein.



Nun kopieren Sie die Befehlszeilen des ‘Compare’ Schalters und fügen diese in die Action Liste der ‘Do Compare’ Aktion ein. Danach ändern Sie Zeile 3 folgendermaßen:

3) Calculate Wave OK with wave=%1
minwave=Wave Envelope:Min
maxwave=Wave Envelope:Max

Das bedeutet, daß das Aktion Objekt die Parameter benutzt, die diesem zur Laufzeit übergeben werden, wenn die Berechnung ausgeführt wird.

Editieren Sie jetzt das ‘Comparison’ benutzerdefinierte Objekt mit dem Menükommando ‘Utilities’ ‘Edit User-Defined’. Ändern Sie die existierende Aktionsdefinition in ‘Do , Do Compare:Execute’. Nun ist die Aktion des benutzerdefinierten Objekts dazu definiert, ein ‘Action’ Objekt aufzurufen und dessen Ausdrucksliste zu benutzen.

Gehen Sie zurück in das Hauptarbeitsfenster, indem Sie den Pfeil nach links in der Objektliste anklicken. Lassen Sie sich die Action Liste für den Schalter ‘Run Test’ anzeigen. Selektieren und löschen Sie die Zeile 4, welche den ‘Compare’ Schalter auslöst. Ziehen Sie nun das ‘Comparison’ Objekt herein und schauen Sie sich die neue Befehlszeile an. Füllen Sie das Parameterfeld mit dem ‘Scope’ Objekt. Die Action Liste sieht nun folgendermaßen aus:

1)	Clear	Result	
2)	Get waveform from Scope		
3)	Draw graph	Wave	with Scope
4)	Do	Comparison	wave=Scope
5)	Set	Result	to Comparison

Nun ist das ‘Comparison’-Objekt ein allgemeines Modul, welches den Vergleich einer beliebigen Kurve mit einer Hüllkurve erlaubt. Mit dem Hinzufügen von weiteren Parametern kann das Objekt auch noch den Dateinamen als Variable enthalten usw.

Wie benutzerdefinierte Objekte in Ihrem Programm verwendet werden können

1) Entscheiden Sie sich für eine zusammengehörige Gruppe von Objekten, welche in ein einzelnes Objekt gepackt werden sollen. Diese Gruppe sollte normalerweise nur Befehlslisten haben, die sich nur auf Objekte innerhalb der Gruppe beziehen. In einigen Fällen können Sie noch kleinere Änderungen vornehmen, damit ein klares separates Modul entsteht.

2) Benutzen Sie das UTILITIES/PACKAGE AS USER_DEFINED Menükommando, um eine Gruppe von Objekten in ein benutzerdefiniertes zu verpacken. Editieren Sie die 'Action' Namen wenn gewünscht und legen Sie das korrekte Daten Objekt fest.

3) (optional). Wenn Sie Aktionen mit Parametern für Ihr neues Objekt haben möchten, fügen Sie 'Action' Objekte mit Ausdruckslisten hinzu, wie in diesem Kapitel bereits beschrieben.

4) Nun benutzen Sie Ihr neues benutzerdefiniertes Objekt in Befehlslisten im Hauptanwendungsfenster.

Rückblick:

- Packen: Benutzerdefinierte Objekte werden durch das Packen eines Fensters mit allen enthaltenen ‘Child’ Objekten erstellt. Die ‘Child’ Objekte bestimmen das Verhalten des neuen benutzerdefinierten Objekts.
- Aktionen: Aktionen können dem benutzerdefinierten Objekt hinzugefügt werden, um völlig neue Aktionszeilen mit beliebigen Ausdrücken und Parameterlisten zu erzeugen.
- Weitergabe: Benutzerdefinierte Objekte können in eine Anwendungsdatei plaziert und an andere TestPoint Programmierer weitergegeben werden. Optional können solche Objekte geschlossen werden, um den Einblick in deren Aufbau zu verbieten.

Kapitel 23.
Aufruf externer Funktionen

TestPoint hat viele mächtige Funktionen bereits eingebaut. Wie auch immer, es gibt immer Anwendungen, in denen TestPoints Möglichkeiten erweitert werden können. Aus diesem Grund beinhaltet TestPoint die Möglichkeit, externe Routinen aufzurufen.

Was in diesem Kapitel behandelt wird:

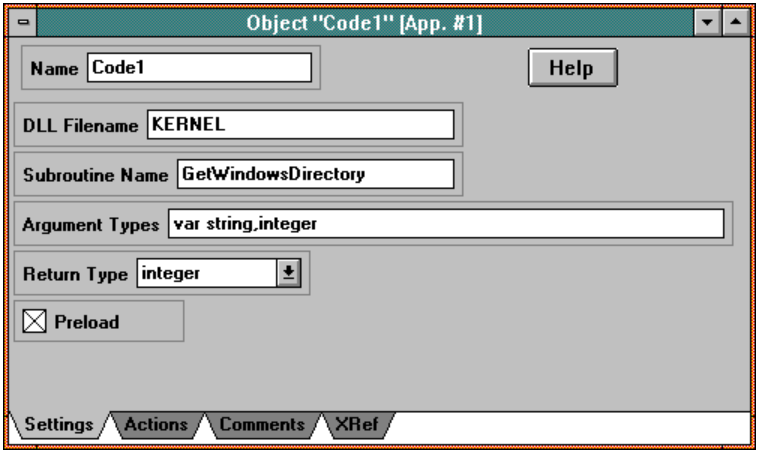
- Das Benutzen des Code Objekts.
- Zugriff auf kundenspezifische Hardware.
- Schreiben von Mathematikfunktionen.
- Generieren von kundenspezifischen TestPoint Ereignissen.

Das Code Object

Das Code Objekt erlaubt es, externe Unterprogramme von TestPoint´s Action Listen aus aufzurufen, Daten an diese Routinen zu übergeben, bzw. Daten von diesen zu empfangen. Jedes Code Objekt repräsentiert eine externe Funktion.

Der externe Code muß in der Form einer Windows Dynamic Link Library (DLL) vorliegen. So werden alle Windows Funktionen zur Verfügung gestellt und die meisten Hardware Treiber geschrieben. Sie können Ihre eigene DLL in Programmiersprachen wie C, C++, oder PASCAL programmieren. Microsoft, Borland und andere Firmen stellen Werkzeuge zur Verfügung, um DLLs zu erstellen. (Achtung: Microsoft Visual Basic 3.0 unterstützt das Erstellen von DLLs nicht).

Die Code Objekt Einstellungen beschreiben den DLL Dateinamen, die die externe Funktion beinhalten, den Namen der externen Routine und die Typen der Parameter, die übergeben werden.



Der DLL Dateiname und der Funktionsname können sowohl groß als auch auch klein geschrieben werden.

Code Argumente

Argumenttypen können sein:

char	1 signed byte
byte	1 unsigned byte
integer	2 bytes (signed)
word	2 bytes (unsigned)
long	4 bytes (signed)
dword	4 bytes (unsigned)
float	4 bytes, floating pt.
double	8 bytes, floating pt.
string	pointer to string (0-terminated)
hwnd	"handle" to an object's window, for calling Windows functions
window	"handle" to the Code object's own event window (see later)

Argumente können als Schlüsselwort auch **var** enthalten, um zu kennzeichnen, daß der Parameter als Pointer übergeben wird und das Ergebnis an TestPoint zurückgegeben wird.

Ein Argument kann auch eckige Klammern und optional die Länge eines Vektors (z. B. **[1024]**, oder **[]**) enthalten, um anzuzeigen, daß es sich um einen Vektor oder ein Feld von Werten handelt. Vektoren werden immer als Pointer auf eine Speicheradresse übergeben.

Code aufrufen

Wenn ein Code Objekt in einer TestPoint Action Liste benutzt wird, dann ist dort ein Parameter für jedes Argument bereitgestellt. Zum Beispiel : das "GetWindowsDirectory" code Objekt, benutzt in dem Einstellungsfenster vorher, produziert diese Befehlszeile:

```
1) Call        Code1        with _____, _____
```

Die Argumente können genauso hineingezogen werden, wie bei jeder andern Befehlszeile auch. Vor den Argumenten, vor denen das Schlüsselwort **var** angegeben wurde, oder bei Vektoren können keine Konstanten eingegeben werden. Sie verlangen nach einer Variablen (TestPoint Objekt), da die von der externen Funktion kommenden Daten in dieser Variablen abgelegt werden. Ein Befehl könnte folgendermaßen aussehen:

```
1) Call        Code1        with Display1, 80
```

Das erste Argument ist ein '**var string'**. Das Display1 Objekt wird benutzt, damit der zurückgegebene String angezeigt werden kann. Sie können auch einen Container verwenden, um zurückgegebene Werte vom Code Objekt zu empfangen. Das zweite Argument gibt die Länge (80 Zeichen) an.

Externer Code in C

DLLs benötigen eine ‘main procedure’, um die Library zu initialisieren (oft macht diese Routine jedoch nicht viel) und eine oder mehrere andere Routinen, welche aufgerufen werden können. Beide, Microsoft und Borland, benutzen das Schlüsselwort "export" um anzuzeigen, daß diese Funktion von anderen Windowsprogrammen außerhalb der DLL aufgerufen werden kann.

Hier ist ein kurzes Beispiel, wie eine DLL in Borland C erstellt wird:

```
#include <windows.h>
int  WINAPI   _export   RoutineABC (int i, double d, char *s)
{
    // put code here for the subroutine
}
int   FAR   PASCAL   LibMain (HANDLE h,WORD w1,
                              WORD w2,LPSTR c)
{
    return 1;
}
```

In Borland C++ fügen Sie **extern "C"** vor der Funktion, die exportiert werden soll, ein.

Hier sind die korrespondierenden Argumenttypen für die C-Sprachen

TestPoint	C/C++
char	signed char
byte	unsigned char
integer	short int
word	unsigned short
long	long
dword	unsigned long
float	float (see note)
double	double
string	char far *
var xxx	xxx far *
xxx []	xxx huge *
hwnd or window	HWND

Achtung: Einige ältere C Compiler übergeben keine **'float'** Argumente. Sie konvertieren diese in 'double'. Dies ist kein Problem in MSC 7.0 oder Borland C 3.1 und später.

Beachten Sie auch, daß Sie 'huge' Pointer für Vektoren und Felder benutzen, da die totale Größe von 64K byte an Daten (das Limit für nicht huge Pointer in Windows 3.1) ist.

Später in diesem Kapitel finden Sie einige C Beispiele.

Externer Code in Borland Pascal

Borland Pascal benutzt das Schlüsselwort **Library** anstelle von ‘program’, um anzuzeigen, daß es sich um eine DLL handelt, die erstellt werden soll. Jede Unterroutine, die von außerhalb der DLL aufgerufen werden soll, muß das Schlüsselwort **export** aufweisen und muß noch einmal in der **exports** Sektion aufgelistet sein:

```
library example;
procedure abc (i : integer; var d : double); export;
begin
    d := i;
end;

exports
    abc index 1;

begin
    { Initialisierungs Code beginnt hier }
end.
```

Hier sind die passenden Datentypen zur Pascal Programmiersprache:

TestPoint		Pascal
char		char
byte		byte
integer		integer
word		word
long		longint
dword		-- none --
float		single
double		double
string		Pchar
	or	array of char[0..n]
var xxx		var
xxx[]		-- siehe Hinweis unten --

Achtung: Pascal kann eine feste Einstellung für Typ und Größe der Elemente eines Vektors n (**array of type[0..n])** bei der Deklaration akzeptieren. Für eine variable Größe können Sie Pascals ‘open array’ **array of type**, ohne eine Dimension verwenden. Dies **muß** von TestPoint als Vektor gefolgt von einem Integer aufgerufen werden (zwei Argumente für das Pascal Argument), wobei der Integerwert die Größe des Vektors minus eins ist.

Borland Pascal beinhaltet die **Strings** Unit, welche es einfach gestaltet, mit 0 terminierten Strings umzugehen (benutzen Sie den Typ **pchar)**.

Windows Argumenttypen

Wenn Sie in Windows eingebaute Funktionen aufrufen, finden Sie häufig spezielle Windows Typen für einige Argumente. Diese korrespondieren zu folgenden Datentypen:

Windows Typ	TestPoint
HANDLE	word
HWND	hwnd
LPSTR	string or var string
LPARAM	dword
WPARAM	word

Beispiele

Beispiel - Aufruf einer Windows Funktion

Erstellen Sie eine Applikation mit einem Schalter, einem Code Objekt und einem Display. Diese Anwendung ruft eine in Windows eingebaute Funktion, um den Verzeichnisnamen für das Windows zurückzugeben, auf (normalerweise C:\WINDOWS).

Die GetWindowsDirectory Funktion ist in dem Windows Programmierhandbuch folgendermaßen definiert:

```
WORD GetWindowsDirectory (LPSTR lpBuffer, int nSize)
```

Benutzen Sie diese Einstellungen für das Code Objekt:

DLL filename:	KERNEL
Subroutine name:	GetWindowsDirectory
Argument types:	var string,integer
Return Type:	word

Erstellen Sie die Action Liste für einen Pushbutton:

```
1) Call        Code1        with  Display1 , 80
```

Jetzt wählen Sie den 'Run' Modus und klicken auf den Schalter. Die Windows 'kernel' Routine gibt Ihnen den Pfad für Ihr Windows Verzeichnis zurück. Da der 'var string' Parameter benutzt wird, wird der zurückgegebene String gleich im Display angezeigt.

Beispiel - Zugriff auf kundenspezifische Hardware I/O Ports

Dieses Beispiel benutzt einen externen Code, um Treiber Funktionen für ein eigenes Interface Board zur Verfügung zu stellen. Nehmen wir an, daß die Hardware eine Kanalnummer als Argument benötigt und einen numerischen Wert als Meßergebnis zurückgibt.
Benutzen Sie folgende Einstellungen für Ihr Code Objekt:

DLL filename:	CUSTOMIO.DLL
Subroutine name:	Measure
Argument types:	integer, var double
Return Type:	word

und diese Action Liste für den Pushbutton:

1) Call Code1 with Data-Entry1 , Display1

und der externe Code, geschrieben in C, würde folgendermaßen aussehen (Borland C++ wurde für dieses Beispiel benutzt, andere Sprachen können ebenfalls benutzt werden):

```
#include  <windows.h>
#define  ADDRESS   0x300
// note: extern "C" ist nur erforderlich für C++
extern  "C"   WORD   WINAPI   _export   Measure
                        (int channel, double *result)
{
    outp (ADDRESS,channel);
    while ((inp(ADDRESS) & 1) == 0) ;  // wait for "done" flag
    char c = inp(ADDRESS+1);
    *result = (double) c * 1.5 + 2.0;
    return 0;
}
int  FAR   PASCAL  LibMain (HANDLE,WORD,WORD,LPSTR)
{
    return 1;          // LibMain ist in allen DLLs erforderlich
}
```

Hier ist das gleiche Beispiel in Borland Pascal:

```
library customio;
procedure Measure (channel : integer; var result : double);
    export;
begin
    Port[$300] := channel;
    while Port[$300] and 1 = 0 do ;
    result := Port[$301] * 1.5 + 2.0;
end;
exports
    Measure index 1;
begin
end.
```

Beispiel - Vektor Arithmetik

Dieses Beispiel benutzt eine externe Routine, um einige Berechnungen von numerischen Vektorwerten durchzuführen. Es ist zwar einfach genug, daß TestPoints eingebaute mathematische Funktionen dieses ebenso erledigen können, aber die Idee dahinter ist lediglich darzustellen, wie Sie Mathematik Funktionen von TestPoint erweitern oder spezielle Hardware wie z.B. ein Coprozessor ansprechen können.

Benutzen Sie diese Einstellungen für das Code Objekt:

DLL filename:	MYMATH.DLL
Subroutine name:	Process
Argument types:	var double[], integer
Return Type:	word

Dies ist eine mögliche Action Liste für diesen Code:

1)	Acquire A/D	A/D1	#samples=1000, rate=10000 Hz, channel(s)=0
2)	Calculate	Length	with v=A/D1
3)	Call	Code1	with A/D1 , Length
4)	Draw Graph	Graph1	with A/D1

'Length' stellt ein Mathematik Objekt mit der Formel **dim(v)-1 dar**. Der 'length' Parameter wird von der Unterroutine benutzt, um festzustellen, wie viele Elemente sich in dem Vektor befinden, indem das Mathematik Objekt die Anzahl der aufgenommenen Werte der 'Aquire' Aktion auswertet.

Nachdem Zeile 3 ausgeführt wurde, werden die Daten des A/D Objekts modifiziert (weil der **var** Parameter benutzt wurde).

Dies ist der externe Code,in C:

```
#include <windows.h>
WORD   WINAPI    _export   Process
                           (double *v, int n)
{
    int i;
    for (i=0;i<=n;i++)
       *v++ = 2.0 * ((*v) * (*v));              // 2 mal v zum Quadrat
    return 0;
}
int far pascal LibMain (HANDLE h,WORD w1,WORD w2,LPSTR c)
{
    return 1;            // LibMain ist in allen DLLs erforderlich
}
```

Und das gleiche Beispiel in Pascal:

```
library mymath;
procedure Process (var v : array of double); export;
var i : integer;
begin
    for i := 0 to High(v) do v[i] := 2.0 * v[i] * v[i];
end;
exports
    process index 1;
begin
end.
```

In diesem Beispiel benutzt Pascal einen "offenen", also nicht dimensionierten Arrayparameter, dessen Dimension in der folgenden Argumentübergabe angegeben werden muß. TestPoint übergibt den Vektor und die Dimension als 2 Argumente, die von Pascal als 1 Argument behandelt werden.

Zugriff auf TestPoint Fenster

Sie können von externem Code aus direkt auf TestPoint Objekt Fenster zugreifen, indem Sie den **hwnd** Argumenttyp verwenden. Dieser Argument Typ erwartet, daß in der Call Aktion der Action Liste ein Objekt übergeben wird. Dann wird das Fenster für dieses Objekt an den externen Code übergeben.

Zwei Beispiele werden mit TestPoint ausgeliefert, die diese Möglichkeit verwenden: DRAW.TST und PAINT.TST

DRAW.TST ruft eine in Windows eingebaute Funktion auf, um in das Picture Objekt zu zeichnen.

PAINT.TST ist etwas komplexer. Es verwendet externen Code, der in C++ geschrieben wurde, um die Handle Ereignisse für das Fenster zu übergeben.

Durch das Programm PAINT.CPP (C++ Quelltext) können Sie Mausereignisse und Fensterverhalten behandeln, um somit Ihr eigenes Objekt mit völlig neuem Design zu erzeugen.

Kundenspezifische Ereignisse und das Code Objekt

TestPoint benutzt ein Ereignis zum Starten der Action Liste. Zum Beispiel wird die Action Liste für den Schalter ausgeführt, sobald auf diesen geklickt wird.

Das Code Objekt stellt eine Möglichkeit zur Verfügung, mit Ihren eigenen Ereignissen eine Action Liste zu starten.

Ereignisse werden in Windows durch Aufruf der Windows Funktion '**PostMessage'**, die den Botschaftstyp und das Fenster, an das die Botschaft gerichtet war, anzeigt.

Jedes Code Objekt in TestPoint hat ein eigenes Fenster, welches Botschaften empfangen kann. Um den "window handle" eines Code Objekts zu bekommen, übergeben Sie ein Argument mit dem speziellen Typ '**Window'** an die Unterroutine Ihrer DLL. Die DLL sollte dann diese Nummer speichern und später bei dem Aufruf der 'PostMessage' Funktion wieder verwenden, wenn ein Ereignis ausgelöst werden soll. Zum Beispiel hier ein C Source Code für eine DLL, die ein Ereignis basierend auf einen Timer, aufruft:

```
#include <windows.h>
HWND hCodeWindow = NULL;
int nTimerId = 0;
void  WINAPI   _export   TimerFunction (HWND h,WORD w,
                                        int n,DWORD time)
{
    if (hCodeWindow  !=  NULL)
        PostMessage (hCodeWindow,WM_USER,0,0L);
}
void  WINAPI   _export   StartEvents (HWND hcw)
{
    hCodeWindow = hcw;
    nTimerId = SetTimer (NULL,0,2000,(FARPROC)TimerFunction);
}
```

```
void WINAPI _export StopEvents()
{
    KillTimer (NULL,nTimerId);
}
```

Die „StartEvents“ Routine übernimmt ein ‘Window handle’ als Argument, speichert diese in der Variable ’hCodeWindow’, und startet den Windows timer. Die ‘StopEvents’ Routine stoppt den Timer.

Hier ist der Aufruf von TestPoint aus:

Code Objekt Settings:

Name:	Start Events
DLL filename:	MYTIMER.DLL
Subroutine name:	StartEvents
Argument types:	window
Return type:	integer

Action Zeile:

1) Call	Start Events

Das ‘**Window’** Argument wird von TestPoint automatisch zur Verfügung gestellt und teilt der DLL mit, wohin Ereignisse zu senden sind.

Wenn der Timer startet (alle 2000 msec in diesem Beispiel), wird die ‘TimerFunction’-Routine aufgerufen, und diese benutzt ‘PostMessage’, um die ‘WM_USER’ Botschaft an das Code Window zu senden.

Die WM_USER Botschaft bringt TestPoint dazu, die Action Liste für das Code Objekt zu starten!

Dieses Beispiel benutzt einen Timer, aber Sie können auch einen Code generieren, der basierend auf Hardware Interrupts oder anderen Bedingungen, die Action Listen startet und somit die Möglichkeiten von TestPoint um ein vielfaches erweitert.

Der letzte Schliff

Kapitel 24.
Fehlersuche (Debugging)

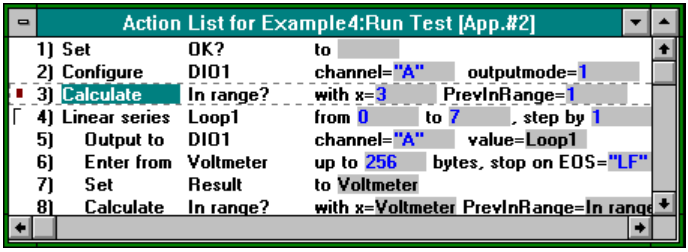
Was in diesem Kapitel behandelt wird:

- Einzelschritt und Unterbrechungspunkte in Action Listen.
- Anschauen von Daten eines Objekts.

Unterbrechungen und Einzelschritt-Modus

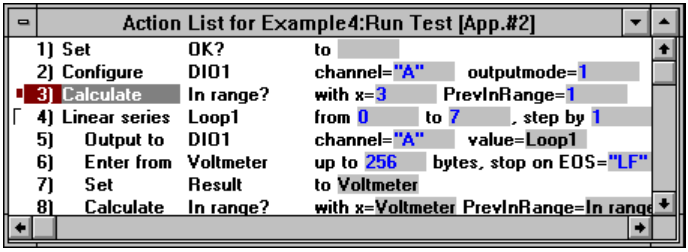
Unterbrechungen (Breakpoints)

Das ‘Debug’ Menü von TestPoint erlaubt Ihnen das Setzen von Haltepunkten (breakpoints), an denen die Programmausführung angehalten wird, damit Sie untersuchen können, welchen Zustand bestimmte Größen besitzen. Um einen „Breakpoint“ zu setzen, wählen Sie die gewünschte Befehlszeile aus der Action Liste aus, klicken auf die Zeilennummer und benutzen das Menükommando DEBUG/SET/RESET BREAKPOINTS oder die Tastenkombination [STRG]+[B].

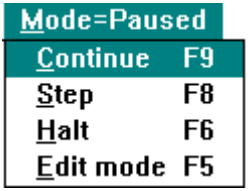


Der rote Punkt neben der Zeilennummer zeigt Ihnen den Haltepunkt an. Zum Löschen des Haltepunktes benutzen Sie aus dem Menü DEBUG den gleichen Befehl. Wollen Sie alle Haltepunkte löschen, wählen Sie unter DEBUG/CLEAR BREAKPOINTS.

Bei Ausführung des Programmes wird das Programm vor dem Breakpoint angehalten und die Zeile unterlegt in der Objektliste dargestellt.



In der Hauptmenüzeile erscheint MODE=PAUSED und Sie können verschiedene Funktionen aus dem Mode Menü auswählen.



Bei Auswahl von CONTINUE wird mit der Programmausführung fortgefahren, bis ein erneuter Haltepunkt auftaucht oder das Programm beendet ist. Bei Auswahl von 'Step' wird nur die nächste Zeile ausgeführt und erneut auf weitere Eingaben gewartet. Die Option HALT bricht das Programm ab, TestPoint bleibt jedoch im RUN Modus. Wählt man den EDIT MODE wird die Abarbeitung des Programms gestoppt und TestPoint kehrt in den Edit Modus zurück.

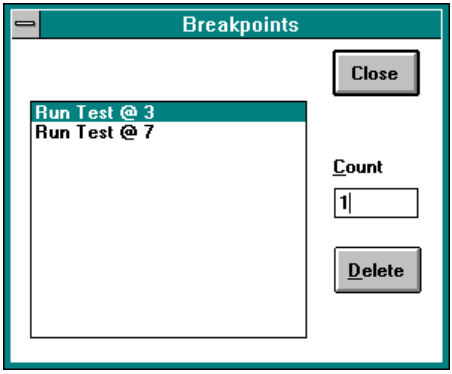
Einzelschritt Modus

Das Menü DEBUG erlaubt auch die Einstellung eines SINGLE STEP MODE, bei dem die Programmausführung nach jedem Befehl (und in jeder Zeile) angehalten wird. Die Option CONTINUE erlaubt Ihnen dennoch, den Einzelschritt-Modus aufzuheben und das Programm schnell zu beenden.

Anzahl der Unterbrechungspunkte

Normalerweise stoppt ein Haltepunkt die Ausführung des Programms, wenn dieser das erste mal erreicht wird. Um eine bestimmte Anzahl von Durchläufen abzuwarten (vielleicht in der Schleife), bevor das Programm angehalten wird, müssen Sie die Anzahl vorher bestimmen.

Benutzen Sie das Menükommando DEBUG/VIEW BREAKPOINTS, um eine Liste von Breakpoints anzuzeigen.



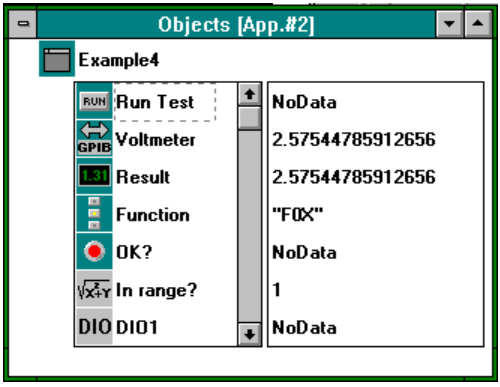
In diesem Fenster können Sie Breakpoints löschen oder die Anzahl verändern. Wenn Sie den Zähler (Count) auf 3 setzen, bedeutet dies, daß beim 3. Erreichen dieser Zeile das Programm angehalten wird.

Nachdem der Haltepunkt einmal erreicht wurde, wird der ‘Count’-Wert auf 1 zurückgesetzt. Die Programmausführung wird danach erst wieder nach dem dritten Durchlauf unterbrochen.

Datenansicht

Wurde die Programmausführung unterbrochen, oder ist eine Action Liste abgearbeitet, ist es sehr oft sinnvoll, die Daten, die ein Objekt nun enthält, zu sehen.

Benutzen Sie das Menükommando VIEW/DATA, um die Object Liste um die Datenansicht zu erweitern. Sie muß vorher als aktives Fenster geschaltet werden.



Das Fenster zeigt den Datentyp und die Daten des Objekts. Sie können das Fenster vergrößern, um mehr Daten zu sehen. Treten lange Vektoren oder Listen als Datentyp auf, können diese meist nicht vollständig angezeigt werden. Für die Fehlersuche in der Anwendung sind die dargestellten Werte zumeist ausreichend.

Für den Fall, daß Sie alle Daten eines langen Vektors sehen möchten, fügen Sie ein File Objekt in Ihre Anwendung ein und schreiben Sie die Daten auf Ihre Festplatte, um diese mit einem Text Editor ansehen können.

Kapitel 25. Weitergeben von Anwendungen – Runtimes

Wenn Sie eine komplette TestPoint Anwendung erstellt haben, möchten Sie vielleicht ein separates Symbol in Windows installieren, um diese Anwendung getrennt vom Editor laufen lassen zu können, oder Sie möchten vielleicht eine Diskette mit Ihrer Anwendung und einem Setup Programm erstellen, um es weiterzugeben.

Der komplette Software Vertrag ist auf Ihrem TestPoint Original Paket abgedruckt. Grundsätzlich ist der TestPoint Editor nur für einen Anwender registriert, die Anwendungen, die Sie jedoch damit erstellen, können beliebig ohne weitere Lizenzgebühren verteilt werden.

Erstellen eines Symbols für Ihre Anwendung

Sie können sehr einfach ein Symbol im Programm-Manager anlegen, das Ihre Runtime Anwendung startet.

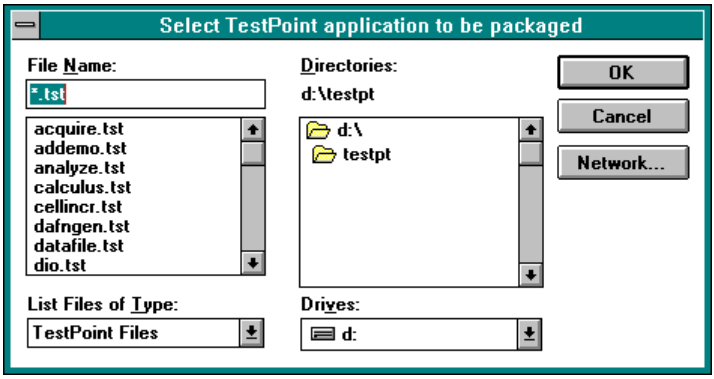
Sie müssen als erstes die komplette Anwendung auf der Festplatte speichern. Wählen Sie dann das Menükommando 'Utilities' 'Add Runtime Icon'. Es erscheint nun ein Fenster, in dem Sie die Anwendungsdatei (*.TST) auswählen können. Sie werden jetzt nach der gewünschten Bezeichnung dafür gefragt.

Das Symbol wird erstellt und in Ihre TestPoint Programmgruppe eingefügt. Wollen Sie es in einer anderen Programmgruppe plazieren, so können sie dies durch Ziehen in die gewünschte Gruppe erreichen.

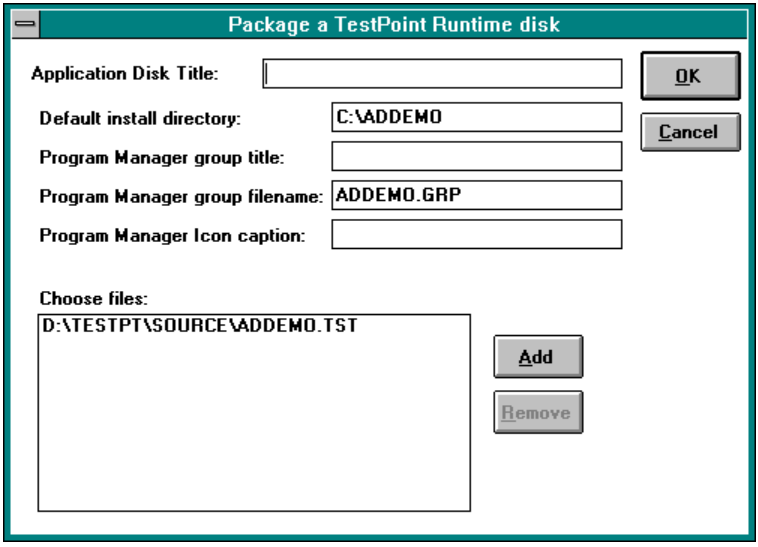
Hinweis: Der Programm-Manager in Windows 3.x begrenzt die Anzahl der Symbole, die Sie in einer einzigen Gruppe plazieren können. Die Grenze hängt von der Anzahl der Farben ab, die Ihre Grafik unterstützt. Beispielsweise unterstützt die 24 Bit Farbeinstellung nur 19 Symbole pro Gruppe. Überschreiten Sie dieses Limit, wird kein weiteres Symbol erzeugt. Sie bekommen aber auch keine Fehlermeldung.

Erstellen einer Runtime Diskette

Nachdem die komplette Anwendung auf der Festplatte gespeichert ist, nutzen Sie das Menükommando ‘UTILITIES/MAKE RUNTIME DISK. Es erscheint nun ein Fenster, welches die Anwendungsdatei (*.TST) enthält.



Wählen Sie die gewünschte Datei aus. Es wird ein weiteres Fenster geöffnet, in welchem Sie die Installationsoptionen anwählen.



Geben Sie einen Titel für Ihre Anwendungsdiskette ein. Dieser Titel erscheint, wenn der Benutzer ‘Setup’ startet. Der Titel wird auch gleich noch automatisch in andere leere Felder kopiert. Sie können auch ein anderes Installationsverzeichnis angeben. Die ‘Programm Manager’ Option läßt Sie den Titel, die Datei und die Symbolüberschrift für die neue Anwendung angeben.

Falls Ihre Applikation weitere Dateien benötigt, benutzen Sie die ‘Add’ Option, um diese auszuwählen. Jede Runtime Disk installiert eine Anwendung mit einem Symbol, Sie können jedoch beliebig viele Dateien darauf kopieren.

Die TestPoint Programmdateien werden automatisch kopiert, so daß Sie diese nicht extra auswählen müssen.

Zum Schluß werden Sie noch nach dem Ziellaufwerk für die 'Runtime' Version gefragt. Sie können die Runtime direkt auf eine formatierte Diskette erstellen oder in ein neues Verzeichnis Ihrer Festplatte und später auf Diskette kopieren.

Wenn Sie auf 'OK' klicken, werden die notwendige TestPoint Unterstützungsbibliotheken komprimiert und kopiert.

Wenn Sie die Runtime Dateien auf Festplatte kopieren, und erst später auf eine Floppy auslagern, müssen Sie sicherstellen, daß, wenn das Fassungsvermögen einer Diskette nicht ausreicht, die Dateien SETUP.EXE und SETUP.INF auf der Diskette Nr. 1 bereitgestellt werden.

Das erstellte Setup Programm läuft unter Windows, installiert automatisch alle Dateien und fragt den Benutzer auch nach Verzeichnissen, usw.

Hinzufügen von Dateien auf einer Runtime Diskette

Im PACKAGE RUNTIME DISK Fenster können Sie den „ADD" Schalter verwenden, um weitere Dateien auf Ihre Diskette zu kopieren. Dies können weitere TestPoint Anwendungen sein, zugeordnete Datendateien, Bilddateien, usw.

Bilddateien

Wenn Sie das Picture Objekt von TestPoint verwendet haben, sollten Sie die Bilddateien auf Ihre Runtime Diskette nehmen. Das Picture Objekt benötigt diese Dateien, weil sie beim Start der Anwendung von der Festplatte geladen werden.

VBX Custom Control Dateien

Haben Sie das VBX Objekt von TestPoint verwendet, sollten Sie die VBX Dateien (.VBX) auf Ihre Runtime Diskette nehmen.

Pfad der Dateien

Sie können jede beliebige Datei, aus jedem Verzeichnis Ihres Computers auf Ihre Runtime Disk nehmen. Wenn die Runtime Version installiert wird, werden die Dateien **nur in das ausgewählte Verzeichnis und dessen Unterverzeichnis kopiert.**

Wenn die Datei, die Sie aufnehmen, im gleichen Verzeichnis wie Ihre Anwendung steht, wird sie **im** gewählten Installationsverzeichnis installiert.

Wenn die Datei, die Sie aufnehmen, im einem Unterverzeichnis Ihrer Anwendung steht, wird sie **unter** dem gewählten Installations-verzeichnis installiert.

Befindet sich die Datei , die Sie aufnehmen, in irgendeinem anderen Verzeichnis (z. B. \WINDOWS), wird sie im gewählten Installations-verzeichnis installiert. **Beachten Sie, daß sich in diesem Fall Ihre Runtime Version nicht im gleichen Verzeichnis wie Ihre Entwicklungsversion befindet.** Im allgemeinen ist dies kein Problem. TestPoint sucht Dateien sowohl im Runtime Verzeichnis als auch im angegebenen Verzeichnis. Somit werden Bilder und externer Code trotzdem gefunden.

Testen Sie Ihr Runtime

Um sicherzustellen, daß Ihre Anwendung läuft und Sie keine Dateien vergessen haben, sollten Sie Ihre Runtime-Version vorzugsweise nochmals auf einen anderen Computer installieren und testen.

Wichtige Dateien für Runtimes

TestPoint findet selbstständig heraus, welche Programmdateien für Ihre Runtime Anwendung erforderlich sind. Wenn Sie eine A/D Karte verwenden, werden alle verfügbaren A/D Treiber kopiert. Dies erlaubt dem Endanwender, die A/D-Karten auszutauschen, ohne das Programm zu ändern.

Auswahl eines Startbildschirms

Standardmäßig wird, während Ihre Anwendung geladen wird, dieses Bild angezeigt:

Sie können ein eigenes Startbild mit einbinden, das Ihr Firmenlogo, Text oder Bild enthält. Folgende Schritte sind erforderlich:

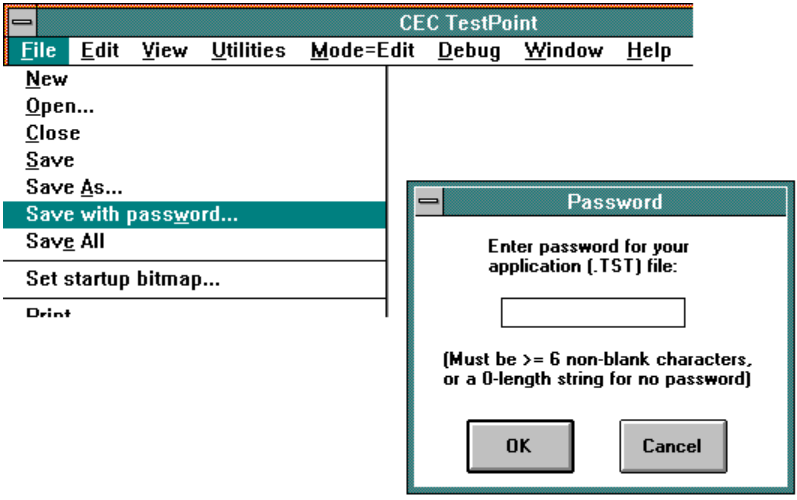
- Laden Sie Ihre Anwendung in den TestPoint Editor.
- Verwenden Sie das FILE/SET STARTUP BITMAP Menükommando, um eine Bitmapdatei auszuwählen. Diese Datei sollte nicht mehr als 200 Punkte breit sein. Maximal sollte es 640 x 480 Punkte breit sein, damit es auf allen Bildschirmen angezeigt werden kann.
- Speichern Sie die Anwendung.

Zwei Beispiel Bitmaps werden mit TestPoint ausgeliefert: RUNTIME.BMP und RUNTIMEU.BMP.

Unsichtbar machen von Quellcode

Wenn Sie eine Anwendung entwickeln, die Sie auch verkaufen möchten, wollen Sie wahrscheinlich Einzelheiten der Anwendung vor Anwendern verbergen, die ebenfalls TestPoint verwenden.

Sie können dies machen, indem Sie Ihre Anwendung mit einem Paßwort versehen. Benutzen Sie dazu das Menükommando **FILE/SAVE WITH PASSWORD**:



Das Paßwort wird in der TESTPT.INI Datei Ihres Computers abgelegt, so daß Sie nicht bei jedem Start das Paßwort angeben müssen. Das Paßwort ist aber nicht in der Runtimeversion verfügbar. Somit kann der Anwender diese Datei nicht öffnen.

Bedenken Sie, daß ein Vorteil der TestPoint Entwicklungsumgebung das leichte Verändern von Anwendungen ist. Ihre Kunden können dies machen, wenn Sie die Anwendung ungesperrt lassen oder Ihnen das Paßwort geben.

Mehr als ein Runtime Symbol

Das MAKE RUNTIME DISK Menükommando erzeugt eine Runtime Diskette, die Ihre Anwendung installiert und legt im Windows Programm-Manager eine neue Programmgruppe an.

Wünschen Sie mehr als eine Anwendung auf einer Diskette, klicken Sie den „ADD“ Schalter an, um weitere .TST Dateien auszuwählen.

Nachdem die Diskette erstellt worden ist, können Sie die Datei SETUP.INF bearbeiten, die auf der ersten Diskette abgelegt ist. Sie können diese ASCII Datei mit dem Notepad Editor oder einem DOS Editor bearbeiten. Am Ende der Datei finden Sie diesen Abschnitt:

```
[PM Info]
TPRUN.EXE YOURAPP.TST, Your title, TPRUN.EXE, 0
```

Jede Zeile in diesem Abschnitt der Datei erzeugt ein Symbol im Programm-Manager. Um weitere Symbole mit .TST Dateien zu erzeugen, duplizieren Sie diese Zeilen und ändern Sie den Namen der .TST Datei entsprechend. Optional kann der Titel für das Symbol verändert werden. Kommas trennen die Abschnitte der Zeile voneinander. Hier ist ein Beispiel:

```
[PM Info]
TPRUN.EXE YOURAPP.TST, Your title, TPRUN.EXE, 0
TPRUN.EXE APP2.TST, Other title, TPRUN.EXE, 0
```

Kapitel 26. Weitergeben von benutzerdefinierten Objekten

Das Benutzen der Libraries von TestPoint

TestPoint wird mit einer großen Anzahl von GPIB Instrumenten- und anderen Funktions-Bibliotheken ausgeliefert. Einige werden automatisch installiert, andere können selektiv von den Librarydisketten installiert werden.

Zum Beispiel: Es gibt GPIB Instrumente Bibliotheken wie das Keithley 2001 Multimeter oder das LeCroy 93XX Oszilloskop usw. Andererseits sind auch Bibliotheken vorhanden, die nützliche Objekte für die Datenanalyse beinhalten und andere für gebräuchliche Datenerfassungsfunktionen, wie z. B. Umwandlung von Spannungen eines Thermoelements in Temperaturwerte (Thermoelemente-Linearisierung).

Einige dieser Dateien werden immer installiert:

THERMO.TST	Thermocouple linearization
FKEY.TST	Action lists on function keys
PID.TST	PID closed loop control
ENGR.TST	Engineering (metric suffix) units

Das sind alle benutzerdefinierte Objekte, die mit TestPoint erstellt wurden.

Sie können ebenfalls solche Objekte erstellen und sie in anderen Anwendungen benutzen oder sie an andere TestPoint Benutzer weitergeben oder verkaufen.

Anwenden einer Bibliothek

1. Benutzen Sie das FILE/OPEN Menükommando, um eine Bibliothek zu laden (im '**Library'** Unterverzeichnis Ihres TestPoint Verzeichnisses auf der Festplatte). Generell beinhaltet eine Library ein oder mehrere benutzerdefinierte Objekte und ein Textobjekt, das erklärt, was die Bibliothek beinhaltet. Einige der benutzerdefinierten Objekte können auch Kommentare enthalten (Benutzen Sie das VIEW/COMMENTS Menükommando).

2. Selektieren Sie das Library Objektsymbol in der Objektliste und benutzen Sie das EDIT/COPY Menükommando, um es in die Windows Zwischenablage zu kopieren.

3. Benutzen Sie das FILE/CLOSE Menükommando, um die Library Datei zu schließen.

4. Öffnen Sie nun Ihr Anwendungsprogramm oder benutzen Sie das FILE/NEW Menükommando, um eine neue Anwendung zu erstellen.

5. Benutzen Sie das EDIT/PASTE Menükommando, um das Objekt aus der Zwischenablage in Ihre Anwendung zu kopieren.

6. Immer, wenn Sie nun dieses Objekt von der Objektliste in die Action Liste ziehen (z. B. für einen Druckschalter), bekommen Sie ein Menü der verfügbaren Funktionen dieses Objekts angezeigt.

Was ist in einer Bibliothek

Library Objekte sind benutzerdefinierte Objekte, erstellt in TestPoint.

Einige sind gesperrt, so daß kein rechter Pfeil neben dem Symbol in der Objektliste erscheint. Um ein gesperrtes Objekt zu entsperren, benutzen Sie das UTILITIES/LOCK/UNLOCK USER-DEFINED Menükommando. Sie müssen nun das Paßwort eingeben, mit dem das Objekt gesperrt wurde. War das Paßwort richtig, so erscheint nun neben dem Objektsymbol in der Objektliste ein Pfeil nach rechts. Sobald Sie diesen anklicken, sehen Sie, wie die 'Library' aufgebaut ist.

Viele Bibliotheksobjekte beinhalten Druckschalter, Selectoren, Dateneingabefelder und 'Action' Objekte.

- Druckschalter werden für Aktionen benutzt, die keine Parameter brauchen.
- Selectoren werden für Aktionen benutzt, die nur einen Parameter benötigen, der eine Auswahlliste ist.
- Dateneingabefelder werden für Aktionen benutzt, die einen einzelnen Parameter benötigen, der eine beliebige Zahl oder ein String sein kann.
- Action' Objekte werden für Aktionen benutzt, die mehrere Parameter benötigen oder eine benutzerdefinierte Eingabezeile ermöglichen sollen.

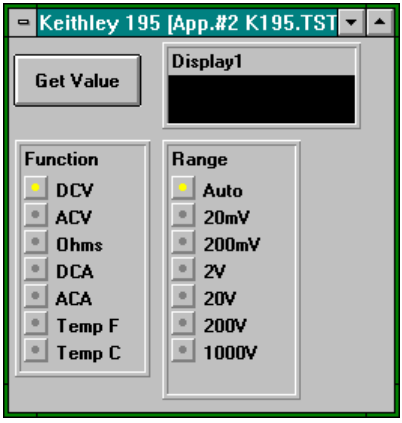
Library Objekte können auch andere Objekte, wie z. B. Mathematik-Objekte enthalten, die mathematische Operationen ausführen; GPIB Objekte, um Geräte zu bedienen; Schleifen.....

Erstellen von eigenen Libraries

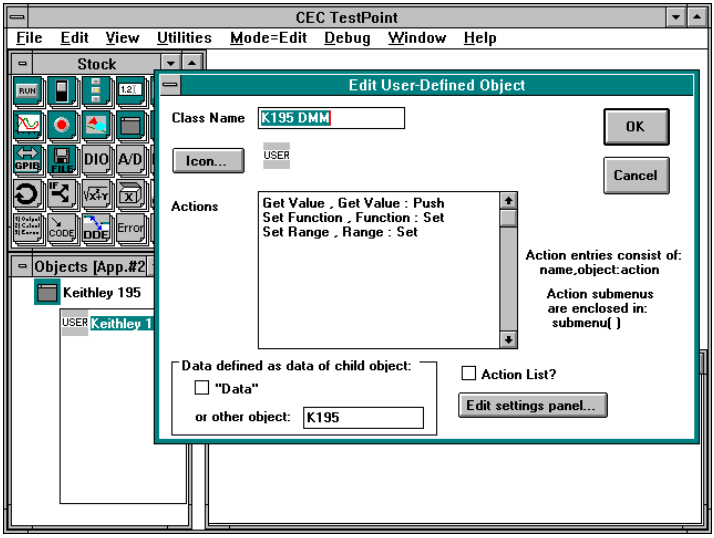
GPIB Instrument Libraries

Der Anfang einer Instrumentenbibliothek ist das Erstellen eines Fensters, welches die interaktive Kontrolle eines Gerätes erlaubt. Starten Sie mit einem GPIB Objekt. Ziehen Sie nun Dateneingabefelder, Selectoren, Kippschalter usw. auf das Fenster, um verschiedene Optionen zu setzen und lesen Sie Daten vom Gerät. Testen Sie nun Ihre Anwendung. Speichern Sie die Anwendung in eine separate Datei für mögliche Änderungen Ihrer Library.

Hier ist ein einfaches Beispiel für solch ein Fenster:



Selektieren Sie jetzt das Fenster durch Anklicken in der Objektliste. Benutzen Sie das ‘Utilities’ ‘Package as user-defined’ Menükommando und wählen Sie den Klassennamen (Der Name sollte nicht mit einer Zahl enden, da das Objekt, sobald es in die Objektliste gezogen wird, automatisch immer mit einer Nummer versehen wird). Sie können den Namen selbst festlegen.



Es ist auch möglich, die Namen von Aktionen zu verändern. Schauen Sie sich die ‘Actions’, die für Ihre Bibliothek erstellt wurden, an. Wäre es sinnvoll, einige Aktionen zu einer einzelnen Aktion mit mehreren Parametern zu kombinieren? Wird der Anwender bestimmte Gruppen zur selben Zeit setzen wollen? Wenn ja, dann erstellen Sie eine neue Aktion mit dem ‘Action’ Objekt.. So fassen Sie mehrere Aktionen in einer zusammen und können so in einer Aktion mehrere Parameter setzen. Nun fügen Sie noch ein Einstellungsfenster hinzu. Editieren Sie ein neues ‘panel’ Objekt aus dem ‘stock’ in Ihr benutzerdefiniertes Objekt. Geben Sie Ihm den Namen ‘Settings’. Benutzen Sie das ‘View’ ‘Panel’ Menükommando, um es zu öffnen. Fügen Sie ein Dateneingabeobjekt mit dem Namen "GPIB Address" und einen Kippschalter mit dem Namen "Demo mode" (setzen Sie den ‘Setting style’ auf checkbox) hinzu. Erstellen Sie die Action Listen für diese Objekte so, daß damit die Einstellungen für Ihr verwendetes GPIB Objekt verändert werden (Sie können in einer der mitgelieferten Bibliotheken die gleiche Vorgehensweise wiederfinden).

Falls gewünscht, können Sie im Windows-Paintbrush eine Bitmap Datei für Ihr eigenes Symbol erstellen (24 x 24 bits, 16 Farben). Um Ihr

eigenes ‘user-defined Icon’ zu verwenden, ändern Sie die Einstellung im benutzerdefinierten Object edit-Fenster.

Mathematik Funktionen und andere Libraries

In TestPoints Mathematik-Objekt sind viele leistungsstarke Funktionen eingebaut. Es ist flexibel, um komplizierte Ausdrücke auszuwerten. Für einige Anwendungen ist es jedoch notwendig, Berechnungen und Auswertungen von Formeln in mehreren Schritten auszuführen.

Beispiel: Ein Zwischenergebnis muß berechnet werden, das an mehreren Stellen der zweiten Formel wieder verwendet wird, wie der nachstehende Ausdruck:

```
first = x^2 + 3*y + ramp(10)
result = sin(first) + rndNormal(0,0.3,dim(first))
```

Würde man versuchen, diese beiden Ausdrücke in eine Formel zu verwandeln, so müßte man doppelt soviel eintippen und die Berechnung würde länger dauern, da der erste Ausdruck zweimal berechnet wird.

Das Benutzen von zwei Mathematik-Objekten funktioniert gut, bedeutet jedoch auch, daß Sie jedesmal zwei Befehlszeilen benötigen, wenn Sie diese Berechnung durchführen wollen.

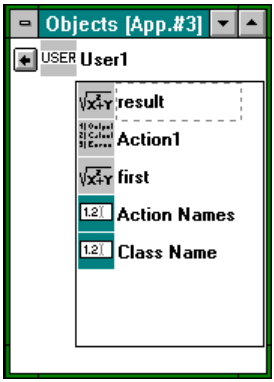
Um ein einzelnes Objekt zur Verfügung zu stellen, das auch noch weit komplexere Ausdrücke mit Schleifen und Bedingungen enthalten kann, packen Sie diese Berechnungen in ein benutzerdefiniertes Objekt wie eine Bibliothek.

Benutzen Sie ein Action Objekt, um die notwendige Aktion zu definieren:

```
    Action Parameter Phrase:        x=%1, y=%2

Action Object Action Liste:
1)  Calculate       first           with x=%1, y=%2
2)  Calculate       result          with first=first
```

Das benutzerdefinierte Objekt beinhaltet:



(Da das erste Objekt ‘result’ ist, hat das User1-Objekt auch dessen Daten aus der Berechnung).

Das Objekt kann nun als eine einzige einfache Zeile in Action Listen verwendet werden.

1) Calculate User1 with x=A/D1, y=4

TestPoint Referenz

Objekt Referenz

Action Objekt 

Das Action Objekt stellt eine Möglichkeit zur Verfügung, neue Typen von Aktion Zeilen zu definieren und sie in eine beliebige Action List einzubinden (ähnlich einer Subroutine in einer konventionellen Programmiersprache).

Action Objekte sind immer dann hilfreich, wenn eine gleiche Anzahl von Aktionen in verschiedenen Teilen der Anwendung (Programs) benötigt wird. Action Objekte sind ebenfalls hilfreich bei der Definition der Aktionen von benutzerdefinierten Objekten. (siehe User-Defined Objekt sowohl im Kapitel "Using TestPoint" als auch im Referenz Teil).

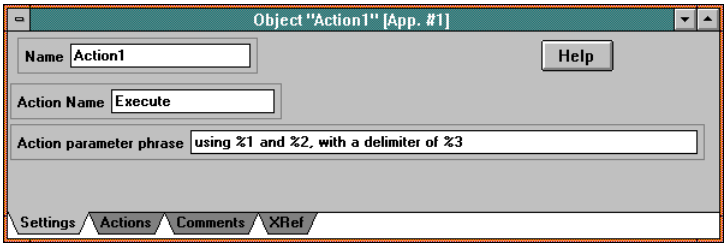
Benutzerschnittstelle - keine

Aktionen - eine Aktion, abhängig von den Einstellungen (Settings)

Hinweis: Um eine Einstellung dieses Objektes in einer Action List eines anderen Objektes zu verändern, ziehen Sie diese Einstellung aus dem Settings-Fenster in die Action List des verändernden Objektes.

Daten - keine

Einstellungen (Settings)



Action Name - der Name, der in den Action List Zeilen später erscheint, wenn dieses Objekt benutzt wird. Dies kann ein beliebiger Text sein. Wenn das Action Objekt in die Action List übernommen wird, ist dieser Text zu Beginn dieser Zeile zu sehen.

Beispiel:

Execute Action1

Action Parameter Phrase - diese Phrase erscheint in der Action List Zeile direkt nach dem Action Name. Die Phrase kann Parameter enthalten, die durch das Prozentzeichen (%) eingeleitet werden, gefolgt von einer Zahl, beginnend bei 1.

Beispiel:

using %1 and %2, with a delimiter of %3

Wenn das Objekt in dieser Form definiert ist, werden in der Action List Zeile Freiräume für die Parameter sichtbar:

Zusätzlich kann einem Parameterfeld eine Liste von Auswahl-möglichkeiten in geschweiften Klammern folgen. Die einzelnen Auswahlmöglichkeiten werden durch ein Komma getrennt. Diese Auswahlliste wird in der Action List als ”Dropdown”-Liste erscheinen, aus der die vorgegebenen Parameter ausgewählt werden können. Weiterhin kann ein Vorgabewert in eckigen Klammern definiert werden (hier A).

Beispiel:

```
with %1 and a choice of %2{A,B,C}[A]
```

erscheint dann als:

Action List

Die Action List wird ausgeführt, wenn das Objekt aus einer anderen Action List aufgerufen wird. Mit anderen Worten: Die Action List des Action Objektes definiert die Aktionen, die ausgeführt werden, wenn das Action Objekt benutzt wird.

In dieser Action List können die Action Parameter benutzt werden, indem ein Prozent-Zeichen und die Nummer des Parameters in die Freiräume der Action List Zeilen eingetragen werden.

Hier ist ein Beispiel für einen Action List, die eine Berechnung auf Basis der übergebenen Parameter durchführt und dann das Ergebnis mit Hilfe eines ‘Display’ Objektes darstellt:

```
1) Calculate    Math1       with x=%1  y=%2
2) Set          Display1    to Math1
```

Beispiel

In diesem Beispiel wird das Action Objekt ‘Adjust’ verwendet, um einen Meßwert mit einem Korrekturwert, unter Verwendung des ‘Math’ Objektes, zu verrechnen. Der Korrekturwert, der benutzt wird, ist von einer Schalterstellung abhängig, so daß zusätzlich ein ‘Conditional’ (Bedingungs-) Objekt zur Anwendung kommt. Das Ergebnis wird in einem ‘Display’ -Object angezeigt. Die Einstellungen (‘Settings’) von ‘Adjust’ sind:

```
Action name:  Do
Action Parameter Phrase:  input=%1
```

und dessen Action List:

```
1)  If/Then/Else    Condition1    with option=Switch1
2)     Calculate    Math1        with x=%1  factor=3.4
3)     Set          Display1     to Math1
4)  Else if not     Condition1
5)     Calculate    Math2        with x=%1  y=5.7  z=2
6)     Set          Display1     to Math2
7)  End if          Condition1
```

‘Adjust’ kann nun in einer Action List wie dieser verwendet werden:

```
1)  Input from      GPIB1        up to 256 bytes
2)  Do              Adjust       input=GPIB1
```

In derselben Anwendung kann das Objekt ‘Adjust’ auch in anderen Action Lists benutzt werden. Dort können andere Daten übergeben werden. Beispielsweise in dieser Action List:

```
1)  Sample          A/D1         once, channel=0
2)  Do              Adjust       input=A/D1
```

Hier werden die Daten des ‘A/D’ Objects korrigiert und angezeigt.

Action und User-Defined Das Action Objekt wird immer dann benutzt, wenn gleiche Aktionen in verschiedenen Action Lists, möglicherweise mit unterschiedlichen Daten (als Parameter) ausgeführt werden müssen. Sehr nützlich ist es, Action Objects im Zusammenhang mit benutzerdefinierten Objekten (User defined Objects) zu verwenden. Ein oder mehrere Action Objects können in benutzerdefinierten Objekten komplexe, mehrzeilige Aktionen bereitstellen. Zusammen mit benutzerdefinierten Objekten und Action Objects können mehrere Objekte und deren Aktionen in Gruppen zusammengefaßt werden und sind somit durch eine einziges 'Icon' darstellbar. Dadurch werden die Anwendungen einfach und gut überschaubar. Beispiele zu diesem Thema sind in den Kapiteln über benutzerdefinierte Objekte und große Anwendungen im "TestPoint Benutzerhandbuch" dargestellt.

Analog/Digital (A/D) Objekt 

Das ‘A/D’ Objekt greift auf PC-Einsteckkarten zur analogen Eingabe von Daten zu. Verschiedene Hersteller von A/D-Karten werden ünterstützt (siehe Kapitel ”Hardware Konfiguration”, die ‘README’ Icons in der TestPoint Programmgruppe oder ‘DOC’ Dateien mit Details über die speiziellen A/D-Karten). Es werden maximal 4 A/D-Karten gleichzeitig im PC unterstützt.

Benutzerschnittstelle - keine

Aktionen

Acquire A/D - Erfaßt die angegebene Anzahl von Meßwerten bei gegebener Abtastrate von einem oder mehreren Kanälen. Die Aktion wird vollständig abgearbeitet, bevor die nächste Zeile der Action List ausgeführt wird, z. B. in der Reihenfolge Acquire A/D und dann Draw Graph. Die Parameter der Aktion sind:

rate - die Abtastrate in Hertz. Die einzige Begrenzung ist die mögliche Abtastrate der A/D-Karte. Für bestimmte Karten kann das zwischen 0.001 und 100 000 Hz liegen. Die Abtastrate bezieht sich auf einen Kanal. Die Grenzen der A/D-Karten liegen bei der Summenabtastrate (rate * Anzahl der Kanäle). Der Spezialwert **external** gibt an, daß der Takt für die Messung extern (nicht auf der A/D-Karte) generiert wird (Hardware clocking).

channel(s) - der Kanal oder die Kanäle, die abgetastet werden sollen mit optionalen Verstärkungsfaktoren. Eine einzelne Zahl kann eingegeben werden, um einen Kanal abzutasten. Die Kanalnummern beginnen mit 0. Sollen mehrere Kanäle benutzt werden, werden die einzelnen Zahlenwerte durch Kommata getrennt eingegeben. Zusätzlich können kanalweise Verstärkungsfaktoren angegeben werden, indem hinter der Kanalnummer in Klammern die Faktoren definiert werden. Hier einige Beispiele für gültige Kanalangaben: ”0”, ”0,1”, ”0,2(10),4,5(100)”. Beachten Sie, das einige A/D-Karten diese Kanal/Verstärkungslisten einschränken. Ebenfalls werden bestimmte Faktoren (je nach Karte unterschiedlich) angeboten. Zwischenwerte werden dabei nicht akzeptiert. Der Kanal Parameter kann auch ein Vektor mit Kanalnummern oder eine Liste, bestehend aus Kanalnummern und Verstärkungsfaktoren sein, die von einem anderen Objekt übergeben wird.

Start A/D - Beginnt die Erfassung eines oder mehrerer Kanäle von A/D-Wandlerkarten bei einer angegebenen Erfassungsrate. Die Erfassung wird im Hintergrund betrieben, während im Vordergrund weiter bearbeitet werden kann. Die Parameter der Aktion sind:

 # samples - die gesamte Anzahl der Meßwerte, die pro Kanal eingelesen werden sollen, oder das Schlüsselwort **continous,** um die Messung zu aktivieren und erst dann zu stoppen, wenn die Aktion Stop A/D verwendet wird. Die Zahl kann zwischen 1 und 2E9 liegen.

event after ... samples - gibt an, wie viele Meßwerte zwischen zwei A/D Ereignissen (Ausführungen der A/D Action List) erfaßt werden sollen. Für eine langsame Erfassung ist es möglich, mit einem Wert von 1 auf jeden Meßwert individuell in der Action List des A/D Objektes zu reagieren. Für schnelle Erfassungen ist es zweckmäßiger, die Daten nach der Messung zu verarbeiten. Der Spezialwert **all** wird verwendet, um ein Ereignis auszulösen, wenn alle Meßwerte erfaßt worden sind. Dies ist nicht möglich, wenn Sie das Schlüsselwort **continous** bei der Anzahl der Meßwerte angegeben haben.

Stop A/D - Stoppt die aktive Hintergrund Erfassung, die durch Start A/D aufgerufen wurde.

Sample A/D - Nimmt einen einzelnen Meßwert von einem oder mehreren Kanälen. Die Kanalliste hat die gleiche Form wie die der Start A/D Aktion. Die Datenwerte des A/D Objektes werden auf die Ergebnisse der Erfassungsaktion gesetzt (ein Zahlenwert oder eine Liste von Zahlen). Diese Aktion stoppt ebenfalls die Hintergrunderfassung.

Set A/D gain - Setzt die Verstärkungsfaktoren, die von den Erfassungs- und Start Aktionen verwendet werden, wenn in der Kanalliste keine Verstärkungen angegeben sind.

Set A/D trigger immediate - Setzt den Trigger Modus, der für die nächste Start A/D Aktion verwendet wird. Immediate bedeutet, daß die Erfassung gestartet wird, sobald die Aktionszeile ausgeführt wird.

Set A/D trigger digital - Setzt den Trigger auf den externen digitalen Hardware Eingang. Die Messung wird erst dann ausgeführt, wenn das externe Triggersignal anliegt.

Set A/D trigger analog - Setzt den Trigger auf einen der Analogen Eingangskanäle. Der Triggerpegel, -polarität, -flanke und -hysterese können angegeben werden. Die Abtastung beginnt erst, wenn der

angegebene Eingangskanal die vorgegebene Triggerspannung hat. Einige A/D-Karten unterstützen die analoge Triggerung nicht. WICHTIGER HINWEIS: Einige A/D-Karten verwenden einen softwaremäßigen Analogtrigger und halten, während sie auf das Triggerereignis warten, alle weiteren Prozesse an.

Set A/D burst mode - Die Erfassung kann im ”Burst”- oder ”Non-Burst”-Modus stattfinden. Der Non-Burst Modus ist voreingestellt. Der Burst-Modus ist nur von Bedeutung, wenn mehrere Kanäle gesampelt werden. Im Nicht-Burst-Modus werden die Kanäle in gleichen zeitlichen Intervallen gesampelt. Im Burst Modus ist der Zeitversatz der einzelnen Kanäle so klein wie möglich (analog einer Sample-and-hold Schaltung). Nur einige A/D-Karten unterstützen die Anwendung des Burst Modus.

Convert to Volts - Konvertiert das binäre Rohdatenformat der A/D Daten, , das als ein **string** Datentyp gegeben ist, in Spannungen der Einheit Volt. Dabei muß die Kanalliste angegeben werden, die beim Aufnehmen der Daten vorlag. Diese Aktion wird zusammen mit dem "Raw data" Setting des Objektes verwendet. Die Daten sollten im binären Rohformat abgespeichert und später in Spannungen umgerechnet werden. Weitere Informationen dazu bei den unten dargestellten Beispielen.

Other A/D command - Hiermit hat der Programmierer Zugang zu herstellerspezifischen Kommandos für erweiterte A/D Features. Diese Aktion benötigt ein Schlüsselwort und einen zugehörigen Wert. Die möglichen Schlüsselwörter hängen von dem Modell des verwendeten A/D-Boards ab, das verwendet wird. Beispielsweise ist es bei einigen Keithley/Metrabyte Boards möglich, das **pretrigger** Schlüsselwort dazu zu benutzen, daß das Datensammeln schon vor dem Hardware Trigger möglich ist. Genauere Informationen dazu in den Beispielen weiter unten.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

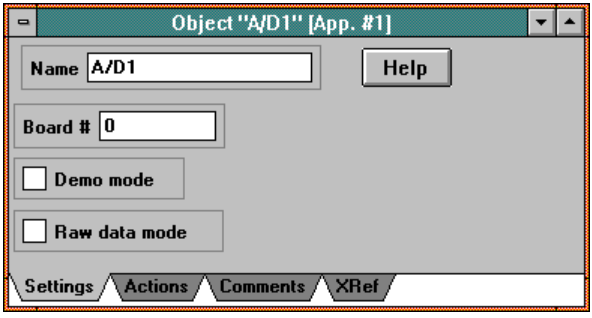
Der Daten- Wert des A/D Objekts ist das Resultat der letzten analogen Eingabeoperation.

Im Fall einer "Sample A/D" Aktion sind diese Daten eine einzelne Zahl oder eine Liste von Zahlen (falls mehrere Kanäle aufgezeichnet wurden).
Im Falle einer "Start A/D" oder "Acquire A/D" Aktion werden die Objektdaten aktualisiert, sobald ein A/D Event auftritt (nach der spezifizierten Anzahl von Meßwerten). Die Daten sind ein Vektor mit der Länge der gewünschten Meßpunktanzahl, oder eine Liste von Vektoren, falls mehr als ein Kanal abgetastet wird.

Wenn die "Raw data mode" Setting gesetzt ist, werden die A/D Daten als Datentyp **string** zurückgegeben, der die binären Daten enthält, die direkt von der Hardware eingeladen wurden. Diese Daten müssen später in normale Spannungswerte konvertiert werden, indem die 'Convert to volts' Aktion benutzt wird.

Initialwert - keine Daten

Einstellungen (Settings)



Board number - Angabe, welches Board benutzt wird. Es ist möglich, mit TestPoint bis zu 4 A/D Boards parallel zu betreiben. Die Board

Number bezieht sich auf die A/D Hardware Konfigurationsinformation in der TESTPT.INI Datei (siehe das Hardware Konfigurationskapitel).

Demo mode - falls diese Einstellung wahr ist, läuft das A/D-Objekt im Demomodus, anstatt auf richtige Hardware zuzugreifen.

Raw data mode - falls wahr, werden die A/D Daten als binäre Rohdaten zurückgegeben (ein **string** Datentyp) anstatt als Spannungswerte. Dieser Modus ist eindeutig schneller in Applikationen, wo ein direktes Schreiben der Daten auf Festplatte oder keine weitere numerische Aufbereitung erforderlich ist. Die "Convert to volts" Aktion kann nach der Datenaufnahme benutzt werden, um diese Rohdaten in Spannungswerte umzuwandeln.

Achtung: Im ‘Raw data’ Modus muß die Anzahl der Samples pro Event kleiner als 32768 sein, da dies die Datenbeschränkung der Länge des String Datentyps ist.

Action List

Die Action List wird ausgeführt, sobald das Background A/D Sampling nach der "Start A/D" Aktion aktiv ist. Bei jeder Ausführung der Action List sind die Daten des A/D Objekts gleich den aufgenommenen Werten. Beispielsweise: Falls die Start Aktion ein Sample pro Event spezifiziert, sind die A/D Objekt Daten eine einzelne Zahl, die vor jeder Ausführung der Action List erneuert wird.

Beispiel

Diese Action List nimmt 100 Meßwerte vom A/D Kanal 0 auf und stellt sie graphisch dar.
Action List für den Pushbutton "Run":

1)	Acquire A/D	A/D1	# samples=100, rate=1000 Hz, channel(s)=0
2)	Draw Graph	Graph1	with A/D1

Beispiel - Strip charting

In diesem Beispiel startet ein "Start" Schalter die Datenaufnahme von zwei Kanälen:

1) Start A/D	A/D1	# samples="continuous", rate=1 Hz, channel(s)="0,1", event after 1 sample

Hierbei ist die Action List für das A/D Objekt:

1) Add points to	Graph1	from A/D1

Bitte beachten Sie, daß der Pushbutton nur die Aufnahme mit der "Start A/D" Aktion beginnt. Sobald jeweils ein Meßwert ankommt (1 pro Sekunde), führt das A/D Objekt seine Action List aus, die die neuen Punkte an den Graphen hinzufügt. Falls der Graph auf den Strip Chart Modus gesetzt ist, wird er zum Scrollen anfangen, sobald die Punkte den rechten Rand des Graphen erreichen.

Beispiel - Raw data mode

Diese Action List sammelt und speichert binäre A/D Rohdaten auf Platte, wenn die "Raw data mode" Setting des A/D Objekts eingeschaltet ist.

Start Pushbutton:

1) Start A/D	A/D1	#samples="continuous", rate=30000, channels=0, event after 30000 samples

A/D Objekt:

1) Output to	File1	from A/D1, term.=none

DieseAction List liest binäre A/D Rohdaten ein und konvertiert sie in Spannungswerte:

1)	Input from	File1	up to 1000000 bytes, stopping at ___
2)	Convert to volts	A/D1	raw data=File1, channels=0

Beispiel - Pretriggering

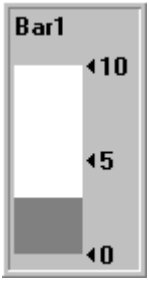
Dieses Beispiel initialisiert eine Keithley/Metrabyte DAS1800 A/D-Karte für einen externen digitalen Trigger, und nimmt 100 Samples vor dem Triggerereignis auf:

1)	Set A/D trigger digital	A/D1	
2)	Other A/D command	A/D1	command word="pretrigger", value=100
3)	Acquire A/D	A/D1	#samples=1000, rate=10000, channels=0

Balken Anzeige Objekt 

Benutzerschnittstelle

Ein Rechteck, das mit einer wählbaren Farbe ausgefüllt wird, und das eine Höhe besitzt, die auf dem Dateneingabewert basiert. Die Beschriftung, die Caption (Beschriftung), und der Bezelrahmen sind optional.



Aktionen

Set - Setzt die Balkenhöhe auf den Wert des Parameterwerts.

Bitte beachten Sie: Um ein Setting dieses Objekts in einer Action List zu verändern, ziehen Sie dieses Setting aus dem Settings Fenster in das Action List Fenster.

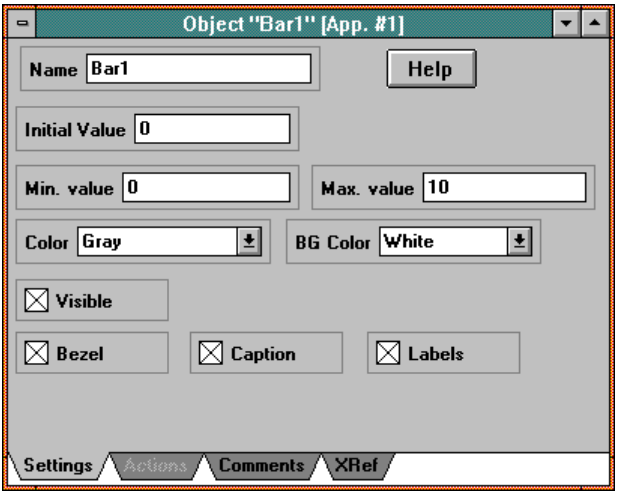
Daten

Der Datenwert des Bar Objekts ist der letzte Wert, der in der Set Action verwendet wurde.

Initialwert

Der Initialwert kann in den Settings angegeben werden.

Einstellungen (Settings)



Min. value - Der Wert, der am unteren Ende der Balkengraphik angezeigt werden soll. Sobald der Datenwert am Minimalwert angekommen ist, ist die Balkenfläche komplett mit der Hintergrundfarbe gefüllt. Unterhalb dieses minimalen Wertes erscheint ein nach unten zeigender Pfeil und zeigt an, daß die Datenwerte unterhalb des Minimums liegen.

Max. value - Der Wert, der oben am Balken erscheint. Oberhalb dieses maximalen Wertes erscheint ein nach oben zeigender Pfeil und zeigt an, daß die Datenwerte oberhalb des Maximums liegen.

Initial value - Der Datenwert, den der Balken beim Start der Applikation bekommt, bevor jegliche "Set" Action ausgeführt wird.

Color - Die Balkenfarbe.

BG Color - Die Farbe der Fläche oberhalb des Balkens.

Visible - Kontrolliert, ob das Objekt sichtbar auf dem Panel ist.

Caption - Falls angekreuzt, wird die Caption um die Balkenfläche gezeichnet.

Bezel - Falls angekreuzt, wird die Bezel, bzw. der Rahmen um das Objekt gezeichnet.

Labels - Falls angekreuzt, werden numerische Skalenlabels am rechten Rand des Balkens angezeigt. Die Anzahl der Labels und deren Abstand wird automatisch bestimmt, und paßt sich bei einer Größenänderung oder Änderung der Min/Max Werte des Balkens an.

Action List

keine

Bitmap Schalter Objekt

Benutzerschnittstelle

Das optische Aussehen, das von diesem Objekt dargestellt wird, kann mit Hilfe kommerzieller Zeichenprogramme erstellt werden. Auf diese Weise kann es jegliches Aussehen bekommen.

Das Bitmap Schalter Objekt kann sich wie ein gewöhnlicher Pushbutton, ein Zwei-Positionsschalter, oder sogar ein n-Rotations-Positionsschalter verhalten. Die Settings des Objekts kontrollieren die Art, wie das Aussehen der Objektfläche in maus-sensitive Regionen eingeteilt wird, und die Anzahl der Zustände (bzw. der Werte), die der Schalter haben darf.

Falls der Bitmap Schalter für Eingabeoperationen nicht mehr zugänglich gemacht wird, kann er als Anzeigeobjekt verwendet werden, der ein Bild zeigt, das von seinem jeweiligem Wert abhängig ist.

Die Objekt Stock Datei SWITCHES.STK, die mit TestPoint mitgeliefert wird, beinhaltet einige nützliche Beispiele von Bitmap Schaltern, die vorkonfiguriert und vorgezeichnet sind. Um diese Datei nutzen zu können, muß das Utilities/Load Stock Menu Kommando benutzt werden. Dann kann mit Hilfe des selben Menukommandos wieder der ursprüngliche TPEDIT.STK geladen werden (Inhalt des original TestPoint Objekt stocks (Lagers)).

Aktionen

Set - Setzt den Schalter auf einen gegebenen Wert. Falls der Parameter ein String ist, wird er erst mit den Labeln des Bitmap Schalters verglichen. Falls eins der Label zutrifft, wird der Schalter auf diese Position gebracht. Falls keins der Label das richtige ist, wird der Parameter als ein numerischer Index für den Schalter verwendet (der bei 0 startet). Nachdem der Schalterwert gesetzt worden ist, wird die zugehörige Action List ausgeführt.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

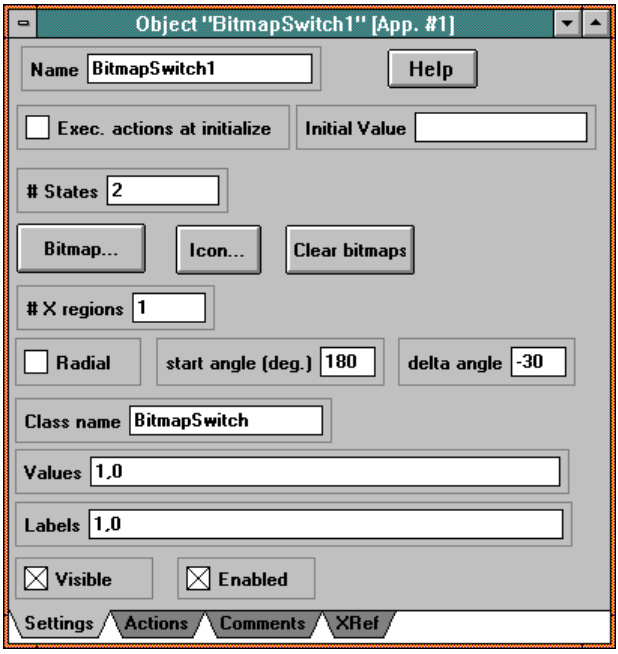
Der Datenwert des Bitmap Schalters ist der letzte Wert der Set Action, bzw. der letzte Wert, der vom Bediener per Maus oder Tastatur eingestellt worden war.

Initialwert

Der Initialwert hängt von den Settings des Objektes ab.

Einstellungen (Settings)

Bitte beachten Sie: Die Settings des Bitmap Schalters erlauben viele Variationen, und sind deswegen ein wenig komplex. Bitte prüfen Sie erst die mitgelieferten Beispiele in SWITCHES.STK, bevor eventuell Schwierigkeiten bei der Definition eigener Schalter auftauchen.



 # states - Die Anzahl der unterschiedlichen Werte (Positionen), die der Schalter annehmen darf. Dies darf eine Zahl größer oder gleich eins sein.

Falls # states **eins** ist, agiert der Schalter als ein Pushbutton. Er hat keine sinnvollen Datenwerte, und die Action List wird ausgeführt, sobald der Schalter angeklickt wird, ebenso wie beim Pushbutton Objekt. Das normale und das hineingedrückte Bild des Knopfes wird über die Bitmaps definiert.

Wenn # states **zwei oder mehr beträgt**, besitzt der Schalter mehrere mögliche Datenwerte (Positionen).

Bitmap - Dieser Knopf wird dazu benutzt, das Bitmap einzulesen, das das Aussehen des Schalters definiert. Das Bitmap File kann mit Windows Paintbrush oder anderen Zeichenprogrammen erstellt werden. Die unterstützten Dateiformate sind: BMP, PCX, TIFF.

Sobald das Bitmap eingelesen ist, wird seine Breite in zwei oder mehr Regionen abhängig vom # states Setting unterteilt. Jede Region definiert, wie der Schalter aussehen wird, sobald er in den zugehörigen Zustand gesetzt wird. (Bitte beachten Sie: für #states=1 gibt es auch zwei Bildhälften: der erste für den "up" Knopfstatus, und der zweite für den "down" Status, das bedeutet, wenn der Knopf gedrückt wird).

Beispielsweise gilt für dieses Bitmap:

Mit #states=2 wird ein Kippschalter definiert. Die linke Hälfte des Bitmaps ist Zustand 0, die rechte entspricht Zustand 1.

Am besten arbeitet man an solchen Bitmaps erst an einem Zustand. Sobald dieser vom Aussehen definiert ist, kopieren Sie ihn und fügen ihn wieder modifiziert ein, um die unterschiedlichen Schalterzustände zu bekommen. Bitte beachten Sie, daß der Abstand der einzelnen Zustände gleichmäßig über der Breite des Bitmaps verteilt liegt. Die totale Breite des Bitmap sollte ein Vielfaches der Anzahl der Zustände sein.

Das Bitmap darf eine beliebige **Anzahl an Farben** aufweisen, und der Bitmap Schalter darf mehrere Bitmaps mit unterschiedlicher Anzahl an Farben beinhalten. Jedes Mal, wenn der Bitmap Knopf im Settingsfenster benutzt wird, kann ein Bitmap eingelesen werden. Falls

das Bitmap die selbe Anzahl an Farben wie ein gerade vorher eingelesenes besaß, wird es das alte ersetzen.

Beispielsweise können Sie ein 16-Farben Bitmap, das auf allen Windows Systemen benutzt werden kann, ebenso einlesen, wie ein 64K Farb Bitmap mit eingescannten Bild. TestPoint speichert beide Bitmaps in dem Objekt, und benutzt die höchste Anzahl an Farben, die die Bildschirmauflösung bietet.

Bitmaps werden in einem komprimiertem Format innerhalb des Objekts gespeichert, so daß das Applikations File so klein wie möglich gehalten wird. Ein Programm, das viele hochauflösende Vielfarb-Bitmaps beinhaltet, kann dennoch sehr groß werden.

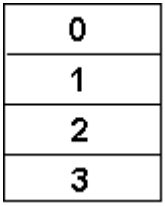
Icon - Dieser Knopf erlaubt es, daß ein anderes Icon für das Objekt anstatt des normalen mit der Aufschrift "USER" ausgewählt werden kann. Falls in Ihrem Stock viele verschiedene Bitmap-Schalter sind, ist es oftmals hilfreich, ein Icon zu verändern, um den Überblick zu wahren. Die Icon Datei sollte ein 24 mal 24 Bitmap in 16 Farben sein.

Reset bitmap - Dieser Knopf löscht alle Bitmaps im Objekt (aller Farbauflösungen) und geht zurück auf ein 16 Farben Bitmap. Oftmals ist es nützlich, wenn große Bitmap Files eingelesen wurden (z. B. 24-bit Bitmaps), daß man die Filegröße reduzieren will und auf 16 Farben zurückgehen will.

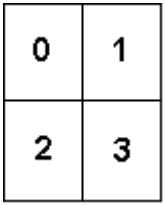
 # X regions - Dieses Setting kontrolliert, wie der Schalter auf die Maus reagieren soll (sofern die Radial Setting ausgeschaltet wurde).

Falls der Schalter N Zustände besitzt, muß die Schalterfläche in N Regionen unterteilt werden, von denen jede, die angeklickt wird, einen Zustand repräsentiert. Die # X regions Setting bestimmt, in wie viele Teile die Breite des Schalters eingeteilt wird. Die Höhe des Schalters wird dann in die notwendige Anzahl an Teilflächen unterteilt. Meistens ist die # X regions gleich eins.

Beispielsweise gilt für einen Schalter mit einem # X regions, und vier # states folgende Flächenverteilung:



Für zwei # X regions, und vier # states, bietet sich folgendes Bild:



Man sollte die # X regions und # states Settings angepasst an das Bitmap Bild vergeben, so daß es für den Bediener offensichtlich ist, mit welchem Klick er welche Auswahl trifft.

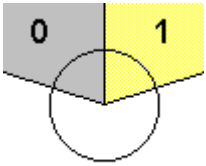
X regions ist gleich **null** in einem speziellem Fall, falls # states = 2. Sobald diese Kombination verwendet wird, gibt es nur eine Flächen-/Mausregion. Der Schalter ändert sich jedes Mal, wenn er geklickt wird, somit kann ein verzögerter Druck-Druckknopf realisiert werden.

Radial - Falls angekreuzt, agiert der Schalter als ein N-Position Drehschalter. Er wird auf Maus-Klicks sensitiv, die auf einem gewissem Winkel basieren um den Schaltermittelpunkt, anstatt einer linearen Klickverteilung. Der Startwinkel und die jeweilige Winkelveränderung (Delta-angle) definieren die Schalterstellungen.

start angle (deg.) - Dieses Setting definiert bei eingeschaltetem Radial Setting den Beginn der ersten Radialregion (Kuchenstück). Dieser

Winkel wird in Grad angegeben, die gegen den Uhrzeigersinn von der positiven x-Achse gezählt werden.

Beispielsweise ergeben sich für einen Start Angle von 160, und einem Delta Angle von -70, und # states gleich zwei, folgende Schalterregionen:



(Falls der Benutzer weiter gegen den Uhrzeigersinn als die Startposition klickt, wird der Zustand 0 geschaltet. Falls er weiter im Uhrzeigersinn als die Endposition klickt, wird der letzte Zustand ausgewählt.)

Dieser Start- und Endwinkel müssen nicht bei 90 Grad (geradlinig nach oben) beginnen.

delta angle - Dieses Setting definiert die Breite jeder Radialregion (Kuchenstück), sofern das Radial Setting gesetzt ist. Es wird in Einheiten von Grad entgegen dem Uhrzeigersinn gemessen, deshalb bedeutet ein negativer Wert, daß die Regionen im Uhrzeigersinn fortlaufen.

Class name - Diese Zeichenkette ist der Default Name, der für neue Kopien dieses Objekts verwendet wird, falls es aus dem Stock Fenster in eine Applikation gezogen wird. Falls Sie einen Stock mit vielen unterschiedlichen Bitmap Schaltern definiert haben, ist es oftmals vorteilhaft, jedem einen eigenen Class Namen zu geben, z. B. "Kippschalter", "Schieberegler", "Zweifach-Drehregler", etc.

Values - Dieses Setting beinhaltet die Datenwerte, die der Bitmap Schalter in den jeweiligen Zuständen besitzt. Falls hier kein Eintrag

vorgenommen wird, sind die verwendeten Werte Nummern von 0 bis # states-1. Diese Settings können beliebige Strings sein, die durch Kommata getrennt sind. Falls Kommata in den gewünschten Strings vorkommen sollen, sollten sie in Anführungsstrichen gesetzt werden.

Bitte beachten Sie: Für einen **Zwei-Stellungs** Schalter, wie z. B. ein **Kippschalter** oder ein **Schieberegler**, sollten die Werte Settings eine 1 und eine 0 sein, wie es im Bild am Start dieses Abschnitts gezeigt wurde. Der Bitmap Schalter erzeugt dann Flächenregionen, die am oberen Ende der Flächenregionen beginnen, somit ist der Zustand #0 die oberste Mausklick Region. Dies ist ebenso für N-Position Auswahlschalter, Zwei-Position Schalter besitzen jedoch gewöhnlich den Wert 1, wenn der Schalter oben ist, und den Wert 0, sobald er unten ist.

Labels - Dieses Setting beinhaltet die Labels für jeden Schalterzustand, ebenso wie das Label Setting des Selector Objekts. Diese Labels werden jedoch nicht am Bildschirm dargestellt. Sie werden nur für die "Set" Action als mögliche Werte für den Action Parameter verwendet. Die Schalterstellung kann mit Hilfe des Labels oder dem Zahlenwert (0 bis # states -1) gesetzt werden.

Visible - Falls nicht angekreuzt, wird der Bitmap Schalter nicht auf dem Panel dargestellt.

Enabled - Falls nicht angekreuzt, reagiert der Bitmap Schalter nicht auf Maus Klicks oder auf die Tastatur. Das Objekt kann aber immer noch als ein selbst-definiertes Anzeigeobjekt verwendet werden, indem man seinen Datenwert mit der "Set" Action benutzt.

Exec. Aktionen at init - Falls angekreuzt, wird die Action List des Objekts bei der Programminitialisierung ausgeführt.

Initial value - Der Zustand, den der Schalter beim Start der Applikation annimmt. Dieser Wert hat dieselbe Form, wie der

Parameter bei der "Set" Action - das bedeutet, daß er einem Label Setting entspricht, oder eine Nummer von 0 bis # states - 1 sein darf.

Action List

Die Action List wird ausgeführt, sobald der Zustand des Schalters mit Maus Klicks, der Tastatur oder der "Set" Action verändert wird.

Beispiel - 2-Positions Schalter

Folgende Settings können verwendet werden (bitte beachten Sie, daß die Werte 1/0 sind, wie es vorher dargestellt wurde):

# states:	2
# X regions:	1
Radial:	off
Values:	1,0
Labels:	1,0

Folgendes Bitmap kann verwendet werden:

Beispiel - Eigendefinierter Druckschalter

Folgende Settings sind zu benutzen:

# states:	1
# X regions:	1
Radial:	off

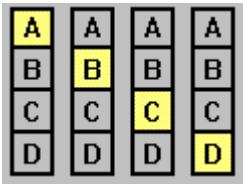
Folgendes Bitmap kann verwendet werden:

Beispiel - 4-Position Schalter

Folgende Settings können verwendet werden:

# states:	4
# X regions:	2
Radial:	off
Values:	first,second,third,fourth
Labels:	A,B,C,D

Das Bitmap kann so aussehen:



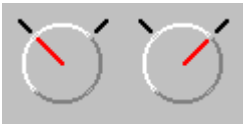
Die Action List kann diesen Schalter benutzen, indem Labels mit den Bezeichnungen "A" oder "B" verwendet werden.

Beispiel - Drehschalter

Settings:

# states:	3
Radial:	on
start angle:	180
delta angle:	-60

Bitmap:



Case Objekt 

Das Case Objekt erlaubt eine bedingte Ausführung einer oder mehrerer Alternativaktionen. Es kann als If/Then Statement angesehen werden, hat aber mehrere Optionen, die auf Basis einer mathematischen Operation gewählt werden können.

Das Case Objekt benutzt das selbe mathematische Ausdruckformat wie das Math Objekt. Weitere Details zu dieser Syntax kann man den Erläuterungen zu dem Mathematik Objekt entnehmen.

Das Case Objekt wird benutzt, indem man seine **Select** Action in eine Action List einfügt. Zwei Action Zeilen erscheinen daraufhin (**Select** und **End**), diese rahmen die Bedingungssektion der Action List ein. Daraufhin können eine oder mehrere Fallunterscheidungen definiert werden, indem man sogenannte **When** Aktionen zwischen **Select** und **End** einfügt. Um die Action Zeilen zu bewegen, kann die Action Zeilen Nummer einfach auf den neuen Platz geschoben werden.

Benutzerschnittstelle - none

Aktionen

Select - Berechnet die Expression Formel mit den gegebenen Parametern. Der Parameter dieser Aktions-Zeile hängt von dem Expression Setting ab. Das Resultat dieser Berechnung wird mit jeder der **When...is** Aktionen verglichen, bis eine gefunden wird, die gleich dem Resultat ist (oder die **Default** Aktion gefunden wurde). Sobald die entsprechende Case Anweisung gefunden wurde, werden die zugehörigen Aktions-Zeilen zwischen dieser **when** Aktion und der nächsten (oder der **End** Zeile) ausgeführt.

Das Resultat von **Select** kann eine Zahl, ein String oder ein anderer Datentyp sein.

When - Startet eine Gruppe von Aktions-Zeilen, die ausgeführt werden, wenn der Select Ausdruck gleich einem vorgegebenem Wert ist. Dieser Vergleichswert in der Action List kann eine Konstante oder eine Variable (TestPoint Objekt) sein.

When/Range - Startet eine Gruppe von Action Zeilen, die ausgeführt werden, wenn der Select Ausdruck innerhalb eines gegebenen Bereiches liegt (gültig nur für numerische Werte, es wird auf jeden Fall immer ein numerischer Vergleich vorgenommen). Der Vergleichswert in dieser Aktion kann eine Konstante oder eine Variable (TestPoint Objekt) sein.

Default - Startet eine Gruppe von Action Zeilen, die ausgeführt werden, wenn keiner der vorhergehenden **when** Werte auf den **select** Ausdruck zutrifft. Diese Möglichkeit sollte normal an letzter Stelle stehen, da alle nachfolgenden **when** Aktionen niemals ausgeführt werden.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des Case Objekts ändert sich, sobald die **Select** Aktion ausgeführt wird, es wird dann auf das Ergebnis des auszuwertenden Ausdrucks gesetzt.

Initial value - keine Daten

Einstellungen (Settings)

Expression - Jeder mathematische oder logische Ausdruck , der der Syntax des Math Objekts folgt.

Action List - keine

Beispiel

Ein Case Objekt namens "Part Number" mit dem Ausdruck:

```
substr(s,1,6)
```

Dies wurde ausgewählt, um eine Teilenummer aus einem längeren String zu extrahieren, und dann anhand dieser Teilenummer jeweils eine Gruppe von Aktionen auszuführen. Beispielsweise kann folgender Aufruf benutzt werden:

```
1)  Select          Part Number     with s=File1
2)  When            Part Number     is 1000
3)     Execute      Action1
4)  When            Part Number     is 2000
5)     Execute      Action2
6)  When/Range      Part Number     from 3000 to 4000
7)     Set          Display1        to Part Number
8)     Execute      Action3
9)  Default         Part Number
10)                    Execute      Other-action
11)                 End             Part Number
```

Code Objekt 

Das Code Objekt erlaubt es, daß eine externe Funktion, die in einer herkömmlichen Programmiersprache geschrieben wurde, aufgerufen werden kann. Diese Routinen müssen in Form einer Windows Dynamic Link Library (DLL) vorliegen, und können in jeder Sprache erzeugt werden, die das Schreiben von DLLs unterstützt, beispielsweise C, C++, Turbo Pascal für Windows, etc. (Bitte beachten Sie: Visual BASIC unterstützt nicht die Erstellung von DLLs).

Benutzerschnittstelle - keine

Aktionen

Call - Aufruf der externen Funktion. Jedes Code Objekt kann eine externe Funktion aufrufen. Falls mehrere Routinen einer DLL aufgerufen werden müssen, muß für jede Funktion ein Objekt verwendet werden.

Die Call Action Zeile kann null oder mehr Parameter haben, dies hängt von den Objekt Settings ab (Argument Typen).

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des Code Objekts ist der Rückgabewert der zuletzt aufgerufenen Funktion.

Initial value - keine Daten

Einstellungen (Settings)

DLL Filename - Der Name der DLL Datei (z.B. MYCODE.DLL). Die speziellen Namen KERNEL, USER, und GDI (alle ohne die .DLL Extension) können benutzt werden, um interne Windows Funktionen aufzurufen.

Subroutine Name - Der Name der aufzurufenden Funktion. Groß- und Kleinschreibung sind hierbei nicht wichtig, da die Windows Standard-Aufrufkonvention Pascal benützt wird.

Argument Types - eine Liste von Argumenttypen, die durch Kommata getrennt sind. Die Anzahl an Elementen in dieser Liste bestimmt die Anzahl an Parametern, die innerhalb der Call Action Zeile erscheinen. Folgende Argumenttypen stehen zur Verfügung:

char, byte, integer, word, long, dword, float, double, string, hwnd, window

Dabei sind **char**, **integer** und **long** mit Vorzeichen versehene, und **byte**, **word** und **dword** nicht vorzeichenbehaftete Zahlen. **char** und **byte** sind 1 Byte-Werte, **integer** und **word** sind 2 Bytes lang, und **long** und **dword** besitzen eine Länge von 4 Bytes. **strings** werden als nullzeichen-terminierte Zeiger übergeben. Ein **hwnd** Parameter übergibt ein Fenster "Handle", oder ID Nummer, eines Objektes an den externen Code (siehe Beispiele). Ein **window** Parameter übergibt ein Fenster "handle" für ein verborgenes Nachrichtenfenster, das es ermöglicht, daß externer Code TestPoint Events erzeugen kann (siehe Beispiele).

Bitte beachten Sie: Einige ältere C Compiler können keine Fließkommaargumente übergeben - sie wandeln sie in Double Werte um. Das gilt nicht für MS C 7.0 oder Borland C 3.1 oder höher.

Das **var** Schlüsselwort kann vor jedem Argument dazu benutzt werden, zu spezifizieren, daß der Wert von der Unterroutine verändert werden kann, und daß der neue Wert an TestPoint übergeben werden soll. Var Parameter werden übergeben, indem ein Zeiger auf die Daten übergeben wird. In der Aufruf Action Zeile muß jeder Var Parameter mit einer Objekt Referenz versehen werden, nicht mit einer Konstanten oder einem leeren Wert. In dem DLL Code müssen diese Zeiger als "huge" deklariert werden, dies erlaubt, daß die totale Argumentdatenmenge größer als 64 Kbytes sein darf.

Jedes der Argumenttypen kann optional von einer Zahl in eckigen Klammern gefolgt sein (z. B. [1024]), die ein Array von Datenwerten der gegebenen Größe spezifiziert. Arrays werden immer in Form eines Pointers übergeben. (Eine Dimension gefolgt von einem **string** Argumententyp spezifiziert kein Array, sondern eine maximale Stringlänge, die ohne weitere Angabe auf 255 steht.)

Man kann auch die eckigen Klammern ohne eine weitere Zahl im Inneren verwenden: [], um eine variable Größe des übergebenen Vektors anzuzeigen. Wenn die Subroutine aufgerufen wird, werden die aktuell übergebenen Daten in ein TestPoint Array mit mindestens einem Element übergeben (die Anzahl der Elemente hängt vom übergebenen Datentyp ab). Normal sollte ein anderes Argument definiert werden, welches die Größe des übergebenen Arrays angibt, somit verhindert man, daß die Array Grenzen überschritten werden. (Siehe folgende Beispiele).

Return Type - Der Datentyp des Rückgabewertes der Subroutine.

Preload? - Diese Option zeigt an, ob die DLL sofort geladen werden sollte, oder ob TestPoint damit warten sollte, bis das erste Mal der Funktionsaufruf dieses Code Objekts stattfindet. Sofortiges Laden verhindert einen Delay, wenn der erste Aufruf getätigt wird. Falls aber die Applikation selber den Pfad des DLL Dateinamens während der Ausführung ausfüllen soll, sollte diese Option ausgeschaltet sein.

Action List

Die Action List des Code Objekts wird ausgeführt, sobald das Code Objekt ein Windows Ereignis erhält. Dies passiert in der DLL Subroutine, indem die Windows PostMessage Funktion verwendet wird, um ein Ereignis an das Code Objekt zu übergeben. Die DLL Subroutine erhält die Fenster "handle" Nummer durch einen Aufruf, der ein "window" Argumenttyp verwendet. Siehe die Beispiele weiter unten.

Das Code Objekt führt seine Action List nur für Windows Ereignisse aus, die im User Event Bereich (WM_USER bis 0x7FFF) liegen, oder wenn es sich um die WM_TIMER Message handelt.

Sobald die Action List ausgeführt wird, wird der Datenwert des Objekts auf einen LIST Datentyp gesetzt. Die Liste enthält drei Elemente: Den Message Code, den Word Message Parameter (WPARAM) und den long Integer Message Parameter (LPARAM). Die Action List kann dann dazu benutzt werden, herauszufinden, welches Ereignis empfangen wurde.

Beispiel - Aufruf einer Windows eingebauten Funktion

Erzeugen Sie eine Applikation mit einem Pushbutton (Druckschalter), einem Code Objekt und einem Display. Diese Applikation soll aus Windows die Information über das Windows Verzeichnis liefern (normal C:\WINDOWS).

Die GetWindowsDirectory Funktion ist im Windows Programmierhandbuch wie folgt definiert:

```
WORD GetWindowsDirectory (LPSTR lpBuffer, int nSize)
```

Dafür müssen folgende Settings im Code Objekt verwendet werden:

DLL filename:	KERNEL
Subroutine name:	GetWindowsDirectory
Argument types:	var string,integer
Return Type:	word

Die Action List für den Pushbutton lautet:

```
1) Call        Code1        with  Display1 , 80
```

Die 80 bedeutet ein character string der Länge 80. Im Run Modus kann dann der Pushbutton gedrückt werden. Die aufgerufene Windows Routine gibt den Pfad des Windows Verzeichnis zurück. Da ein "var string" Parameter verwendet wird, wird der Rückgabewert direkt in den Datenwert des Displayobjekts, und damit in das Display geschrieben.

Beispiel - Zugang zu eigener Hardware durch I/O Ports

Dieses Beispiel nutzt externen Code, um Treiber Funktionen für eigene oder noch nicht unterstützte Hardware anzusteuern. Nehmen wir an, daß die Hardware eine Kanalnummer als Argument benötigt und einen numerischen Wert seiner Messung zurückgibt.
Mit diesen Settings für das Code Objekt ist ein Ansprechen möglich:

DLL filename:	CUSTOMIO.DLL
Subroutine name:	Measure
Argument types:	integer, var double
Return Type:	word

Die Action List für den Pushbutton lautet:

1) Call Code1 with Daten-Entry1 , Display1

Der externe Code in C könnte wie folgt aussehen (Borland C++ wird für dieses Beispiel verwendet, dennoch sind auch andere Sprachen möglich):

```
#include <windows.h>
#define ADDRESS 0x300
// note: extern "C" is required for C++, but not for regular C language:
extern "C" WORD WINAPI    _export   Measure
                    (int channel, double *result)
{
    outp (ADDRESS,channel);
    while ((inp(ADDRESS) & 1) == 0) ;  // wait for "done" flag
    char c = inp(ADDRESS+1);
    *result = (double) c * 1.5 + 2.0;
    return 0;
}
int FAR PASCAL LibMain (HANDLE,WORD,WORD,LPSTR)
{
    return 1;          // LibMain required in all DLLs
}
```

Beispiel - Vektor Arithmetik

Dieses Beispiel benutzt eine externe Subroutine, um einen Vektor numerischer Werte mathematisch zu manipulieren. Das Beispiel ist einfach genug, um es auch mit TestPoint sehr viel einfacher zu lösen, aber die Idee dahinter kann erweitert werden, so daß speziell optimierte Funktionen geschrieben werden können, oder spezielle Hardware wie beispielsweise ein Koprozessor angesprochen werden kann, um diese Manipulationen durchzuführen.

Die Code Objekt Settings lauten:

DLL filename:	MYMATH.DLL
Subroutine name:	Process
Argument types:	var double[], integer
Return Type:	word

Hier ist eine mögliche Action List für diesen Code:

1)	Acquire A/D	A/D1	#samples=1000, rate=10000 Hz, channel(s)=0
2)	Calculate	Length	with v=A/D1
3)	Call	Code1	with A/D1 , Length
4)	Draw Graph	Graph1	with A/D1

"Length" ist hierbei ein Math Objekt mit der Formel **dim(v)**. Der length Parameter wird hierzu von der Subroutine benutzt, um zu erkennen, wie viele Parameter im Vektor vorhanden sind. Um length mit einem Math Objekt zu ermitteln, können wir die Anzahl der Samples in der "Acquire" Action nehmen, oder sie variabel machen.

Nach der Ausführung der Zeile 3, wird der Datenwert des A/D Objekts geändert (da ein **var** Parameter benutzt wurde).

Im folgenden der externe C Code:

```
#include <windows.h>
WORD WINAPI    _export   Process
                         (double *v, int n)
{
    int i;
    for (i=0;i<n;i++)
       *v++ = 2.0 * ((*v) * (*v));              // 2 times v squared
    return 0;
}
int far pascal LibMain (HANDLE h,WORD w1,WORD w2,LPSTR c)
{
    return 1;            // LibMain required in all DLLs
}
```

Beispiel - Manipulation von TestPoint Fenstern durch externen Code

Dieses Beispiel zeichnet Linien innerhalb des Fensters eines Picture (Bild) Objekts. Windows bietet zahlreiche Funktionen, um Linien, Kreise, und ähnliche Muster zu zeichnen. Diese Routinen erfordern alle einen Parameter, der "device context" ist, und der erhältlich ist, indem eine Windows interne Routine aufgerufen wird und ein "window handle" zurückgibt. Ein "window handle" ist ein Identifizierer eines entsprechenden Fensters am Bildschirm.

Die Windows internen Funktionen, die verwendet werden, sind im Windows Programmier Referenz Manual beschrieben:

```
HDC GetDC (HWND hwnd)
BOOL Rectangle (HDC hdc, int l, int t, int r, int b)
int ReleaseDC (HWND hwnd, HDC hdc)
```

Ein HDC ist der C Sprachentyp für einen "device context", und ist durch einen **word** Wert repräsentiert.

Demnach besteht der erste Schritt darin, drei Code Objekte für die folgenden drei Windows Routines zu erstellen:

GetDC:

DLL filename	USER
Subroutine	GetDC
Arguments	hwnd
Return type	word

Rectangle:

DLL filename	GDI
Subroutine	Rectangle
Arguments	word, int, int, int, int
Return type	int

ReleaseDC:

DLL filename	USER
Subroutine	ReleaseDC
Arguments	hwnd, word
Return type	int

Als nächstes sollten folgende Aktions-Zeilen diese Routinen aufrufen:

1)	Call	GetDC	Picture1
2)	Call	Rectangle	GetDC, 10, 10, 100, 100
3)	Call	ReleaseDC	Picture1, GetDC

Somit wird ein Rechteck auf dem Bild Objekt gezeichnet. Zeile 1 holt sich einen Device Context, der ein Fenster Handle des Picture (Bild) Objekts bekommt. Zeile 2 zeichnet das Rechteck, indem es den Device Context benutzt. Zeile 3 schließlich hebt den Device Context wieder auf.

Weitere Informationen sind den Beispielen EXAMPLES\DRAW.TST und EXAMPLES\PAINT.TST zu entnehmen.

Conditional Objekt 

Das Conditional (Bedingungs) Objekt läßt Action Lists verschiedene Aktionen ausführen, die anhand einer logischen Bedingung, eines Vergleichs oder einer Berechnung ausgewählt werden.

Das Conditional Objekt benutzt die gleiche Syntax für mathematische Ausdrücke, wie das Math Objekt. Weitere Details dazu und zu der Funktionsweise sind in der Dokumentation des Math Objekts ersichtlich.

Das Conditional Objekt wird benutzt, indem seine Aktion (if/then oder if/then/else) in eine Action List eingefügt wird. Mehrere Aktions-Zeilen erscheinen, die die Bedingungen innerhalb der Action List einrahmen. Um das Ende oder die Else Aktions-Zeilen zu bewegen, muß die Aktions-Zeilennummer auf- oder abwärts bewegt werden

Benutzerschnittstelle - keine

Aktionen

Zuerst wird der Bedingungsausdruck ausgewertet, indem die Parameter in der Aktions-Zeile verwendet werden. Danach können folgende Aktionen gewählt werden:

If/Then - Diese Aktion führt die Aktions-Zeilen zwischen der If/Then Zeile und der End Zeile aus, wenn der Bedingungsausdruck wahr ist (verschieden von Null).

If/Then/Else - Diese Aktion führt die Aktions-Zeilen zwischen der If/Then/Else Zeile und der Else Zeile aus, wenn der Bedingungsausdruck wahr ist (verschieden von Null). Falls der Ausdruck falsch ist (Null), werden die Zeilen zwischen der Else und der End Anweisung ausgeführt.

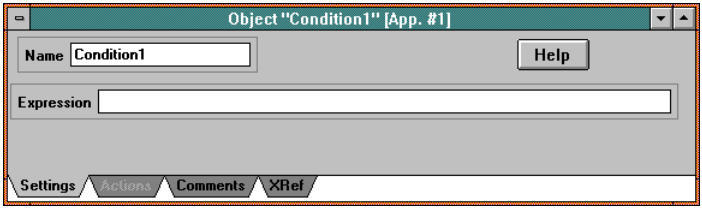
Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des Conditional Objekts ändert sich, sobald die Formel berechnet wird, diese wird auf das Resultat der Berechnung der Conditional Formel gesetzt.

Initial value -

Einstellungen (Settings)



Expression - Jeder mathematische/logische Ausdruck, der gemäß den Math Objekt Regeln formuliert ist.

Action List - keine

Beispiel

Ein Conditional Objekt mit dem Namen "X gleich Y" mit dem Expression:

```
X=Y
```

kann dafür benutzt folgendermaßen werden:

```
1)  If/Then          X equals Y        with X=Daten-Entry1, Y=4
2)     Set           Display1          to "You entered a four!"
3)  End if           X equals Y
```

Es kann ebenso dafür gebraucht werden, um mehrere Fälle zu unterscheiden, indem if/then/else Aktionen eingebaut werden:

```
1)  If/Then/Else     X equals Y        with X=File1, Y=1
2)     Set           Display1          to "A one was in the file."
3)  Else if not      X equals Y
4)     If/Then/Else  X equals Y        with X=File1, Y=2
5)        Set        Display1          to "It was a two."
6)     Else if not   X equals Y
7)        Set        Display1          to File1
8)     End if        X equals Y
9)  End if           X equals Y
```

Beispiel

Dieses Beispiel prüft das Resultat einer A/D Messung auf die Tatsache, ob alle Werte des Kanals 0 kleiner als die Hälfte der Werte von Kanal 1 sind. Im einen Fall leuchtet eine Fehleranzeige, im anderen Fall wird das Resultat graphisch dargestellt.

Das Conditional Objekt wird "data valid" genannt und hat folgenden Expression (Eintrag):

```
select(input,0) < select(input,1) * 2
```

Die select Funktion wird dazu benutzt, die beiden Kanäle zu trennen, da das A/D Objekt die Werte der beiden Kanäle als Liste zurückgibt.

Die Action List des "Run" Pushbutton lautet:

1)	Acquire A/D	A/D1	#samples=100, rate=1000 Hz, channel(s)="0,1"
2)	If/Then/Else	data valid	with input=A/D1
3)	Draw Graph	Graph1	with A/D1
4)	Else if not	data valid	
5)	Set	Indicator1	to 0
6)	End if	data valid	

Stil Tips

Oft kann man Objekte in der eigenen Applikation öfters benutzen, indem man den Expression des Conditional Objekts genereller formuliert. Damit kann das Objekt an mehreren Plätzen der Applikation verwendet und eine Menge an weiteren Conditional Objekten eingespart werden.

Beispielsweise kann anstatt des Expressions X=5, um auf den Wert 5 zu testen, die Formel X=Y verwendet werden, wie es im ersten Beispiel gezeigt wurde. Sobald eine Aktions-Zeile mit diesem Objekt verwendet werden soll, kann eine 5 in das Y= Parameterfeld geschrieben werden. Das selbe Objekt kann zusätzlich aber noch dazu benutzt werden, um die Gleichheit mit anderen Konstanten zu testen, oder um zu prüfen, ob zwei Objekte gleiche Werte besitzen.

Sehr selten sollte ein Conditional Objekt erforderlich sein, um den Wert eines Selector Objekts zu testen. Falls die Aktionen für jede gewählte Selector Option gleich sind, müssen nur unterschiedliche Daten in diesen Action Lists verwendet werden (beispielsweise, wenn unterschiedliche Kommandostrings an ein GPIB Instrument geschickt werden sollen). Zu diesem Zweck sollte der Selector die entsprechenden Datenwerte besitzen, die dann verwendet werden können, ohne daß weitere If/Then/Else Statements notwendig sind.

Container Objekt 

Das Container (Speicher) Objekt bietet die Möglichkeit, temporär Daten in einer TestPoint Applikation zu speichern. Container können benutzt werden, um Daten für den späteren Gebrauch in einer Action List zu speichern, falls die Orginal Datenquelle modifiziert wird. Es kann ebenfalls dazu benutzt werden, um Daten über mehrere Programmschritte oder Schleifen zu sammeln.

Benutzerschnittstelle - keine

Aktionen

Clear - Setzt den Datenwert des Containers auf "no data", oder leer.

Store in - Speichert die Daten, die in dem Parameterfeldern des Container Objekts angegeben sind. Jede beliebige Anzahl an Parametern kann angegeben werden. Falls ein einzelner Parameter verwendet wird, werden die Daten im Container gespeichert. Falls mehrere Parameter angegeben werden, wird der Datenwert des Containers eine **'Liste'** der Daten von jedem Parameter.

Append to - Fügt neue Daten den schon existierenden Daten im Container hinzu. Diese Aktion erzeugt Vektoren aus einzelnen Werten, indem die Werte hintereinander angehängt werden. Beispielsweise erzeugt das Speichern und das anschließende neunmalige Anhängen einer Zahl einen Vektor mit 10 Elementen. Die Aktion des Anhängens ist für Zahlen und Strings gültig. Das Anhängen an eine Liste ist nur gültig, falls die Daten an eine passende Liste angefügt werden, bei der jedes Listenelement ein Vektor wird (ähnlich wie man Reihen an eine Tabelle mit mehreren Spalten anhängt).

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des Objekts hängt von den Aktionen ab, die für die Füllung des Containers verantwortlich sind.

Initial value

Beim Programmstart ist der Container leer (enthält no data).

Einstellungen (Settings) - keine

Action List - keine

Data Entry Objekt 

Das Data Entry Objekt erlaubt dem Benutzer, Informationen mittels Applikationspanels einzugeben.

Falls der Benutzer in das Feld schreibt, und danach die ESC-Taste drückt, nimmt das Objekt seinen vorigen Wert wieder an und keine weitere Aktion läuft ab (somit erfüllt die ESC-Taste die Abbruchfunktion). Falls der Benutzer Enter drückt oder mit der Maus auf einen anderen Teil des Bildschirms klickt oder TAB benutzt, wird der neu eingegebene Wert verwendet und die Action List des Objekts ausgeführt.

Benutzerschnittstelle

Ein Texteingabefeld mit der Überschrift, die gleich dem Objektnamen ist:

Die Caption (Beschriftung) und Bezel (Rahmen) sind optional.

Aktionen

Set to - Der Datenwert des Data Entry Feldes kann auf einen neuen Wert gesetzt werde, indem diese Aktion verwendet wird. Nachdem der neue Wert in das Data Entry Feld durch das Programm gesetzt wurde, wird die Action List des Objekts ausgeführt, als hätte der Benutzer einen neuen Wert eingetippt.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

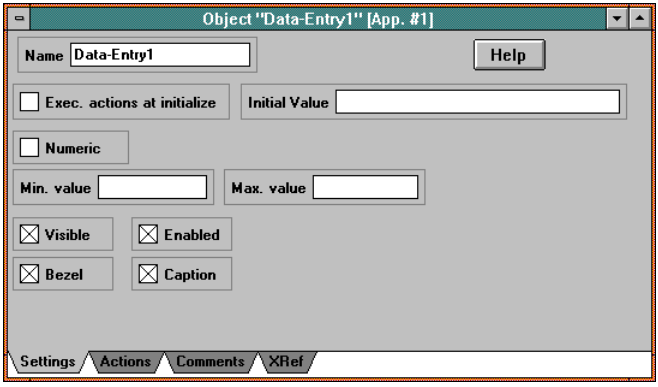
Daten

Der Datenwert des Objekts ist der Wert der letzten Eingabe. Dies kann eine Zahl oder ein String sein. (Bitte beachten Sie: der Wert wird immer als String definiert, außer wenn die Numeric Option in den Settings angewählt ist. TestPoint konvertiert jedoch von Strings in Zahlen und umgekehrt, sofern es erforderlich ist.)

Initial value

Der Initial Value des Data Entry Feldes kann in den Settings spezifiziert werden.

Einstellungen (Settings)



Visible - Kontrolliert, ob das Data Entry Feld auf dem Panel sichtbar ist.

Enabled - Kontrolliert, ob das Data Entry Objekt für Eingaben offen oder gesperrt ist. Falls nicht angekreuzt, wird die Überschrift grau und der Benutzer kann keine Eingaben vornehmen.

Exec. Aktionen at initialize - Falls angekreuzt, wird die Action List ausgeführt, sobald das gesamte Programm initialisiert wird.

Initial Value - Jeder Wert kann bei der Programminitialisierung in das Eingabefeld gesetzt werden.

Numeric - Falls angekreuzt, erlaubt dieses Setting nur numerische Eingaben.

Minimum Value - Falls die Numeric Option angekreuzt ist, und dieses Setting nicht leer ist, müssen die eingegebenen Werte gleich oder größer dem spezifizierten minimalen Wert sein. Andere Werte (Out-of-Range) erzeugen eine Fehlermeldung und ermöglichen dem Benutzer eine Neueingabe.

Maximum Value - Ähnlich zu vorigem Setting. Spezifiziert den größten Wert, der erlaubt ist. Entweder das Minimum oder das Maximum Value Setting kann freibleiben, falls es nicht verwendet werden soll.

Bezel - Falls angekreuzt, wird die Bezel oder der Rahmen um das Objekt gezeichnet.

Caption - Falls angekreuzt, wird die Überschrift links vom Eingabefeld gezeichnet.

Action List

Sie wird ausgeführt, sobald das Eingabefeld auf einen neuen Wert gesetzt wird, entweder indem der Benutzer in das Feld tippt und TAB oder Enter drückt oder auf ein anderes Feld klickt, oder durch die "Set to" Action.

DDE Objekt 

Das DDE Objekt erlaubt die Kontrolle von und die Kommunikation mit anderen Windows Programmen, wie beispielsweise Tabellenkalkulationen, Textverarbeitungsprogrammen, Datenbanken, und mathematischen bzw. Grafik-Programmen.

Bitte beachten Sie, daß alle TestPoint Front-Panel Objekte (wie beispielsweise Data-Entry Felder, Auswahlregler, Schalter, Anzeige-, und Graphikobjekte) Kopieren und Einfügen Ihrer Daten in die Windows-Zwischenablage erlauben, genauso, wie es auch möglich ist, sogenannte "Links" einzubinden, die sich automatisch aktualisieren. Das DDE Objekt ist nicht für diese Funktionen notwendig. Das DDE Objekt wird dazu benutzt, den Datenaustausch mit anderen Programmen zu automatisieren, anstatt manuell zu verfahren.

Benutzerschnittstelle - keine

Aktionen

Start application - Startet eine andere Applikation. Benutzt das "Application name" Setting als Name der Programmdatei, die ausgeführt werden soll. Die Kommandozeilen Parameter können als Option spezifiziert werden.

Stop application - Stoppt eine andere Applikation (Schließen).

Set remote - Ändert den Wert eines sogenannten Items in einer anderen Applikation. Der Item Name ist ein String, dessen Wert an die benutzte Applikation angepaßt werden muß. Beispielsweise benutzt Microsoft Excel Reihen und Spaltennummern, um sich auf Tabellenkalkulationszellen zu beziehen (z. B. "Z1S1"). Der Datenwert, der übermittelt wird, kann optional durch einen Doppelklick auf das Parameterfeld und dem Eintrag eines Datentyps und -format, formatiert

werden. Nach der Formatierung werden die Daten via DDE "Text" Information gesendet.

Get remote - Lesen eines Wertes eines Items aus einer anderen Applikation. Optional können die Eingabedaten formatiert werden, indem auf den DDE Objektnamen doppelgeklickt wird. Hiermit werden die DDE "Text" Information übermittelt.

Execute remote - Senden eines Kommandostrings an eine andere Applikation. Der Inhalt dieses Strings hängt von der Applikation ab. Oft sind diese Strings Kommandos in der internen Makrosprache des anderen Programms.

Link remote - Erstellen eines sogenannten "hot-link" zu einem Item in einer anderen Applikation. Jedesmal, wenn sich das andere Item ändert, erhält das DDE Objekt ein Ereignis und führt seine Action List aus, dabei werden die Daten des DDE Objekts gleich den Daten des Items aus einer anderen Applikation. Optionale Eingabeformatierung kann erreicht werden, indem auf den DDE Objektnamen doppelgeklickt wird. Bitte beachten Sie: Falls ein Item sofort gelesen werden soll, ohne daß eine Änderung stattfindet, sollte die "Get remote" Aktion vor der "Link remote Action" verwendet werden. Nur ein anderes Item kann gleichzeitig an ein DDE Objekt gelinkt werden. Mehrere DDE Objekte können jedoch mit der selben Applikation verbunden sein.

Unlink remote - Aufheben jeglicher aktiven (Verbindungen) Remote Link. Es werden keine weiteren Update Benachrichtigungs Ereignisse weitergegeben.

Send keys to - Senden von Tastaturanschlägen an die andere Applikation, so als ob der Benutzer sie manuell getippt hätte. Dies erlaubt den Zugriff auf Programme, die sonst nicht DDE-fähig sind. Der Parameter String wird zeichenweise gesendet.

Jede Taste ist durch einen oder mehrere Zeichen charakterisiert. Um einen einzelnen Tastaturanschlag zu spezifizieren, sollte das Zeichen

selbst gebraucht werden. Die At (@), Caret (^), und Prozentzeichen (%) haben eine spezielle Bedeutung. Um eins dieser Zeichen zu spezifizieren, sollte es innerhalb von geschweiften Klammern geschrieben werden. Beispielsweise muß für das At Zeichen folgendes geschrieben werden {@}. Um ein { oder ein } Zeichen zu schreiben, sollte folgendes spezifiziert werden: {{} and {}}.

Um spezielle Zeichen, die nicht mit einer Taste darstellbar sind, zu verwenden (z. B. Enter oder Tab), oder andere Tasten, die eher Aktionen als Zeichen verwenden, muß folgender Code benutzt werden:

Taste	**Code**	**Taste**	**Code**
Backspace	{bs}	Caps Lock	{capslock}
Clear	{clear}	Del	{del}
down	{down}	End	{end}
Enter	{enter}	Esc	{esc}
Help	{help}	Home	{home}
Insert	{insert}	left	{left}
Num Lock	{numlock}	Page Down	{pgdn}
Page Up	{pgup}	right	{right}
Scroll Lock	{scrolllock}	Tab	{tab}
up	{up}	F-keys	{f1} to {f10}

Um eine Taste in Kombination mit Shift, Ctrl (Strg), und Alt zu spezifizieren, sollte dem normalen Zeichen folgender Tastencode vorausgehen:

Taste	Code
Shift	+
Strg	^
Alt	@

Für das Senden einer Kombination verschiedener Tasten mit Shift, Strg, und/oder Alt sollte das Zeichen in Klammern gesetzt werden.

Beispielsweise, um die Shift Taste mit gleichzeitig gedrücktem E und C zu drücken, sollte +(EC) verwendet werden. Um Shift gleichzeitig mit E, gefolgt aber von einem C ohne die Shift Taste zu drücken, muß +EC benutzt werden. Sich wiederholende Tasten können in folgender Form spezifiziert werden: {Taste Anzahl}, wobei immer ein Leerzeichen zwischen der Taste und der Anzahl sein muß. Beispielsweise bedeutet {left 42} 42mal die Cursor-nach-links-Taste; {x 10} bedeutet 10mal das Zeichen x.

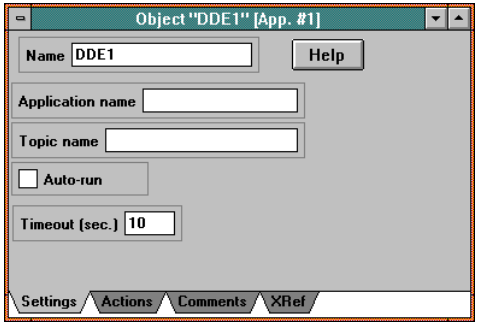
Achtung: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Durch die „Get remote“ Funktion werden dem DDE Objekt neue Daten zugewiesen, eine Aktualisierung kann aber auch automatisch durch eine Verknüpfung geschehen. Der Datentyp ist beliebig: number, string, vector, list, usw., je nach dem, wie die Datentyp-Referenz in der Action List Zeile spezifiziert ist.

Initial value - Keine Daten

Einstellungen (Settings)



Application name - Der Name der Applikation, zu der eine DDE Verbindung hergestellt werden soll. Der Name ist normalerweise gleich dem Namen der *.EXE Datei, ohne die Extension, z. B. Excel. Der Applikationsname kann (sollte) den Pfad zur Anwendung enthalten.

Topic name - Das Topic der DDE Verbindung, normalerweise der Dateiname oder das Dokument, auf das von TestPoint aus zugegriffen wird. Das Topic für Excel wäre z. B. „[HALLO.XLS]Tabelle1“, auch das Topic kann einen Pfad enthalten.

Auto-run - Bei der Aktivierung dieser Option wird die DDE Applikation automatisch bei der ersten DDE Aktion gestartet. Bei nochmaligem Start wird die DDE-Applikation ein zweites Mal geladen. Achtung Speicherbedarf !!!

Timeout - Maximale Zeit zur Ausführung einer DDE Aktion, bevor eine Fehlermeldung (time out) angezeigt wird.

Action List

Die Action List des Objektes wird abgearbeitet, wenn eine Verbindung aktiv ist und sich die Daten geändert haben. Die Daten des DDE Objektes sind gleich den neuen angeforderten Daten.

Beispiel - Borland Quattro Pro

Der Applikationsname für Quattro Pro ist „QPW“. Wenn Sie es wünschen, können Sie auch den kompletten Pfad angegeben (z. B. „E:\QP\QPW“).

Der Topic Name ist der Dateiname Ihres Tabellenblattes. Sie sollten unbedingt den vollen Pfadnamen verwenden, damit die Datei unabhängig vom momentan akutellen Verzeichnis gefunden werden kann (z. B. „D:\WORK\SHEET1.WB1“):

Quattro Pro benutzt die normalen Zellenbezeichnungen Ihres Tabellenblattes. Hier einige mögliche Formen:

A1
A5
A1..A10
B3..D5

Sie können als „item name“ den Namen eines Bereichs von Zellen verwenden. Sie können einen Breichsnamen vergeben, indem Sie die Zellen markieren, den Bereich anklicken und einen Namen eingeben.

Als Remote Befehle können Sie die Quattro Makro Kommandos verwenden. Als Beispiel „{Print.DoPrint}“ druckt den aktuellen ausgewählten Druckbereich aus. Quattro Pro hat eine sehr ausführliche On-Line Hilfe für Makro Kommandos.

Beispiel - Microsoft Excel

Der Applikationsname für Excel ist „EXCEL“. Sie können auch den kompletten Pfad mit angeben. Als Topic Name verwenden Sie den Dateinamen Ihres Tabellenblattes.
Für Excel ab Version 5.0 müssen Sie ein anderes Format verwenden, Sie müssen den Namen der Mappe in eckigen Klammern gefolgt von dem Tabellenblatt angeben, z. B. „[MAPPE1.XLS]Tabelle1“.

In Excel ist der „item name“ **nicht** gleich der Zellenbezeichnung. Sie müssen die jeweilige Zeile (Z) und Spalte (S) der Zelle angeben.
Hier einige Beispiel:

Z1S1
Z4S4..Z5S6

Als Remote Kommandos verwenden Sie die Excel Makro-Befehle. Die Befehle müssen grundsätzlich in eckige Klammern eingebunden

werden, z. B. lautet der Befehl, um markierte Zellen in die Zwischenablage zu kopieren „=COPY()“. Die Syntax für den Befehl aus einer DDE Verbindung heraus lautet dann „[COPY()]“.

Beispiel - Microsoft Word

Der Appliaktionsname für Word ist „WINWORD“. Sie können den kompletten Pfad optional angeben. Der Topic Name ist der Dateiname Ihres Word Dokumentes, z. B. „E:\REPORTS\TESTX.DOC“.

Bevor Sie Daten an ein Word Dokument übergeben können, müssen Sie eine oder mehrere Textmarken in Ihr Dokument einfügen. In Word plazieren Sie den Cursor an der entsprechenden Stelle und wählen aus der Menüleiste Einfügen/Textmarke, danach können Sie für die Textmarke einen Namen vergeben. Der Name der Textmarke ist der DDE „item name“. Als Beispiel, wenn Sie eine Textmarke mit dem Namen „Value“ erzeugen, ist der „item name“ in der TestPoint Action List Zeile „Value“.

Beispiel - Lotus 123

Der Applikationsname für 123 für Windows ist „123W“. Sie können optional den kompletten Pfad zur „123W.EXE“ Datei mit angeben.

Der Topic Name ist der Dateiname Ihres Tabellenblattes., z. B. „E:\123R4W\MINE.WK4“. Sie können keine Verbindung zu unbenannten Arbeitsblättern herstellen.

123 benutzt normale Zellenbezeichnungen als „item names“ für Tabellenblätter. Hier einige Möglichkeiten:

A1
A5
A:A1..A:A10
B3..D5

Als Remote Befehle können Sie die 123 Makro Kommandos benutzen. Sie müssen Makro Kommandos nicht nur in geschweifte Klammern „{}“ einzufassen, sondern sie auch noch in den DDE Befehl „[run()]“ einfügen. Als Beispiel:

"[run({SELECT B3..C4})]"

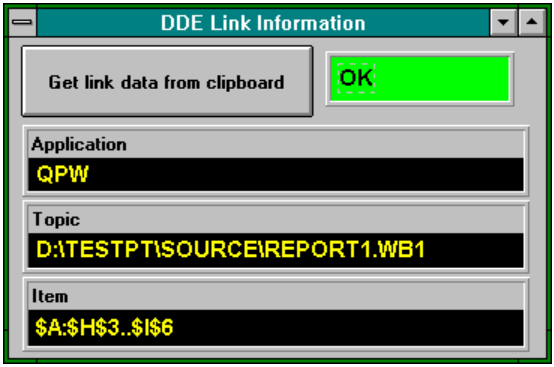
Anmerkung - Weitere Software

Was ist zu tun, wenn Sie andere Programme benutzen möchten, die DDE unterstützen und hier nicht beschrieben sind. Wie können Sie weiter vorgehen, wenn in der Beschreibung Ihrer Anwendung Informationen über „topic names“ und „item names“ fehlen?

Im Lieferumfang von TestPoint ist ein Programm enthalten: LINK.TST.

Alles was Sie machen müssen, ist die entsprechende Anwendung zu starten,einen Bereich Ihres Dokuments oder Tabellenblattes zu markieren und die Funktion „Bearbeiten/Kopieren“ aufzurufen.

Als nächstes starten Sie den TestPoint Editor und öffnen die Datei LINK.TST. Starten Sie das Programm. Wenn das andere Programm DDE unterstützt, sollten der entsprechende Applikationsname, der „topic name“ und „item name“ über die Windows Zwischenablage verfügbar sein. LINK.TST zeigt Ihnen diese dann an:


DDE Link Information
Get link data from clipboard
OK
Application
QPW
Topic
D:\TESTPT\SOURCE\REPORT1.WB1
Item
$A:$H$3..$I$6

Digital/Analog (D/A) Objekt 

Das D/A Objekt erlaubt Ihnen, Werte über eine entsprechende Einsteckkarte auszugeben. Es werden Karten von verschiedenen Herstellern unterstützt (siehe auch Kapitel zur Hardware-Konfiguration).

Benutzerschnittstelle - Keine

Aktionen

Start D/A - Startet die Ausgabe von Werten über die D/A Kanäle mit einer angegebenen Taktrate. Achtung: viele kombinierte A/D, D/A Karten ermöglichen nicht gleichzeitig getaktete A/D und D/A Operationen. Diese Aktion unterbricht dann jede aktive A/D Operation, die im Hintergrund auf der gleichen Karte läuft. Die Parameter für diese Funktion (Aktion) sind:

rate - die Taktrate in Hertz.

channel(s) - die Kanäle, über die Daten heraus gegeben werden sollen. Es kann eine einzelne Nummer für einen Kanal (erster Kanal = 0) sein oder eine Liste vom Typ String, wobei die einzelnen Kanäle durch Komma zu trennen sind. Es kann auch ein Vektor von Kanalnummern sein, der ein Ergebnis eines anderen Objektes ist.

values - die auszugebene Spannung. Sollte vom Format her ein Vektor vom Typ Nummer sein, z. B. das Ergebnis einer mathematischen Operation.

mode - entweder „ONCE“ oder „CONTINUOUS“. Bei „CONTINUOUS“ wird der Datensatz wiederholend ausgegeben.

Stop D/A - hält jede getaktete D/A Ausgabe an. Dient zum Beenden einer Ausgabe im „continuous“ Modus.

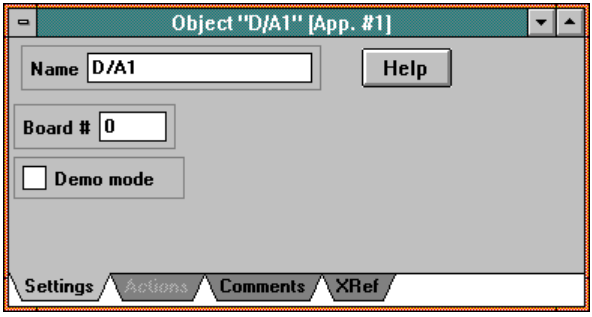
Output D/A - gibt einen einzelnen analogen Wert über einen D/A Kanal aus. Unterbricht jede getaktete D/A Ausgabe, stoppt aber nicht eine A/D Operation.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten - keine

Initial value - keine

Einstellungen (Settings)



Board number - welches D/A Board benutzt wird. Es können von TestPoint bis zu 4 A/D,D/A Boards unterstützt werden. Die Board-number bezieht sich auf die D/A Hardware Konfigurationsinformation in der TESTPT.INI Datei (siehe Kapitel Hardware Konfiguration).

Demo mode - falls angekreuzt, läuft der D/A Vorgang im Demo modus, anstatt daß auf richtige D/A Hardware zugegriffen wird.

Action List - keine

Beispiel

Mit einem Schieberegler- und einem D/A-Objekt kann folgende Action List des Schiebereglers erzeugt werden:

```
1) Output D/A      D/A1          once, channel=0, value=Slider1
```

Sobald der Schieberegler bewegt wird, verändert sich der D/A Output.

Beispiel - Kurvenformgenerierung

Durch die Hinzunahme eines Math Objekts namens "Wave" mit folgender Formula (Formel):

```
GenerateSin(128,128) * 3 + RndNormal(0,0.3,128)
```

die eine Sinuskurve mit addiertem Rauschen der Länge 128 erzeugt.

Dafür muß ein Pushbutton erzeugt werden, dessen Action List damit beginnt, die Kurvenform auszugeben:

```
1) Calculate       Wave
2) Start D/A       D/A1          rate=100 Hz, channel(s)=0,
                                 values=Wave, mode="continuous"
```

Digital I/O (DIO) Objekt 

Das DIO Objekt erlaubt die digitale Ein- und Ausgabe mittels einer digitalen I/O Hardware. Die meisten aller digitalen I/O Karten werden unterstützt, falls Sie den Standardbaustein Intel 8255 Digital I/O Controller Chip verwenden (siehe Hardware Konfigurationskapitel). Falls eine andere Hardware verwendet wird, kann das Port I/O Objekt verwendet werden.

Ein DIO Board in TestPoint hat einen oder mehr 24-bit digitale Interface Chips. Jeder dieser Interface Chips besitzt drei Kanäle (Ports): A, B, und C. Falls sich mehrere Interface Chips auf dem Board befinden, definiert sich die Syntax der Kanalbezeichnungen aus einer Zahl gefolgt von einem Buchstaben: 0A, 0B, 0C, 1A, 1B, 1C, 2A, 2B, etc...
Jeder 8-Bit Kanal kann sowohl für Ein- als auch für die Ausgabe konfiguriert werden.

Benutzerschnittstelle - keine

Aktionen

Configure - Setzt den spezifizierten Kanal auf Ausgabe- oder Eingabemodus.

Output to - Setzt den spezifizierten Kanal auf den angegebenen numerischen Wert. Falls der Wert außerhalb des Bereichs 0-255 (8 Bit) liegt, werden nur die untersten 8 Bit verwendet. Bitte beachten Sie: Falls ein Kanal für die Eingabe konfiguriert ist, hat diese Aktion keine Auswirkungen.

Input from - Lesen des spezifizierten Kanals, das Ergebnis dieser Aktion ist eine Zahl zwischen 0 und 255.

Set bits of - Ausgabe eines Wertes auf einem spezifizierten Kanal, indem aber nur die Bits gesetzt werden, die auf 1 in dem Mask Parameter gesetzt wurden. Die Bits, die 0 in dem Mask Parameter sind, bleiben unverändert.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des DIO Objekts ist der zuletzt eingelesene digitale Eingabewert. Der DIO Objektwert ist immer eine Zahl (oder no data).

Initial value - keine Daten

Einstellungen (Settings)

Board # - Die DIO Board Number. Dieses Setting bezieht sich auf die DIO Konfigurationsinformation in der TESTPT.INI Datei (siehe das Hardwarekonfigurationskapitel).

Demo mode - Falls angekreuzt, wird keine aktuelle DIO Hardware angesprochen, und die Eingabedaten werden von dem Demodatengenerator erzeugt.

Event channel, Event mask - Der DIO Kanal und die Bit Maske, die verwendet wird, um die Action List zu triggern. Sobald der selektierte Kanal (der auf Eingabe konfiguriert sein muß), logisch AND mit der Maske verknüpft, von 0 auf einen Wert nicht 0 geht, wird die Action List ausgeführt. Falls das Channel und/oder Mask Setting unbeschrieben ist, wird keine DIO Action List ausgeführt. Der ausgewählte Kanal wird 20 mal pro Sekunde abgetastet, um das Bitmuster zu prüfen.

Action List

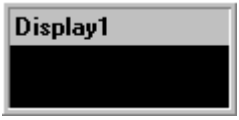
Wird ausgeführt, wenn ein selektiertes Bit oder mehrerer Bits auf 1 gehen (siehe "Event Channel und "Event Mask" Settings)

Display Objekt 

Das Display Objekt ist ein ausschließlich für die Ausgabe benutztes Objekt, das eine Zahl oder einen String auf dem Panel anzeigen kann.

Benutzerschnittstelle

Eine Caption (Beschriftung), die gleich dem Objektnamen ist, mit einer Anzeigefläche darunter. Die Anzeigefläche kann verschiedene Schriftgrößen und verschiedene Farben anzeigen. Die Caption und Bezel (Rahmen) sind optional.



Aktionen

Set - Kopiert die Parameterdaten auf das Display und zeigt sie an. Das Format der angezeigten Daten hängt von den Settings ab.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

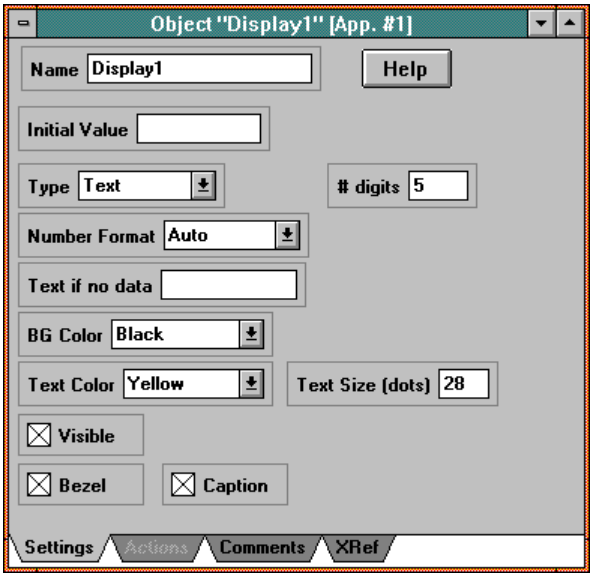
Daten

Der Datenwert des Objekts ist der letzte Wert in der Set Aktion.

Initial value

Der Initial Value des Displays kann in den Settings spezifiziert werden. Ein leerer, oder no data Eintrag, ist hierbei sinnvoll, falls man noch keinen gültigen Anzeigewert vorliegen hat.

Einstellungen (Settings)



Type - Numerisch oder Text. Falls das Setting Text ist, werden die 'number format' und '# of decimal places Settings' ignoriert.

Number format - Werden verwendet, wenn der Display Type numerisch ist. Die möglichen Formate sind Auto, Fixed, und Scientific. Auto Format zeigt nur die Digits an, die nicht Null sind, und wählt die Fixpunkt- oder wissenschaftliche Anzeige abhängig von der Größe des verwendeten Werts. Fixed Format benutzt einen Dezimalpunkt, aber keinen Exponenten. Scientific Format setzt immer genau eine Stelle vor den Dezimalpunkt und benutzt die "E+/-nn" Notation als Exponent.

 # digits - In Fixed Format spezifiziert dieses Setting die Anzahl der Stellen nach dem Dezimalpunkt. In Auto oder Scientific Format wird dadurch die Anzahl der signifikanten Stellen angegeben.

Text if no data - Falls das Display auf "no data" gesetzt wurde, entweder durch das Kopieren von Daten eines Objekts ohne Daten, oder indem die Set Action mit einem leerem Parameterfeld verwendet wird, kann ein beliebiger String für die Anzeige verwendet werden. Beispielsweise kann es sinnvoller sein, das Display mit dem String "---" anzuzeigen, als es leer zu lassen. Bitte beachten Sie, daß "no data" nicht die selbe Bedeutung hat wie 0.

BG color - Die Hintergrundfarbe, die für den Anzeigebereich verwendet wird.

Text color - Die Textfarbe, die für die Anzeige verwendet wird.

Text size - Die Textgröße in Punkten. Text wird immer in der Windows True Type "Arial" Schriftart angezeigt.

Visible - Kontrolliert, ob das Display auf dem Panel sichtbar ist.

Initial value - Jeder Wert kann in das Display bei der Programminitialisierung gesetzt werden.

Bezel - Falls angekreuzt, wird die Bezel (oder Rahmen) um das Objekt gezeichnet.

Caption - Falls angekreuzt, wird die Caption (Beschriftung) oberhalb des Anzeigebereichs angezeigt. Falls nicht, füllt der Anzeigebereich das gesamte Innere des Objekts.

Action List - keine

Error Handler Objekt 

Das Error Handler Objekt erlaubt das selbstdefinierte Abarbeiten von Runtime Fehlerroutinen, anstelle der Standard TestPoint Fehlermeldungen.

Sobald ein Fehler auftritt, wird die Action List des Error Handler Objekts ausgeführt, um den Fehler zu umgehen, oder eine alternative Fehlermeldung zu erzeugen.

Jedes Error Handler Objekt in einer Applikation kann so eingestellt werden, daß es auf gewisse mögliche TestPoint Fehler reagieren kann, demnach können unterschiedliche Fehlerbehandlungsroutinen vorgesehen werden.

Beim Auftreten eines Fehlers innerhalb einer Aktion eines Objekts sucht TestPoint nach einer gewissen Reihenfolge nach einem Error Handler Objekt: zuerst im selben Panel wie das fehlerauslösende Objekt, danach in allen Subpanels dieses Panels, und zuletzt im Hauptpanel der Applikation. Dies bedeutet, daß mehrere Error Handlers für den selben Fehler benutzt werden können, aber nur das am nahesten eingegliederte Objekt wird ausgewählt, den Fehler abzuarbeiten.

Benutzerschnittstelle - keine

Aktionen

Continue after - Diese Aktion veranlaßt TestPoint, die originale Action List auszuführen, sobald die Error Handler Action List beendet ist.

Stop after - Diese Aktion veranlaßt TestPoint, die originale Action List zu stoppen, sobald die Error Handler Action List beendet ist.

Retry after - Diese Aktion veranlaßt TestPoint, die Aktion, die den Fehler in der originalen Action List erzeugt hatte, noch einmal auszuführen.

Error message - Diese Aktion zeigt eine Fehlermeldung an und erlaubt dem Benutzer, entweder in der Action List weiterzufahren oder zu stoppen. Ein Action Parameter wählt entweder einen "Retry" oder einen "Continue" Knopf.

Cause error - Diese Aktion erzeugt eine Fehlerbedingung mit gegebener Error Code Nummer und optionalem zusätzlichen Fehlertext. Mit Hilfe dieser Aktion kann man TestPoint interne Fehlerbedingungen simulieren, oder eigene Fehler erzeugen, die durch einen Error Handler abgefangen werden. Die hierfür benutzbaren Fehlernummern dürfen im Bereich 20000-29999 liegen.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

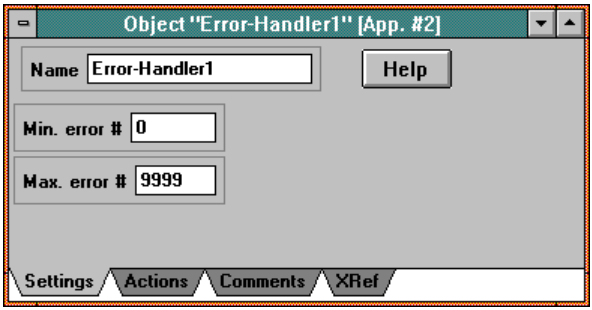
Daten

Der Datenwert des Error Handler Objekts wird gesetzt, sobald ein Fehler auftritt, und bevor die Action List ausgeführt wird. Der Datenwert besteht aus einer **list** folgender Elemente:

Fehlercode Nummer
Fehlermeldungsstring
Objektnamenstring
Actionnamenstring
Action List Objekt Namen String
Action Zeilen Nummer

Initial value - keine

Einstellungen (Settings)



Min. error # - Die kleinste Fehlercodenummer, die von dem Objekt behandelt werden soll.

Max. error # - Die höchste Fehlercodenummer, die von dem Objekt behandelt werden soll.

Bitte beachten Sie: Die Fehlercodenummern sind in der Datei ERRORNUM.DOC im TestPoint Verzeichnis aufgelistet.

Action List

Die Action List wird ausgeführt, sobald ein Fehler auftritt. Diese Action List sollte eine der Error Handler Objekt Aktionen beinhalten, um die Fehlerbehandlung auszuwählen. Falls keine dieser Aktionen angegeben wird, wird standardmäßig die Action List an der Stelle gestoppt, an der der Fehler aufgetreten ist.

Example

Dieses Fehler Objekt wird dazu benutzt, GPIB Timeout Fehler (Fehlernummer 251) abzufangen, indem sowohl die min. als auch die max. Fehlernummer in den Settings auf 251 gestellt wird. Die Action List muß folgendes Aussehen haben:

```
1)  Continue after    GPIB timeout        with Daten=0
```

Sie wird ausgeführt, falls ein Timeout auftritt. Sie benutzt die "continue after" Aktion, um die Standard TestPoint Fehlermeldung zu unterdrücken, und den Datenwert 0 anstatt des GPIB Datenwertes zurückzugeben.

File Objekt 

Das File Objekt greift auf Dateien auf Festplatte oder Diskette zu, und bietet eine Benutzerschnittstelle für die Dateinamensauswahl.

Benutzerschnittstelle

Der Dateiname wird auf dem Objekt gezeigt (möglicherweise in abgekürzter Form), mit einer Caption darüber, die den Objektnamen enthält. Ein Knopf namens "File" erlaubt dem Benutzer, einen neuen Dateinamen auszuwählen. Der Dateiauswahldialog wird von den Objekt Settings kontrolliert. Die Caption (Beschriftung) und Bezel (Rahmen) sind optional.

Aktionen

Open - Öffnet die Datei. Setzt ebenfalls den Eingabezeiger auf den Start der Datei.

Output to - Schreiben der Daten, die in dem Action Parameterfeld festgelegt sind, in eine Datei. Diese Aktion verwendet eine beliebige Anzahl an Parametern, und fügt neue leere Parameterfelder hinzu, sobald ein Feld aufgefüllt wird. Die Daten werden jeweils am Ende der Datei angehängt. Falls einmal nicht das Standardformat der Daten erforderlich ist, können die Formate durch ein Doppelklicken auf die Parameterfelder geändert werden.

Input from - Auslesen der Daten einer Datei. Die Action Parameter spezifizieren die maximale Anzahl an Zeichen oder Zeilen, die gelesen werden sollen, einschließlich eines optionalen Endezeichens. Der sogenannte Terminator kann jegliches Zeichen sein, üblicherweise werden aber folgende speziellen Codes verwendet: CR, LF, CRLF,

NONE, oder TAB. Die Daten, die eingelesen wurden, werden zum Datenwert des File Objekts. Falls einmal nicht das Standardformat der Daten erforderlich ist, können die Formate durch ein Doppelklicken auf den Namen des Objekts in der Action List geändert werden. Die Daten werden jeweils an der laufenden Eingabeposition innerhalb der Datei eingelesen, demnach wird die nächste Einleseoperation an dem Punkt fortgeführt, an dem die vorige gestoppt hat.

Close - Schließt die Datei, und schreibt alle Daten im TestPoint internen Dateipuffer in die Datei. Setzt ebenfalls die Eingabeposition auf den Anfang der Datei.

Erase - Löschen der Datei von Festplatte oder Diskette. Da die Dateiausgabe Daten immer an das Ende einer Datei anhängt, muß für die komplette Neuerstellung einer Datei zuvor die Erase Action ausgeführt werden, bevor wieder neu geschrieben werden kann.

Set filename of - Setzen des Dateinamens für das File Objekt. Diese Aktion ist equivalent zu dem benutzeraktivierten Drücken des File Knopfes und der anschließenden Auswahl einer Datei. Nach der Zuweisung wird die Action List ausgeführt. Ebenfalls wird der Eingabezeiger auf den Beginn der neuen Datei gesetzt.

Get filename of - Auslesen des Dateinamens des File Objekts. Somit kann der Benutzer-gewählte Dateiname als Datenwert für weitere Programmschritte verwendet werden. Dies ist beispielsweise dann hilfreich, wenn ein Math Objekt verwendet werden soll, um den Dateinamen zu modifizieren und automatisch einen anderen Namen aus den Benutzerangaben zu generieren.

Does file exist - Testet, ob die gegenwärtig ausgewählte Datei exisitiert. Der Datenwert des File Objekts wird auf 1 gesetzt, falls die Datei existiert, bzw. 0 falls nicht.

Set file position - Verändert die Position des Lesezeigers, von dem aus mit der nächsten **Input from** Aktion gelesen wird. Dieser Zeiger

wird in der Anzahl von Bytes gezählt vom Start der Datei spezifiziert. Normalerweise hinterläßt jede Leseoperation diesen Lesezeiger an der Position des nächsten Bytes, diese Aktion erlaubt dem Benutzer jedoch einen wahlfreien Zugriff auf den Inhalt jeglicher Datei. Besonders nützlich ist diese Aktion für den wahlfreien Zugriff auf Binärfiles.

Get file size - Mit dieser Aktion wird der Datenwert des File Objekts gleich der Anzahl der Bytes in der Datei (0 falls die Datei nicht existiert - um diesen Fall zu prüfen, kann die "Does file exist" Action verwendet werden).

Test EOF - Setzt den Datenwert des File Objekts auf 1 (wahr), falls die Eingabeposition der Datei momentan am Ende der Datei (end-of-file), oder 0 (falsch), falls nicht. Diese Aktion kann dafür benutzt werden, Dateien mit Hilfe von Schleifen auszulesen, die eine variable Länge an Zeichen beinhalten.

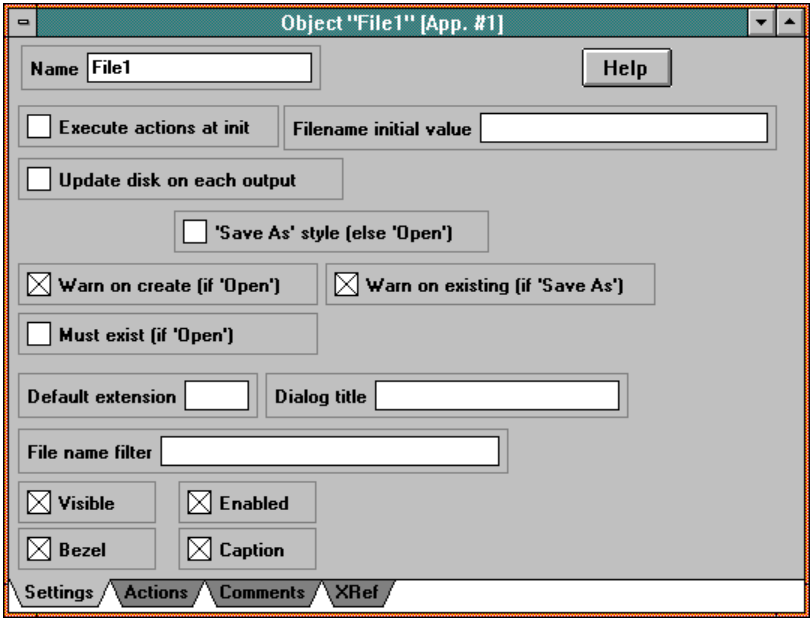
Get file number - Setzt den Datenwert des File Objekts gleich der DOS "File Handle Zahl" für die verwendete Datei. Dies funktioniert nur, falls die Datei geöffnet wurde, diese Handle Zahl wird ungültig, falls die Datei geschlossen wird. Eine File Zahl von 0 sagt aus, daß die Datei nicht geöffnet ist, oder daß keine Datei gefunden wurde. Diese Dateinummern können dafür benutzt werden, Parameter für direkte Aufrufe zu Low-Level Routinen in einigen Programmiersprachen zu verwenden. Demnach ist diese Aktion sinnvoll für die Benutzung mit externem Code (Code Objekt), oder Custom Controls (VBX Objekt).

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten - the result of the most recent input operation.

Initial value - keine Daten

Settings



Filename initial value - Jeder gewünschte Dateiname kann bei der Initialisierung gesetzt werden. Es muß hier kein Eintrag vorhanden sein.

Update disk on each output? - Falls angekreuzt, wird die Datei nach jeder Schreiboperation geschlossen. Falls nicht, wird ein effizienter Dateiupdate gemacht, sobald genug Daten in dem TestPoint Zwischenspeicher sind, oder die Datei geschloßen wird, oder die Applikation beendet wird. Diese Option ist nützlich, um Datenverlust in einer lang laufenden Applikation zu verhindern, sobald der Rechner während des Programmablaufs ausgeschaltet wird.

Save As style? - Kontrolliert das Aussehen des Dateiauswahldialogs, der am Bildschirm erscheint, sobald der Bediener den File Knopf drückt. Ein Open File Dialog sollte für eine Einleseoperation, der Save As Stil sollte für Ausgabedateien verwendet werden.

Warn on create? - Für das Einlesen von Dateien (Save As Stil nicht angekreuzt) kann mit Hilfe dieser Einstellung der Bediener gewarnt werden, wenn die Datei noch nicht existiert und neu erzeugt werden muß?

Must exist? - Für das Einlesen der Daten (Save As Stil nicht angekreuzt) kann hiermit festgelegt werden, ob der Bediener gezwungen werden soll, nur schon existierende Dateien zu verwenden?

Warn on existing? - Für das Schreiben von Dateien (Save As style angekreuzt) kann festgelegt werden, ob der Bediener gewarnt werden soll, wenn die Datei schon existiert. Bitte beachten Sie, daß die Daten an schon existierende Dateien angehängt werden, und nicht die vorhandene Datei gelöscht wird.

Default extension - Falls der Bediener einen Dateinamen ohne eine Extension eingibt, welche Extension soll dann benutzt werden? Falls kein Eintrag, wird auch keine Extension angehängt.

Dialog title - Hiermit kann ein Titel für den Fileauswahldialog spezifiziert werden. Falls kein Eintrag, wird der Standardtitel verwendet, der abhängig von dem gewählten Save As Style ist.

File name filter - Falls hier kein Eintrag ist, erscheinen alle Dateien (*.*) in der Auswahlliste. Mit diesem Eintrag kann eine Liste von Dateitypen und Dateispezifikationen angegeben werden, um die Anzahl der angezeigten Dateien zu limitieren. Die verschiedenen Dateitypbezeichnungen und Wildcard Spezifikationen sollten mit senkrechten Strichen getrennt werden "|". Beispielsweise kann folgende Form verwendet werden: "Daten Files|*.DAT|Alle Files|*.*|".
Die Filter Spezifikation sollte ebenso mit einem "|" enden.

Visible - Gibt an, ob die Benutzerschnittstelle auf dem Panel erscheint. Falls keine Benutzerauswahl stattfinden soll und auch der Dateiname nicht sichtbar gemacht werden soll, sollte das Objekt unsichtbar gemacht werden.

Enabled - Kontrolliert, ob der File-Knopf aktiv ist. Falls nicht, kann der Benutzer nicht neue Dateien auswählen.

Execute actions at init - Falls angekreuzt, wird die Action List bei der Programminitialisierung abgearbeitet.

Bezel - Falls angekreuzt, wird die Bezel (Rahmen) um das Objekt gezeichnet.

Caption - Falls angekreuzt, wird die Caption (Beschriftung) oberhalb des Dateinamens geschrieben.

Action List

Die Action List wird ausgeführt, sobald ein neuer Dateiname ausgewählt wurde.

Bemerkungen

Falls der Dateiname (entweder durch das "initial filename" Setting oder die "Set filename" Action) einen Wert zugewiesen bekommt, der **nicht** die volle Laufwerks- und Pfadangabe enthält (z. B. "test.dat"), dann sucht TestPoint nach der Datei in der folgenden Reihenfolge: dem aktuellen Verzeichnis, dem Windows Verzeichnis, dem Windows\system Verzeichnis, dem Verzeichnis, von dem der TestPoint Code geladen worden war, dem DOS PATH, und schließlich jedem Netzwerkpfad, der im Dateimanager angegeben worden ist.

Beispiel

Dieses Beispiel liest Daten von einer ausgewählten Datei und stellt sie graphisch dar, sobald der Benutzer eine Datei auswählt. Die Action List für das File Objekt lautet:

1)	Input from	File1	up to 32768 bytes, stopping at __
2)	Draw Graph	Graph1	with File1

Zeile 1 liest die Daten aus der Datei. Es wird die Standard Datenformatierung verwendet (falls auf das "File1" doppelgeklickt wird, zeigt das erscheinende Fenster den Datentyp ANY), die TestPoint automatisch den Datentyp in der Datei bestimmen läßt.

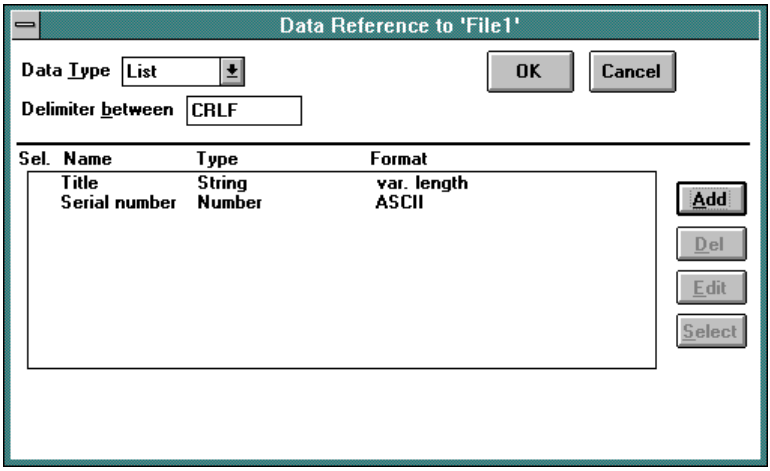
Dieses Beispiel findet sich auch als EXAMPLE5.TST in dem TESTPT\EXAMPLES Verzeichnis.

Beispiel - Einlesen einer Datendatei mit einem Dateikopf (Header)

Manchmal besitzen Datenfiles einige Zeilen Kopfinformation, da dieses Dateiformat auch von anderen Programmen verwendet wird. Im Anschluß an diese Kopfinformation der Datei sind dann die eigentlichen Daten in Reihen und Spalten aufgelistet. Um so eine Datei einzulesen, sollte folgende Action List benutzt werden:

1)	Input from	File1*	up to 2 lines, stopping at __
2)	Set	Display1	to File1:Title
3)	Input from	File1	up to 32768 bytes, stopping at __
4)	Draw Graph	Graph1	with File1

Zeile 1 liest den Dateikopf, der in diesem Fall zwei Zeilen lang ist. Hierbei wird ein spezielles Datenformat angewandt, indem auf den "File1" Eintrag in Zeile 1 doppelgeklickt wird:



Das Format erlaubt es, aus der Datei individuelle Datenelemente aus dem Header herauszulesen, wie es in Zeile 2 realisiert ist. Der Parameter am rechten Ende von Zeile 2 wird dadurch erzeugt, indem die File1 Referenz aus Zeile 1 markiert, dann doppelgeklickt und der Select Knopf verwendet wird, um das Titelfeld zu spezifizieren. Die Zeilen 3 und 4 lesen und zeichnen die Graphik aus dem Rest der Datei, ebenso wie im letzten Beispiel gezeigt.

Beispiel - Konfiguration-/Kalibrierungsdateien

In einigen Applikationen gibt es eine Gruppe ähnlicher Tests oder Messungen, die sich nur in einigen Parameterwerten oder in Kalibrierkonstanten für die verwendeten Sensoren unterscheiden.

Dieses Beispiel nimmt an, daß die analogen Messungen die Signale eines Drucksensors aufnehmen. Jeder Transducer besitzt seine eigenen Kalibrierkonstanten, die dazu erforderlich sind, den Meßwert zu korrigieren und die richtige Druckzahl zu ermitteln. Der Test Operator benötigt nun eine Methode, die Seriennummer des Transducers zu ermitteln, der gerade vermessen wird.

Ein File Objekt wird für die Speicherung einer Kalibrierdatei verwendet. Jeder Transducer wird durch ein Kalibrierfile charakterisiert. In diesem Beispiel nehmen wir an, daß zwei Zahlen verwendet werden, um jeden Transducer zu charakterisieren, somit enthält jede Datei zwei Zeilen.

Das File Objekt wird "Transducer file" genannt und auf einen Standard File Namen gesetzt. Bei gesetztem Setting Execute at initialization wird dann folgende Action List aufgerufen:

1) Input from Transducer file up to 2 lines, stopping at __

Sobald die Applikation startet, und jedesmal, wenn der Benutzer eine neue Transducer Datei auswählt, werden diese Action List ausgeführt und die Kalibrierwerte in das File Objekt eingelesen.

Irgendwo in der Applikation werden Messungen vorgenommen. Ein Math Objekt namens "Pressure" korrigiert die reinen Meßwerte, indem es eine lineare Korrektur mittels folgender Formel durchführt:

```
cal[0] * input + cal[1]
```

Hier ist die Beispiel Action List, die die A/D Wandlung durchführt und den entsprechnenden Wert korrigiert:

1)	Sample A/D	A/D1	once, channel(s)=0
2)	Calculate	Pressure	with cal=Transducer file, input=A/D1

Beispiel - Variable Dateinamen

In einigen Applikationen ist es erforderlich, einen Dateinamen automatisch zu erstellen, anstatt den Benutzer eine Datei auswählen zu lassen. Beispielsweise ist es möglich, sequentielle Dateien zu benennen, deren Namen mit den Ziffern 1 bis 9 enden. Um dies zu realisieren, sollte ein Math Objekt verwendet werden, um den notwendigen Dateinamenstring zu erzeugen, gefolgt von einer Set filename Action:

1)	Linear series	Loop1	from 1 to 9
2)	Calculate	Name	with n=Loop1
3)	Set filename	File1	to Name
4)	Execute	Action1	
5)	Output to	File1	with Results
6)	Close	File1	
7)	End	Loop1	

wobei die Formel für "Name" ist: **"FILE" & n & ".DAT"**

GPIB Objekt

Das GPIB Objekt repräsentiert ein externes Gerät, das durch eine GPIB-, oder IEEE-488 Schnittstellenkarte an den PC angebunden ist.

Benutzerschnittstelle

Ein Eingabefeld, das es erlaubt, die GPIB Addresse des Gerätes einzugeben. Die Caption ist der Name des Objekts, die von dem Wort "address" gefolgt ist. Falls Sekundäradressen verwendet werden, wird die Primary Addresse eingegeben, gefolgt von einem Komma und der Sekundäradresse. Die Caption (Beschriftung) und Bezel (Rahmen) sind optional.

Aktionen

Output to - Sendet Daten an ein Gerät. Diese Aktion nimmt die angegebenen Parameter und sendet diese an das Gerät. Es verwendet ebenfalls Endezeichenparameter, die standardmäßig auf den Wert gesetzt werden, die die Settings angeben (gewöhnlich line feed, oder LF). Der "eoi=" Parameter kontrolliert, ob das letzte gesendete Zeichen das EOI Signal setzt, um anzuzeigen, daß das Ende der GPIB Übertragung erreicht ist. Falls Nicht-Standard-Formatierung verwendet werden soll, können die Formate durch Doppelklicken auf den Datenparameter festgelegt werden.

Enter from - Liest Daten von dem Gerät und ändert den Datenwert des GPIB Objekts entsprechend ab. Als Parameter wird die maximale Anzahl an Bytes spezifiziert, die gelesen werden soll und ein optionaler Terminator, der ein beliebiges Zeichen sein kann, oder ein spezieller String CR, LF, CRLF, NONE, oder TAB. Falls Nicht-Standard-Formatierung verwendet werden soll, können die Formate durch

Doppelklicken auf den GPIB Objektnamen in der Action Line spezifiziert werden.

Trigger - Sendet ein GPIB Trigger Kommando an das Gerät.

Clear - Sendet ein GPIB Selected Device Clear Kommando an das Gerät.

Serial poll - Benutzt eine GPIB Serial Poll (Abfrage) Operation, um das Status Byte des Geräts auszulesen. Der Datenwert ist eine Zahl von 0 bis 255.

Parallel poll - Führt eine GPIB Parallel Poll (Abfrage) Operation durch und setzt die Objektdaten auf das Ergebnis, das eine Zahl im Intervall 0 bis 255 ist.

Local - Setzt das Gerät in den Local Modus, der eine manuelle Bedienung des Gerätefrontpanels ermöglicht.

Remote - Setzt das Gerät in den Remote Modus, für Computerkontrolle. Ein Parameter spezifiziert, ob ebenso das GPIB Kommando Local Lockout verwendet werden soll, das das Front-Panel des Gerätes für die manuelle Eingabe blockiert.

Transmit GPIB commands - Senden einer beliebigen Sequenz von GPIB Kommandos. Diese Aktion führt Kommandos aus, wie sie im selben Format der Dokumentation zu dem CEC IEEE-488 (KPC-488.2) Programmier Referenz Handbuch für die Transmit Prozedur beschrieben sind.

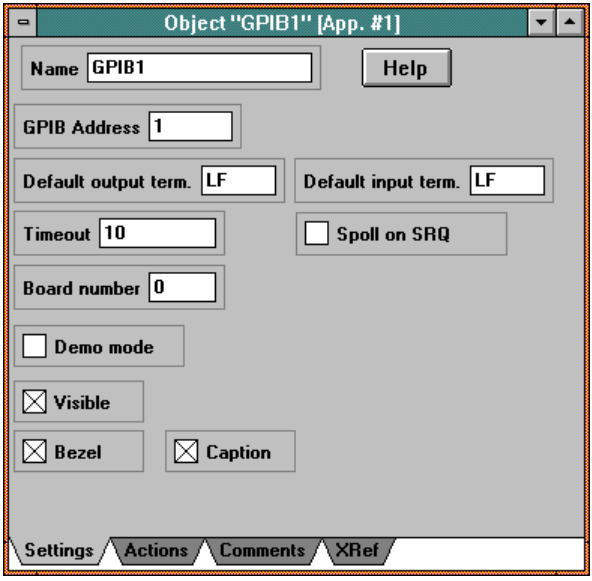
Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des Objekts ist das Resultat der letzten Einleseoperation: entweder Enter from, oder Serial poll, oder Parallel poll.

Initial value - keine Daten

Einstellungen (Settings)



GPIB Address - Die Busaddresse des Geräts, von 0 bis 30. Falls eine Sekundäraddresse erforderlich ist, wird die Primäradresse mit einem Komma getrennt von der Sekundäradresse gefolgt. Der Bediener kann diesen Wert während der Ausführung des Programms ändern, indem er neue Werte in das Eingabefeld einträgt.

Default output term. - Das Endezeichen, das standardmäßig eingefügt wird, sobald neue "Output to" Action Lines erzeugt werden. Falls ein

spezielles Gerät sowohl den Carriage Return als auch den Line Feed Parameter benötigt, sollte das Setting beispielsweise CRLF lauten, um standardmäßig dieses Endezeichen vorzuschlagen.

Default input term. - Das Endezeichen, das standardmäßig gesetzt wird, wenn eine neue "Enter from" Aktions-Zeile erzeugt wird.

Timeout - Die Timeout Dauer in Sekunden, wenn mit diesem Gerät kommuniziert wird. Diese Einstellung ist ungefähr auf 55 msec Auflösung genau. Gültige Werte für den Timeout liegen im Bereich 100 msec bis 65 Sekunden.

Spoll on SRQ? - Falls angekreuzt, führt das Objekt automatisch eine Serial Poll (Abfrage) Operation aus, wenn ein Service Request (SRQ) Ereignis auftritt.

Board number - Die GPIB Board Nummer, an die die Informationen der IEC-Karte geknüpft sind. Das erste GPIB Board ist Nummer 0. Diese Nummer korrespondiert zu der GPIB Board Configuration Information in der TESTPT.INI Datei (siehe Hardware Konfigurationskapitel).

Visible - Gibt an, ob die Benutzerschnittstelle für das Objekt sichtbar ist.

Demo mode - Falls angekreuzt, wird nicht auf richtige IEC-Bus Hardware zugegriffen. Einleseoperationen benutzen in diesem Fall den Demodaten-Generator.

Bezel - Falls angekreuzt, wird die Bezel (Rahmen) um das Objekt gezeichnet.

Caption - Falls angekreuzt, wird die Caption (Beschriftung) links vom Adress-Eingabefeld geschrieben. Die Caption besteht aus dem Objektnamen gefolgt von "GPIB address".

Action List

Die Action List wird ausgeführt, wenn ein SRQ (Service Request) Signal am GPIB-Bus auftritt, und das "Spoll on SRQ?" Setting angekreuzt, und es sich um das Gerät handelt, das Service erfordert. Die GPIB Objektdaten werden in diesem Fall auf das Ergebnis der Serial Poll Operation gesetzt, bevor seine Action List ausgeführt wird.

Beispiel - Keithley 2001 Multimeter

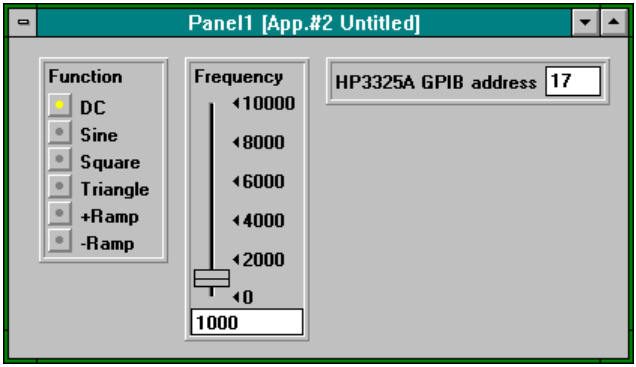
Diese Action List veranlaßt das Multimeter, eine DC Spannungsmessung durchzuführen, anschließend wird das Ergebnis im TestPoint Panel angezeigt:

1)	Output to	DMM	with ":MEAS:VOLT:DC?" term.=LF send EOI?=1
2)	Enter from	DMM	up to 256 bytes, stop on EOS=LF
3)	Set	Display1	to DMM

Dieses spezielle Gerät gibt einen String zurück, der folgendermaßen aussehen kann: "NDCV+3.45678E+00" mit einem CRLF am Ende. Falls nur der reine Zahlenwert verwendet werden soll, um beispielsweise Berechnungen damit auszuführen, kann TestPoints Datenformatierung dazu verwendet werden, um eine Zahl durch die "Enter from" Action einzulesen. Dafür muß auf auf das "DMM" in Zeile 2 doppelgeklickt werden und der Datentyp "Number" ausgewählt werden.

Beispiel - HP 3325 Funktionsgenerator

Dieses Beispiel bietet die Steuerung eines Funktionsgenerators durch mehrere Objekte auf einem TestPoint Panel:



Die Label- und Value-Settings des "Function" Auswahlschalter sind:

labels:	DC,Sine,Square,Triangle,+Ramp,-Ramp
values:	0,1,2,3,4,5

Die Action List des "Function" Auswahlschalters ist:

1)	Output to	HP3325A	with "FU", Function term.=LF send EOI?=1

Die Action List des "Frequency" Schiebereglers ist:

1)	Output to	HP3325A	with "FR", Frequency, "HZ" term.=LF send EOI?=1

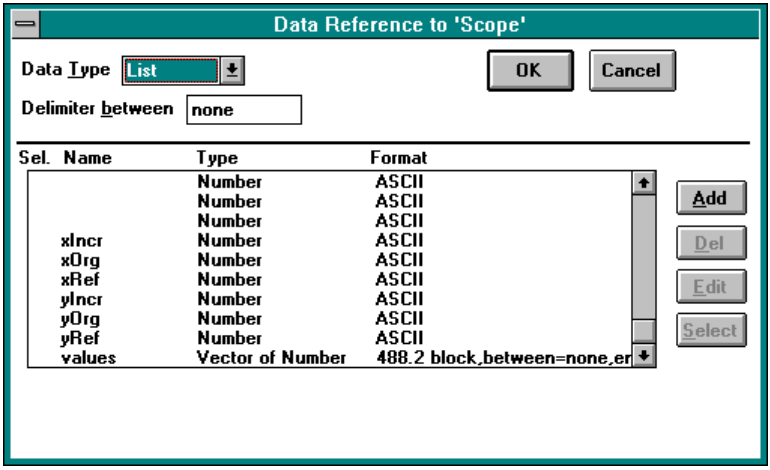
Beispiel - HP 54502 Oszilloskop

Dieses Beispiel liest eine Kurvenform eines Scopes ein. Das HP54502 Scope gibt seine Kurvendaten in Zahlen zwischen 0 und 65535 zurück, die skaliert und mit einem Offset als gültige Spannungswerte versehen werden müssen. Die Skalierungsinformation muß demnach aus dem Scop gelesen werden.

Hier ist die Action List:

1) Output to	Scope	with ":wav:form word;pre?;Daten?"
2) Enter from	Scope*	up to 4096 bytes
3) Calculate	Waveform	with Daten=Scope:values,
y0=Scope:ref,		
		scale=Scope:yincr,
offset=Scope:yorig		
4) Draw Graph	Graph1	with Waveform

Zeile 2 definiert den Einlesedatentyp und die zugehörige Formatierung, wie es unten gezeigt ist. Das Scope sendet 10 ASCII Zahlen als "prefix", oder Skalierungsinformation, gefolgt von den gewünschten Kurvenformdaten im Binärformat.



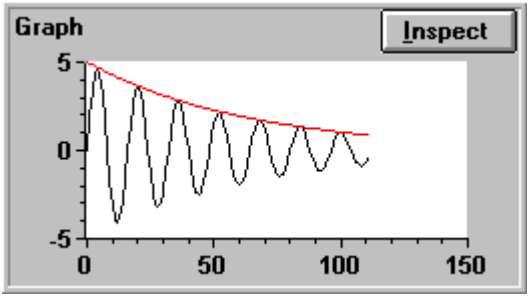
Dieses Beispiel befindet sich im EXAMPLE Verzeichnis unter dem Dateinamen HPSCOPE.TST.

Graph Objekt

Das Graph Objekt zeigt Daten in einem XY Graphik Format, wobei eine breite Palette an Variationen möglich ist. Die Graphen können sogenannte Strip Charts (Schreibermodus) sein (jeweils mit neuen Daten erneuerte Graphen), oder statische Graphen (gleichzeitiges Zeichnen aller Daten eines Vektors).

Benutzerschnittstelle

Der Graph wird mit einer Caption (Beschriftung) in der oberen Hälfte gezeichnet, die gleich dem Objektnamen ist.
Grafiken sind bei der Programminitialisierung leer, bis eine Aktion dafür verwendet wird, eine neue Graphik zu zeichnen.
Falls der Benutzer auf den Graphen doppelklickt, kann er die Graph Settings ändern, wie beipsielsweise Trace Farben, Achsenskalierung, etc.
Es gibt ebenfalls einen "Inspect" Knopf, der ein neues Fenster mit einer Kopie des Graphen erstellt. Dieses Inspect Fenster kann zoomen, scrollen, das Aussehen des Graphen verändern, den Graphen drucken, etc. Es ist ebenfalls möglich, auf einen Trace (Linie) in dem Inspect Fenster doppelzuklicken und die Daten in Tabellenform anzusehen.



Aktionen

Draw graph - Zeigt den Graphen an, in dem die Daten verwendet werden, die in den Parameterfeldern spezifiziert werden. Die Parameterdaten sollten als Vector of numbers oder als List of vectors of numbers (vielleicht schon zusammengesammelt in einem Container Objekt) vorliegen.

Clear graph - Setzt das Graph Objekt auf ein leeres Fenster zurück. Diese Aktion ist dann sinnvoll, wenn eine Applikation lange Zeiträume enthält, in denen neue Daten generiert werden, und die alten Daten nicht mehr gültig sind und deshalb auch nicht mehr gezeigt werden sollen. Sie sollte ebenfalls angewandt werden, wenn mehrere Settings des Graphen in einer Action List geändert wurden, um zu verhindern, daß die Grafik mehrere Male hintereinander neu gezeichnet wird. In diesem Fall schaut es besser aus, wenn der Graph gelöscht und die Daten neu gezeichnet werden.

Add point(s) to - Wird dazu benutzt, um einen oder mehrere Datenpunkte an das Ende des Graphen anzufügen. Die Daten können entweder eine Zahl (ein einzelner Punkt), oder ein Vector (mehrere Punkte, die zur selben Zeit angefügt werden), oder eine Liste von entweder mehreren Traces sein und angefügt werden. Im Strip Chart Modus scrollt der Graph nach links, sobald neue Punkte angefügt werden (Linienschreiber Modus).

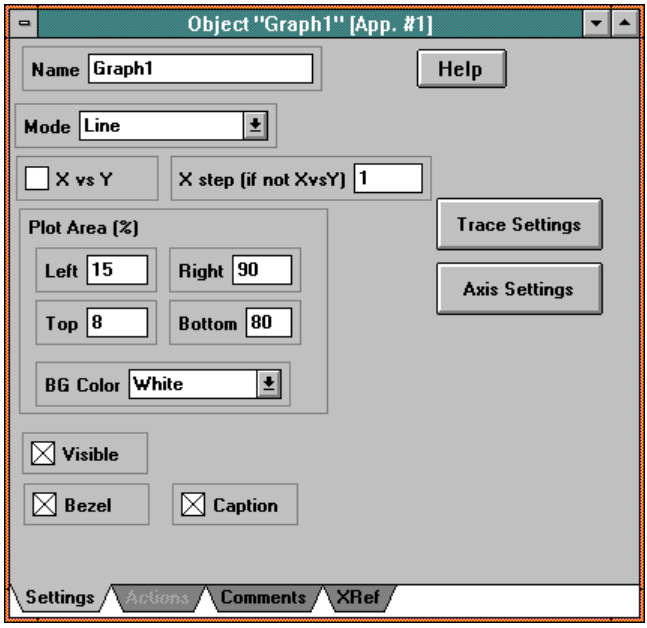
Print - Drucken des Graphen am momentan ausgewählten Windows Drucker.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster. Falls eine Achseneinstellung, beispielsweise die "from" oder "to" Einstellung, geändert werden soll, muß zuerst das "Axis" Setting für die gewünschte Achse ausgewählt werden: X, Y1, Y2, etc. Falls ein Trace Setting, wie beispielsweise die "Width",

geändert werden soll, muß zuerst das gewünschte "Trace" Setting auf die entsprechende Trace Number von 1 bis 8 gesetzt werden.

Daten - keine

Einstellungen (Settings)



Mode - Line, Strip Chart, Bar Chart, oder XY Pairs. Line, Bar, und XY Pairs Modi sind dafür da, daß die Grafik sofort mit einem oder mehreren Vektoren gezeichnet wird. Der Strip Chart Modus ist für eine ständige Erneuerung der Graphen, die nach links scrollen, und den X-Achsen Bereich ändern, sobald Punkte hinzugefügt werden. Der Bar Chart Modus zeichnet die erste Trace nur als Balkengrafik.

Wenn der Mode XY Pairs eingeschaltet ist, und eine Liste mehrerer Vektoren an die Draw Graph Action gegeben werden, werden die Daten als eine alternierende Sequenz von X Daten, Y Daten, X Daten, Y Daten, etc verwendet. In diesem Modus wird das "X vs Y" Setting ignoriert.

X vs Y - Falls angekreuzt, und falls die Daten in der Action eine Liste von mindestens zwei Elementen sind, wird das erste Element als X-Achse Daten benutzt. Falls nicht, werden alle Daten auf die Y-Achsen bezogen und gegen eine lineare X-Sequenz aufgetragen.

X step - Das Inkrement zwischen den einzelnen X Werten, die der Graph benutzen soll, wenn er nicht im X vs. Y Modus ist. Bitte beachten Sie: Der Startwert der X-Achse wird in den Axis Settings festgelegt.

Plot area - Der Anteil der Graph Objekt Fläche, die dür den Datenplot verwendet wird. Die oberste Zeile ist dabei immer für die Caption reserviert und wird nicht in diese Aufteilung mit einbezogen. Die Plotbereich Settings allokieren ebenfalls Platz für die Achsenlabel.

BG color - Die Hintergrundfarbe innerhalb des Datenplotbereichs. Die Farbe der Ränder außerhalb des Plotbereichs ist immer hellgrau, der Datenplotbereich jedoch kann auf jede Farbe gesetzt werden.

Visible - Kontrolliert, ob der Graph auf dem Panel sichtbar ist.

Bezel - Falls angekreuzt, wird die Bezel (Rahmen) um das Objekt gezeichnet.

Caption - Falls angekreuzt, wird die Caption (Beschriftung) am linken oberen Rand des Grafikobjekts plaziert.

Trace Einstellungen (Settings)

Trace number - Auswahl des Trace (der Linie), auf welchen sich die weiteren Settings beziehen. Traces mit den Nummern 1 bis 8 sind verwendbar. Ein Graph kann bis zu 32 Traces besitzen, aber nur die 8 Trace Settings werden für jeweils Gruppen von 8 Traces benutzt.

Trace line - Setting, um die Farbe, Stil, und Breite der Linien zwischen den aufeinanderliegenden Datenpunkten festzulegen. (Für eine Balkengrafik ist die Breite des Trace 1 angegeben in Einheiten der X-Achse. In allen anderen Fällen handelt es sich dabei um Display Pixel.). Eine Breite (Width) von 0 bedeutet, die kleinste mögliche Anzeigebreite zu verwenden.

Markers - Settings, um das Aussehen, die Farbe, und die Größe der verwendeten Marker der Datenpunkte zu bestimmen.

Y Axis - Bestimmt, welche Y-Achse mit welchem Datentrace verbunden ist. Es sind bis zu vier Y-Achsen verwendbar. Sobald mehrere Y-Achsen verwendet werden, können verschiedene Traces unterschiedlichen Achsen zugeordnet werden und damit unterschiedliche vertikale Skalierungsfaktoren verwendet werden. Bitte beachten Sie, daß ein Trace (eine Linie), der mit einer Achse verbunden ist, die nicht sichtbar geschaltet ist, nicht angezeigt wird.

Axis Settings

Axis - Auswahl der Achse, auf die sich die weiteren Settings beziehen. Eine X-Achse und vier Y-Achsen sind möglich.

On? - Gibt an, ob die jeweilige Achse gezeichnet wird. Im Falle der Y-Achsen, wird durch dieses Setting ebenfalls der Trace an- bzw. abgeschaltet.

From - Der Startwert der Achse, falls der Auto Mode abgeschaltet ist. In dem Fall einer X-Achse, falls der X vs Y Mode abgeschaltet ist,

repräsentiert dieses Setting den Startwert der X-Werte, gegen die die Daten aufgetragen werden.

To - Der Endwert der Achse, falls der Auto Mode abgesschaltet ist.

Auto? - Falls angekreuzt, wird die Achse basierend auf den aktuellen Datenwerten automatisch skaliert. Die Minimum- und Maximumwerte werden verwendet, um die Achsenbegrenzung festzulegen. Die Grenzen werden dabei abgerundet, um ein gleichmäßig verteiltes Aussehen der Achse zu garantieren. X-Achsen Autoskalierung wird im Strip Chart Mode ignoriert. Y-Achsen Autoskalierung beginnt mit den from/to Werten im Strip Chart Mode.

Intercept - Die Plazierung der spezifizierten Achse an der senkrechten Achse, angegeben als prozentualer Anteil des gesamten Plotbereiches. Diese Zahl kann negativ oder größer als 100% sein, um die Achsen außerhalb des Plotbereichs zu plazieren, dies ist wichtig im Fall von mehreren Y-Achsen.

Color - Die Achsenfarbe. Der Text wird immer in Schwarz dargestellt.

Logarithmic? - Kontrolliert, ob eine logarithmische Skalierung für die jeweilige Achse verwendet werden soll.

Title - Ein optionaler Titel, der entlang der Achse dargestellt wird.

Ticks - Settings, die bestimmen, ob und wie die Tick Marks entlang der Achse angeordnet sind. Das Step Setting entspricht dem Intervall zwischen den Tick Marks (falls Auto angeschaltet ist, wird dieses Setting ignoriert). Das # minor Setting gibt die Anzahl der kleineren Tick Marks zwischen den größeren an (falls Auto angeschaltet ist, wird dieses Setting ignoriert). Das Position Setting zeigt an, auf welcher Seite der Achse die Tick Marks plaziert werden. Falls eine logarithmische Achse vorliegt, werden die Step und # Minor Settings ignoriert.

Grids - Settings, die die Gitterlinien im Plotbereich kontrollieren. Das größere Gitternetz wird auf den Plätzen der größeren Tick Marks gezeichnet. Das kleinere Gitternetz wird an den Plätzen der kleineren Tick Marks gezeichnet. Jeder Typ von Gitternetz kann separat an- oder ausgeschaltet und für jeden die zugehörige Farbe gewählt werden.

Action List - keine

Cursor

Programmgesteuert können Cursor zu dem Graphen hinzugefügt werden, die im XY Pairs Mode erstellt werden können, indem die Cursor als zusätzlicher Trace mit einem oder zwei Punkten verwendet werden. Zwei Beispiele sind TestPoint beigefügt: CURSOR1.TST und CURSOR2.TST.

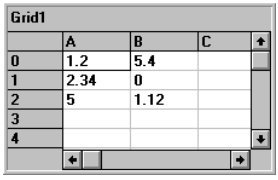
Grid Objekt

Das Grid Objekt kann dazu verwendet werden, Vektoren, Listen oder Arrays im Tabellenformat zu visualisieren und dem Benutzer eine Manipulation zu erlauben.

Grid Objekte können in zwei grundlegenden Arten verwendet werden: Dateneingabe oder Datenansicht. Sobald die Ergebnisse angezeigt werden, kann das Grid Objekt so gesetzt werden, daß die Werte nur angesehen und nicht geändert werden dürfen (siehe Enabled Setting weiter unten).

Grid Objekte können zwei Typen von Daten enthalten: Arrays und Lists of Vectors. Ein Array hat immer eine rechteckige Form, wobei alle Elemente den selben Typ besitzen (Number oder String). Lists of Vectors können allgemeiner verwendet werden, wobei jede Spalte einen separaten Vektor darstellt. Bitte beachten Sie, daß Graph Objekte ebenfalls mit Lists of Vectors verwendet werden können.

Benutzerschnittstelle



Das Grid Objekt schaut aus wie eine Tabelle, die mit bezifferten Reihen und belabelten Spalten ausgestattet sind. Die Spaltenbreite und Label werden von den Settings bestimmt (siehe weiter unten). Neue Werte können vom Bediener in die Zellen der Tabelle eingetragen werden. Die Anzahl der Reihen und Spalten, die vorhanden sind, sind einstellbar.

Um sich innerhalb der einzelnen Zellen des Grids zu bewegen, können die Cursortasten benutzt werden. Die Pos1 Taste bewegt den Cursor

auf die erste Spalte in der momentanen Reihe, die Ende Taste bewegt ihn auf die zuletzt benutzte Spalte in der Reihe. Die Strg-Pos1 und Strg-Ende Taste bewegen den Cursor auf die jeweilig äußerste linke oder rechte Zelle des Grids.

Um einen Wert in einer Grid Zelle zu ersetzen, muß der Benutzer nur den neuen Wert noch einmal eintippen. Die Enter Taste kann dafür benutzt werden, eine Zelle zu editieren (Enter öffnet ein Editierfenster mit dem aktuellen Zellenwert darin). Nachdem ein neuer Wert eingegeben wurde, kann mit der Enter Taste diese Änderung bestätigt werden. Die Auf- oder Abwärts-Tasten können ebenfalls eine Änderung bestätigen und den Cursor auf eine andere Zelle in einem Schritt bewegen.

Scroll Balken erscheinen automatisch, falls die Reihen und Spalten nicht in die momentane Größe des Grids Fensters passen.

Aktionen

Set - Diese Aktion setzt den gesamten Datenwert des Grids. Die Set Aktion wird normalerweise mit einem Vektor, einem Array, oder einer List of Vectors als Parameter verwendet.

Set cell of - Diese Aktion setzt den Wert einer einzelnen Zelle in dem Grid mit gegebenem Reihen- und Spaltenindex (startend bei 0, unabhängig von dem Reihen- und Spaltenlabelling). Diese Aktion funktioniert ebenso, als ob der Benutzer einen Wert in eine Zelle manuell eintippt.

Clear - Diese Aktion setzt den Datenwert des Grid auf NODATA, dies erzeugt ein leeres Grid.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des Grid Objekts entspricht den Daten, die am Bildschirm dargestellt werden. Die Daten können entweder ein Array oder eine List of Vectors sein (oder NODATA , falls das Grid leer ist). Diese Unterscheidung wird in dem Settings getroffen (siehe weiter unten).

Falls das Grid auf den Array Datentyp gesetzt wird, müssen alle Werte vom gleichen Typ sein, entweder alle numerisch oder alle Strings. Die Gridwerte formen in diesem Fall immer ein Rechteck. Beispielsweise, falls der Benutzer in die Zelle B3 eines leeren Grids schreibt, werden Nullen in den Zellen A1,A2,A3,B1 und B2 erscheinen, da das Array nun aus 3 Reihen und 2 Spalten besteht.

Falls das Grid eine List of Vectors ist, ist jede Spalte ein separater Vector von Daten. Alle Werte in der Spalte müssen vom selben Datentyp sein, die Spalten untereinander dürfen aber voneinander abweichen. Beipielsweise kann die Spalte A aus Strings und die Spalte B aus Zahlen bestehen. Ebenso dürfen die Spalten unterschiedliche Länge haben.

Initial Value

Das Grid wird ohne Werteinhalt initialisiert (NODATA), sobald die Applikation startet.

Einstellungen (Settings)

Max. rows - Die maximale Anzahl an Reihen, die im Grid erscheint.

Max. columns - Die maximale Anzahl an Spalten, die im Grid erscheint.

Numeric - Falls angekreuzt, können die Griddaten nur noch numerische Werte annehmen. Jede Eingabe eines Strings wird in eine Zahl umgewandelt (gewöhnlich 0).

Daten type - Array oder List of vector. Siehe obige Beschreibung des Datenwertes.

Col. Widths - Falls in diesem Setting kein Eintrag enthalten ist, wird die Standard Spaltenbreite verwendet. Dieses Setting kann auch auf mehrere Spalten angewendet werden, indem die Anzahl der Digits durch Kommata getrennt verwendet werden. Die Breiten werden dann in der gegebenen Reihenfolge verwendet, wobei der letzte Wert für die noch übrigen letzten Spalten verwendet wird. Beipielsweise könnte dieses Setting aussehen: "4,4,10,8,6".

Col. Labels - Falls leer, werden die Standard Spaltenlabels A,B,C... benutzt. Dieses Setting darf auch aus mehreren Labels durch Kommata getrennt bestehen. Falls dieses die Serie 0,1,..etc., 1,2,.. etc., oder A,B...etc. ist, wird es automatisch für alle Spalten erweitert. Jegliche andere Serie von Labels wird nur für die korrespondierende Anzahl an Spalten verwendet, jede zusätzliche Spalte besitzt dann kein Label.

Visible - Falls nicht angekreuzt, erscheint das Grid nicht auf dem Panel. Es kann jedoch noch dafür benutzt werden, Daten zu speichern, und programmgesteuert auf Array Elemente zuzugreifen.

Enabled - Falls angekreuzt, darf der Benutzer neue Werte in die Zellen eingeben, mit Hilfe der Maus oder der Tastatur. Falls nicht angekreuzt, zeigt das Grid nur Daten an, aber erlaubt dem Bediener keine Modifikation.

Grayed - Falls angekreuzt, wird die Caption (Beschriftung) des Grid grau, um anzuzeigen, daß das Eingabefeld inaktiv ist.

Action List

Es gibt keine Action List für das Grid Objekt.

Beispiele

Ein Grid kann für die Dateneingabe benutzt werden, wobei ein Pushbutton (Druckschalter) in seiner Action List die eingegebenen Daten anzeigt:

```
1) Draw graph     Graph1          with Grid1
```

Eine weitere übliche Verwendungsart für das Grid Objekt ist das Anschauen von Testergebnissen. Es können sowohl ein Grid als auch ein Graph verwendet werden, um Daten darzustellen:

```
1) Draw graph     Graph1          with Results
2) Set            Grid1           to Results
```

Ein Grid zeigt hier Zahlen in voller Genauigkeit (bis zu 15 Digits) an. Um die numerischen Daten leichter lesbar zu gestalten, kann ein Abrunden mit Hilfe eines Math Objekts durchgeführt werden, wie beispielsweise:

```
1) Calculate      Rounded         with x=A/D1
2) Set            Grid1           to Rounded
```

Die Formel für "Rounded" ist hiermit: **round(x*1000)/1000**

Gruppieren von Objekten 

TestPoint Gruppen können aus jeglicher Kombination von Objekten gebildet werden, hierbei werden die Objekte zusammengeführt und es können Aktionen von der gesamten Gruppe ausgeführt werden.

Um eine Gruppe zu erzeugen, müssen die gewünschten Objekte im Objektfenster oder im Panel ausgewählt werden, und daraufhin das **Edit/Group** Kommando verwendet werden. Ein neues Group Icon wird daraufhin im Objektfenster erscheinen, das einen Pfeil nach rechts aufweist, der es dem Programmierer erlaubt, die Objekte innerhalb der Gruppe zu bearbeiten. Die Objekte selber bleiben aber auf dem Panel.

Bitte beachten Sie: Die Gruppierung von Objekten kann wieder aufgelöst werden, indem die Gruppe ausgewählt wird und das **Edit/Ungroup** Kommando verwendet wird.

Eine Gruppe erleichtert dem Programmierer die Organisation und Übersichtlichkeit der Objektliste.

Eine Gruppe von Objekten bildet auch auf dem Panel eine Einheit: Falls ein Objekt bewegt wird, folgen auch die anderen. **Falls aber während des Bewegens die Shift Taste gedrückt wird, wird NUR das Objekt bewegt, das Kontakt mit der Maus hat.**

Eine Gruppe besitzt ebenfalls Aktionen, die sich auf die ganze Gruppe beziehen, auf diese Weise kann eine Gruppe wie ein indizierbares Array behandelt werden. Somit stellt die Gruppierung ein mächtiges Programmierfeature dar.

Beispielsweise können drei Display Objekte gruppiert werden und die resultierende Gruppe in eine Action List gezogen werden. Daraufhin erscheint folgendes Menu von Aktionen:

Die erste Aktion, "Set", ist die reguläre Set Aktion, die normalerweise für die Display Objekte vorhanden ist (der Typ des ersten Objekts in der Gruppe ist für die Auflistung der Aktionen verantwortlich). Wenn Sie diese Aktion auswählen, erhalten Sie die folgende Action List Zeile:

welche die normale Set Action für ein Display Objekt enthält, mit einem zusätzlichen Parameter "index=".

Indizieren

Mit dem Index Parameter können Sie jedes Display einzeln setzen oder alle gleichzeitig mit einer einzigen Action List Zeile:

1) Set Group1 to "abc" [index=0]

setzt das erste Display Objekt in der Gruppe auf „abc“. Beachten Sie, daß der Index eine Variable oder auch eine Konstante sein kann.

2) Set Group1 to "xxx" [index="all"]

setzt alle drei Display auf „xxx“

3) Set Group1 to A/D1 [index="each"]

setzt jedes einzelne Display auf einen Wert, entsprechend den Daten der Liste des A/D1 Objektes.

4) Set Group1 to A/D1 [index="0,2"]

setzt das erste und dritte Display, basierend auf der Liste der Daten des A/D1 Objektes. Sie können hier direkt die Kanal-Liste angeben und die Daten gehen dann entsprechend in das korrespondierende Display.

(Siehe auch Beispiel ADGROUP.TST).

Der Index Parameter kann sein:

- Eine einzelne Nummer, entsprechend dem Index der Objekte.
- Das Wort "**all**", welche die Aktion für alle Objekte mit dem gleichen Parameter ausführt.
- Das Wort "**each**", welche die Aktion für jedes Objekt ausführt, die Parameter entsprechen der Liste der Daten, die übergeben werden. Wenn die Liste der Daten kleiner als die Anzahl der Argumente ist, erhalten die übrigen Objekte den Wert „nodata“.
- Eine Liste/Vektor des Typs (Zahl) Nummer, ein String, in dem die (Zahl) Nummer durch Komma getrennt sind (wie z. B. eine A/D Kanal Liste). Ähnlich der „each“ Funktion, die Aktionen werden aber nur den Indizes entsprechend ausgeführt.

Aktionen

Eine Gruppe hat alle Aktionen des ersten Objektes der Gruppe zur Verfügung. Wenn Sie also eine Gruppe von GPIB Objekten erstellen, haben Sie als Aktion „Output to“, „Enter from“ und alle andere Aktionen des Objektes zur Verfügung, mit dem zusätzlichen „index=“ parameter.

Außerdem sind folgende Aktionen vorhanden:

Get Daten - Liest die Daten eines Objektes (oder aller) der Gruppe aus. Der Index Parameter funktioniert für die Aktion identisch.

Get # objects - Das Group Objekt erhält als Wert die Anzahl der Objekte der Gruppe.

Achtung: es ist problemlos möglich, eine Gruppe aus verschiedenen Typen von Objekten zu erstellen, aber die Aktionen, die dann zur Verfügung stehen, entsprechen denen des ersten Objektes in der Gruppe. Wenn Sie eine Aktion mit einem Objekt der Gruppe aufrufen, die bei diesem Objekt nicht zur Verfügung steht, erhalten Sie eine Fehlermeldung.

Indicator Objekt 

Das Indicator Objekt ist ein reines Ausgabe Objekt, es ermöglicht die Darstellung eines Ja/Nein bzw. pass/fail Status.

Benutzerschnittstelle

Es gibt zwei Darstellungsarten: LED und Backlit Text.
Typ LED ist ein runder beleuchteter Bereich mit einer Umrahmung, zusätzlich ist ein Text vorhanden, der den Status anzeigt. Wenn die LED den „keine Daten“ Status hat, ist die LED schwarz und kein Text wird angezeigt.
Bei Typ Backlit Text wird das ganze Display mit einer Farbe ausgefüllt und ein Text in dem Feld angezeigt. Wenn keine Daten im Objekt vorhanden sind, ist das ganze Rechteck hellgrau.

Beide Arten bieten optional eine Beschriftung über der Anzeige.

Aktionen

Set - Setzt das Indicator Objekt auf den angegebenen Wert. Eine Null entspricht dem Wert „false“ (unwahr). Jeder andere Wert (nicht Null) setzt das Objekt auf „true“ (wahr) bzw. den Status 1.

Clear - Rücksetzen den Indicators auf den „keine Daten“ Status. Diese Aktion dient dazu, um ein Indicator Objekt wieder in einen „neutralen“ Zustand zu bringen, so daß das letzte Ergebnis nicht mehr angezeigt wird.

Achtung: Um eine Eigenschaft (Setting) des Objektes aus einer Action List heraus zu verändern, ziehen Sie die gewünschte Eigenschaft aus dem Settings Fenster in die jeweilige Action List.

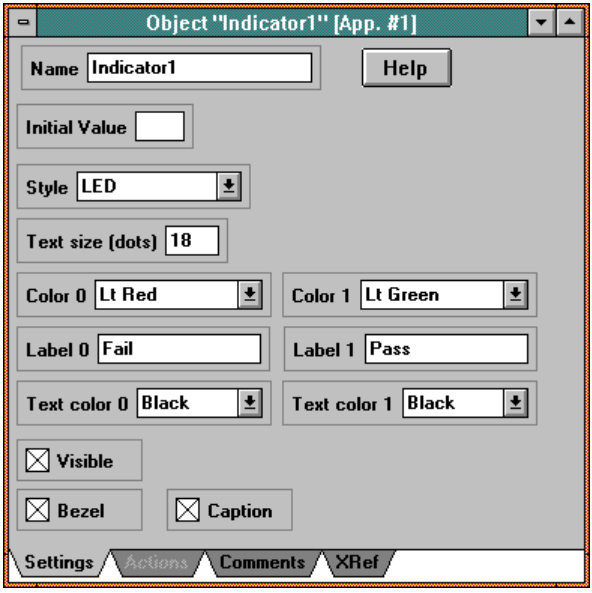
Daten

Der Daten Wert des Indicator Objekt ist der, auf den das Objekt zuletzt mit der „Set“ Action gesetzt wurde.

Initial value

Der Anfangs-Wert hängt von den Eigenschaften „Settings“ ab.

Einstellungen (Settings)



Style - LED oder Backlit Text. Siehe Beschreibung oben.

Caption - Wenn aktiviert, wird der Name des Objektes über dem Indicator angezeigt.

Text size - Die Größe des Textes in Punkten. Der Schriftsatz ist immer der Windows Arial TTF.

Color n - Die Farbe für den Status n (0 oder 1). Für eine LED Anzeige ist es die Farbe der LED. Für Backlit Text ist es die Hintergundfarbe der Anzeige.

Label n - Der Text der für den Status n (0 oder 1).

Text color n - Die Farbe des Textes, der angezeigt wird für Status n (0 oder 1).

Visible - Einstellung, ob das Objekt sichtbar sein soll.

Initial Value - Der Indicator kann bei der Initialisierung (Start der Anwendung) auf einen beliebigen Wert gesetzt werden. Wird kein Wert eingetragen, bekommt das Objekt den Wert „keine Daten“ beim Start.

Bezel - Bei Aktivierung dieser Option wird ein Rahmen um das Objekt gezeichnet.

Action List - keine

Indirect objects (Indirekte Objekte)

Ein indirektes Objekt dient als Platzhalter, der eingesetzt werden kann, um Action Lists zu erzeugen, die mit verschiedenen Objekten als dem dann aktuellen ausführbar sind.

Es ist ein leistungsfähiges Programmierwerkzeug für den fortgeschrittenen Programmierer. Es kann hilfreich sein bei der Erstellung von Benutzer-definierten Objekten (user-defined Object) das eine Aktion mit einem Objekt ausführen soll, daß erst später vom Anwender bereitgestellt wird. Da das Objekt bei der Entwicklung Ihres UDO (user-defined objects) nicht zur Verfügung steht, können Sie es auch nicht in eine Action List Ihres Objektes übernehmen.
Statt dessen setzen Sie einfach das indirekte Objekt ein.

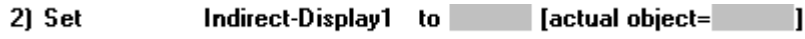
Indirekte Objekte entsprechen Zeigern in konventionellen Programmiersprachen wie Pascal oder C.

Sie erzeugen ein indirektes Objekt, indem Sie ein existierendes Objekt markieren und aus der Menüleiste **Utilities/Make indirect Objekt** auswählen. Damit erstellen Sie ein Objekt, das dem Original entspricht, nur daß es zusätzlich einen kleinen Pfeil in der Ecke hat:

aus  wird 

Indirekte Objekte enthalten exakt die gleichen Aktionen wie das reguläre Objekt, aus dem Sie es erzeugt haben, aber mit dem zusätzlichen Parameter: **actual Objekt=**.

Als Beispiel, ein indirektes Display Objekt hat diese Aktion:



Der "actual object" Parameter ist erforderlich und sollte ein Objekt gleichen Typs seien.

Eine sehr nützliche Einsatzmöglichkeit für ein indirektes Objekt ist die Verwendung in der Action List eines Action Objektes. Wobei der Parameter „actual object“ ein Argument enthält, das in das Action Objekt übergeben wird:

```
2) Set          Indirect-Display1   to "xyz"   [actual object="%1"   ]
```

Diese Zeile, eingesetzt im Action Objekt, übernimmt den Parameter %1, der dem Action Objekt übergeben wird in das indirekte Display Objekt und setzt es auf „xyz“.

Durch indirekte Objekte können Sie vielseitig einsetzbare Unterroutinen gestalten, die eine Serie von Aktionen mit verschiedenen Objekten ausführen können.
Das ermöglicht einen flexiblen Einsatz an unterschiedlichen Stellen Ihrer Applikation.

Siehe auch Beispiel INDIRECT.TST

Loop (Schleifen) Objekt 

Das Loop (Schleifen) Objekt wird benutzt, um bestimmte Aktionen wiederholend auszuführen. Es stellt auch eine Serie von Zahlen bereit, für Berechnungen in der Schleife.

Wenn das Loop Objekt in einer Action List eingefügt wird, erscheinen zwei Zeilen für Start und Ende der Schleife. Diese Begrenzungen der Schleife können in der Action List mit der Maus (Drag & Drop an der Zeilennummer) auch verschoben werden, um die auszuführenden Aktionen einzuschließen.

Benutzerschnittstelle - keine

Aktionen

Linear series - Generiert eine lineare Folge von Zahlen, basierend auf den Parametern „from“ (von), „to“ (bis) und „step“ (Schrittweite). Der „step“ Parameter hat den Wert 1, wenn er nicht angegeben wird (For-next-Schleife).
Beispiel: from=1, to=5, step=1 erzeugt die Folge: 1,2,3,4,5

Geometric series - Generiert eine geometrische Folge von Zahlen, basierend auf den Parametern „from“ (von), „to“ (bis) und „multiply by“ (multipliziert mit).
Beispiel: from=1, to=5, multiply by=2 erzeugt die Folge: 1,2,4,8,16

Decade series - Generiert eine 1,2,5 Serie von Zahlen über eine oder mehrere Dekaden, basierend auf 10^N bis 10^{N+n}. Die Parameter sind „from“ (von) und „to“ (bis). Als Beispiel: from=10^1 , to=10^3 erzeugt die Folge: 10,20,50,100,200,500,1000,2000,5000

Do loop - Führt eine Schleife aus, solange gegebene Parameter nicht Null sind (Wahr). Beachten Sie, daß der Parameter ein Objekt sein

muß, keine Konstante. Drücken Sie Ctrl-Break, um die Schleife abzubrechen.

Repeat loop - Führt die Schleife aus, solange der gegebene Parameter Null (Falsch) ist. Die Schleife wird mindestens einmal ausgeführt.

Achtung: Um eine Eigenschaft (Setting) des Objektes aus einer Action List heraus zu verändern, ziehen Sie die gewünschte Eigenschaft aus dem Settings Fenster in die jeweilige Action List.

Daten

Der Datenwert des Objektes ist der momentane numerische Wert der Schleife, in Abhängigkeit der Aktion am Anfang der Schleife. Der Datenwert wird vor der Ausführung jeder Schleife erneuert.

Initial value - keine

Settings - keine

Action List - keine

Beispiel

```
1) Linear series   Loop1      from 1 to 10, step by 1
2)    Set          Display1   to Loop1
3) End             Loop1
```

Beispiel - Sich wiederholende Aktionen, während ein Schalter auf EIN geschaltet ist

Dafür muß ein Switch-Objekt benutzt werden, das folgende Action List besitzt:

```
1) Do loop        Loop1       while Switch1 is true
2)    Enter from  GPIB1       up to 256 bytes
3)    Set         Display1    to GPIB1
4) End            Loop1
```

Sobald der Schalter das erste Mal angeschaltet wird, wird seine Action List ausgeführt (da der Schalterwert geändert wurde). Die Do Loop Bedingung ist wahr, deshalb beginnt die Schleife und läuft solange der Schalter angeschaltet ist. Wenn der Schalter ausgeschaltet wird, stoppt die Schleife. Die Action List wird zwar ebenso ausgeführt, weil wieder der Schalterwert geändert wurde, aber diesmal ist der Schalter ausgeschaltet, demnach wird die Schleife übersprungen.

Da TestPoint ein Multitasking von Action Lists erlaubt, können auch andere Action Lists ausgeführt werden und andere Knöpfe und Schalter während der Ausführung der Schleife betätigt werden.

Beispiel - Hintergrundverarbeitung

In einigen Applikationen kann es nützlich sein, Aktionen kontinuierlich während der aktiven Programmausführung auszuführen.

Alles was dazu benötigt wird, ist eine Schleife wie im vorigen Beispiel, mit einem Schalter, der den Initialwert **1** (angeschaltet) besitzt, und bei der Initialisierung ausgeführt wird. Der Schalter kann auch unsichtbar geschaltet werden, indem das "Visible" Setting ausgeschaltet wird. Sobald das Programm startet, wird der Schalter angeschaltet und damit seine Action List ausgeführt. Da der Schalter nicht sichtbar ist, bleibt er aktiv, und die Schleife in der Action List wird bis zum Ende des Programms ausgeführt.

Math Objekt 

Mit dem Math-Objekt können Sie Formeln mit beliebiger Anzahl von Variablen berechnen. Die Funktionen können auf beliebige Datentypen angewendet werden: Zahlen, Zeichenketten, Vektoren, Felder und Listen. Es ist hierbei nicht erforderlich, Schleifen zu programmieren. Nach der Benutzung eines Math-Objektes steht das Ergebnis der Berechnung als neuer Datenwert des Objektes zur Verfügung. Das Math-Objekt mit der Formel **X*2+3** arbeitet genauso, wie der Befehl **Math1=X*2+3** in einer konventionellen Programmiersprache.

Benutzerschnittstelle - keine

Aktionen

Calculate - die Parameter variieren in Abhängigkeit der Einstellungen in der Formel. Es steht für jede Variable in der Formel ein Parameter zur Verfügung. Die Aktion berechnet das Ergebnis des mathematischen Ausdruckes und setzt den Datenwert des Math-Objektes auf das Ergebnis.

Set - setzt das Math-Objekt auf einen spezifischen Wert. Die Aktion benutzt nicht die Formel der Einstellungen. Die „Set“-Aktion ist vor allem dann sinnvoll, wenn die Formel „previous()“ benutzt wird, um Bezug auf den vorhergehenden Wert des Math-Objektes zu nehmen. Dies ist z. B. bei einem Zähler der Fall. Siehe Beispiel.

Daten

Der Datenwert des Objektes ist abhängig vom berechneten Ergebnis.

Anmerkung: Um die Einstellungen dieses Objektes in einer Action List zu verändern, ziehen Sie mit der Maus diesen Einstellungsparameter in die gewünschte Action List.

Initial value - keine

Einstellungen (Settings)

Formula - Ein mathematischer Ausdruck, bestehend aus Variablen (beliebige alphanumerische Bezeichnungen), Konstanten und mathematischen Operatoren (wie +,-,*,/). In den Ausdrücken können Zahlen (Skalare) und Vektoren benutzt werden. Beispielsweise kann der Ausdruck „3*b+4“ verwendet werden, wobei „b“ eine Zahl oder ein Vektor sein ist. Ebenso können ganze Vektoren verglichen werden, wie z. B. in „a>b“.

Eine komplette Referenz aller verfügbaren mathematischen Funktionen finden Sie im „Math Reference“ Kapitel dieses Handbuches auf Seite Math-1.

Action List - keine

Beispiel

Mit der Formeleinstellung:

```
1 / (R*C)
```

berechnet die nachfolgende Action List die Zeitkonstante für ein Widerstand-Kondensator Netzwerk:

```
1) Calculate      Time Constant     with R=Widerstand,
                                    C=Kondensator'
```

wobei Widerstand und Kondensator TestPoint Objekte sind, wie z. B. Dateneingabefelder.

Beispiel - Interpolation mit Benutzung einer Vergleichstabelle aus einer Datei

Erstellen Sie ein File-Objekt mit dem Namen „Vergleichstabelle“, um auf Dateien mit der Endung „.TBL“ zuzugreifen. Vergeben Sie dabei den Startwert (Initial Value) „PLATRTD.TBL“. Erzeugen Sie „.TBL“-Datendateien, die je zwei Spalten mit Zahlen beinhalten. Die erste Spalte enthält direkt gemessene Werte, wie Widerstand; die zweite enthält das bei dem entsprechenden Widerstandswert erwartete Ergebnis, wie Temperatur.

Die Action List des File-Objektes sollte lauten:

1)	Input from	Lookup Table	up to 32768 bytes

Immer wenn eine Datei ausgewählt wird, liest das File-Objekt die Tabelle ein, und stellt sie zur weiteren Benutzung zur Verfügung, beispielsweise im Math-Objekt.

Erstellen Sie ein Math-Objekt mit dem Namen „Temperature“ mit der Formel:

```
interpolate (input,table)
```

Diese Action List für einen „Messen“-Pushbutton führt eine Widerstandsmessung mit dem „DMM“-GPIB-Objekt durch, berechnet die dazugehörige Temperatur und zeigt sie in einem „Value“-Display-Objekt an:

1)	Output to	DMM	":MEAS:RES?"
2)	Enter from	DMM	up to 256 bytes
3)	Calculate	Temperature	with input=DMM table=Lookup Table
4)	Set	Display1	to Temperature

Beispiel - Zähler

Verschiedene Anwendungen erfordern einen Zähler, der erhöht wird, wenn ein Schalter geklickt wird oder wenn ein Testdurchlauf angeschlossen ist usw.

Ein Zähler ist ein Beispiel für eine mathematische Operation, deren Ergebnis abhängig ist von dessen vorhergehenden Wert. Das Math-Objekt besitzt die „previous()“-Funktion für diese Fälle. Die „Set“-Aktion wird benutzt, um den Zähler zu initialisieren.

Die Formel lautet:

```
previous()+1
```

Diese Action List kann benutzt werden, um den Zähler auf 0 zu setzen:

```
1) Set          Counter          to 0
```

Und diese Action List erhöht den Wert des „Counter“-Objektes bei jedem Aufruf:

```
1) Calculate    Counter
```

Mathematik: Anwendung von Formeln auf verschiedene Datentypen

TestPoint unterstützt viele Arten von Daten: Zahlen, Zeichenketten, Vektoren, Felder und Listen. Die meisten mathematischen Funktionen sind so definiert, daß sie die verschiedenen Datentypen akzeptieren und mit ihnen entsprechend arbeiten.

Beispielsweise arbeiten die arithmetischen Operatoren mit Zahlen. Sie arbeiten ebenfalls mit Vektoren von Zahlen, Feldern aus Zahlen oder Listen, die Felder, Vektoren oder Zahlen enthalten. Es ist dadurch möglich, Vektoren oder eine Zahl zu jedem Element eines Vektors mit einer einzigen mathematischen Operation zu addieren. Die Notwendigkeit, Schleifen zu benutzen, die auf jedes Element zugreifen, ist nicht mehr gegeben.

Hier sehen Sie einige Beispiele, wie sich die Additionsoperation auf die verschiedenen Datentype auswirkt:

Diese Formel:	führt zu:
3+5	8
3+vector(1,2,3)	vector(4,5,6)
vector(1,2,3)+vector(5,0,2)	vector(6,2,5)
1+list(4,2,3)	list(5,3,4)
list(vector(1,2),5)+2	list(vector(3,4),7)
list(1,2,3)+list(4,3,2)	list(5,5,5)

Die Möglichkeit, bei mathematischen Funktionen von TestPoint verschiedene Datentypen zu verwenden, ist in der Technik als **Polymorphie** bekannt. Sie vereinfacht viele Anwendungen dadurch, daß Schleifen und Vektorindizierungen wegfallen.

Beispiel: Je 100 Analogwerte von 3 Kanälen, erfaßt von einer A/D-Karte, werden in TestPoint als Liste mit 3 Vektoren, die jeweils 100 Werte beinhalten, gespeichert. Wenn diese Werte mit einem Skalierungsfaktor multipliziert, und dazu ein Offset addiert werden soll, um die externe Signalverarbeitung zu korrigieren, dann erfordert dies in den meisten konventionellen Programmiersprachen verschachtelte Schleifen. In TestPoint kann ein Math-Objekt mit der Formel

```
input*scale+offset
```

benutzt werden, um diese Operationen in einem Schritt durchzuführen:

1)	Acquire A/D	A/D1	#samples=100, rate=1000 Hz, channel(s)="0,1,2"
2)	Calculate	Corrected	with input=A/D1, scale=2, offset=3

OLE Objekt 

OLE (Object Linking and Embedding) ist ein Feature in Windows, das es ermöglicht, Informationen, die von einem Programm bearbeitet und dargestellt werden, direkt in einem anderen Programm anzuzeigen.

Das TestPoint OLE Objekt kann Daten von anderen Programmen auf einem TestPoint Panel anzeigen.

Es kann auf zwei Arten benutzt werden: mit sogenannten **embedded** (eingebetteten) Daten oder mit **linked** (verbundenen) Daten.

Eingebettete Daten werden innerhalb der TestPoint Applikationsdatei (.TST File) gespeichert. Beispielsweise können Sie ein OLE Objekt in die TestPoint Applikation ziehen, dann "Paintbrush picture" für den Objekttyp wählen, und dann den "New" Knopf drücken. Das Paintbrush Bild, das Sie daraufhin erzeugen können, wird in der .TST Datei gespeichert - kein separates Bitmap File ist erforderlich. Dies ist ein Vorteil von eingebetteten OLE Daten: Sie brauchen nicht separate Dateien mit der eigenen Applikation kombinieren, oder mit Runtime Versionen verteilen. Der Nachteil jedoch liegt darin, daß es durch das Einbetten nicht möglich ist, automatische Erneuerungen von verbundenen TestPoint Daten zu erzeugen.

Verbundene Daten stammen von einer separaten Datei. Beispielsweise kann eine Excel Tabellenkalkulationsdatei erzeugt werden und dann mit TestPoint verbunden werden. Um ein verbundenes OLE Objekt zu erstellen, muß das OLE Objekt aus dem Stock gezogen, dann in das andere Programm (z. B. Excel) geschaltet, die gewünschte Datei ausgewählt, der Edit/Copy Befehl im anderen Programm benutzt, zurück zu TestPoint geschaltet und dann der "Paste" Knopf des OLE Objekts gedrückt werden.

In jedem Fall erscheinen die OLE Daten auf dem TestPoint Panel als ein Bild, das von einer anderen Applikation stammt.

OLE-fähige Programme sind unter anderem Excel, Lotus 123, Microsoft Word, Paintbrush, und viele andere.

Bidirektionale Verbindungen

Eine der nützlichsten Applikationen von OLE liegt darin, bidirektionale Verbindungen zu erzeugen. Das bedeutet, daß ein Bild, das von einem anderen Programm erzeugt wurde, beispielsweise Excel, Daten von TestPoint bekommt. Jedesmal, wenn sich die TestPoint Daten verändern, paßt sich das OLE-Bild automatisch an!

Dies erfordert, daß einerseits die "andere" OLE Applikation gleichzeitig läuft, und den Update der Daten automatisch erzeugt. Andererseits muß die andere Applikation Daten in sich speichern, die durch eine "Paste link" Operation von TestPoint geliefert werden.

Im folgenden ein schrittweiser Ablaufplan für das Erstellen einer solchen OLE Verbindung (mit Excel als Beispiel):

1. Öffnen Sie TestPoint und Excel.

2. Stellen Sie sicher, daß sowohl die .TST, als auch die Excel Datei gespeichert sind (damit keines ohne Namen ist - da die Verbindung immer einen Dateinamen erfordert).

3. Erzeugen Sie das gewünschte Objekt in TestPoint, das die Quelldaten enthalten wird. Beispielsweise kann ein Container Objekt mit dem Namen "OLE source" verwendet werden. Jedesmal, wenn die OLE Daten in der Action List erneuert werden sollen, kann die gewünschte Information in diesen Container geschrieben werden.

4. Wählen Sie das Quelldatenobjekt ("OLE Source") in dem Objektfenster von TestPoint aus, indem Sie darauf klicken.

5. Benutzen Sie das Edit/Copy Kommando, um es zu kopieren.

6. Wechseln Sie zu Excel, und wählen Sie eine Zelle oder einen Bereich von Zellen aus. Dann benutzen Sie Inhalte Einfügen / Verknüpfung. Dies erzeugt eine Verbindung von TestPoint **nach** Excel.

7. Dann erzeugen Sie die gewünschte OLE Information in Excel. Beispielsweise kann die Chart Funktion verwendet werden, um eine Grafik der verbundenen Daten zu erstellen.

8. Wählen Sie die OLE Information in Excel aus (durch Klicken auf die Grafik). Benutzen Sie das Edit/Copy Kommando, um sie zu kopieren.

9. Wechseln Sie zurück zu TestPoint. In den Settings des OLE Objekts muß der "Paste" Knopf benutzt werden.

10. Nun sollte sichergestellt sein, daß beide Dateien abgespeichert werden.

11. Hiermit ist das OLE Programm lauffähig. Jedesmal, wenn die Daten in TestPoint verändert werden, erneuert sich das OLE Bild.

Bitte beachten Sie: Einige Programme unterstützen **nicht** diese bidirektionalen Verbindungen, da sie nicht erfolgreich die Verbindungen zurückverbinden können, die von den OLE Funktionen gestartet wurden. Im folgenden einige spezifische Informationen für Standardprogramme:

Microsoft Excel: Funktioniert ohne Probleme!

Microsoft Word 2.0: Funktioniert OK, aber es wird nur ein Word Icon angezeigt, nicht der Text.

Microsoft Draw (ist Word und anderen Programmen beigefügt): Funktioniert OK. Bidirektionale Verbindungen können nicht angewendet werden.

Lotus 1-2-3: Unterstützt OLE für Grafiken nicht, nur für ganze Tabellenkalkulationen. Arbeitet sonst OK.

Quattro Pro: Kann nicht bidirektionale Verbindungen automatisch zurückverbinden - erfordert ein manuelles Tools/Links/Refresh Kommando, nachdem Quattro geöffnet wurde.

Microsoft PowerPoint: Funktioniert OK. Das "Slide Show" Kommando arbeitet nur dann korrekt, falls ein verbundenes OLE Dokument benutzt wird, nicht im eingebetteten Modus.

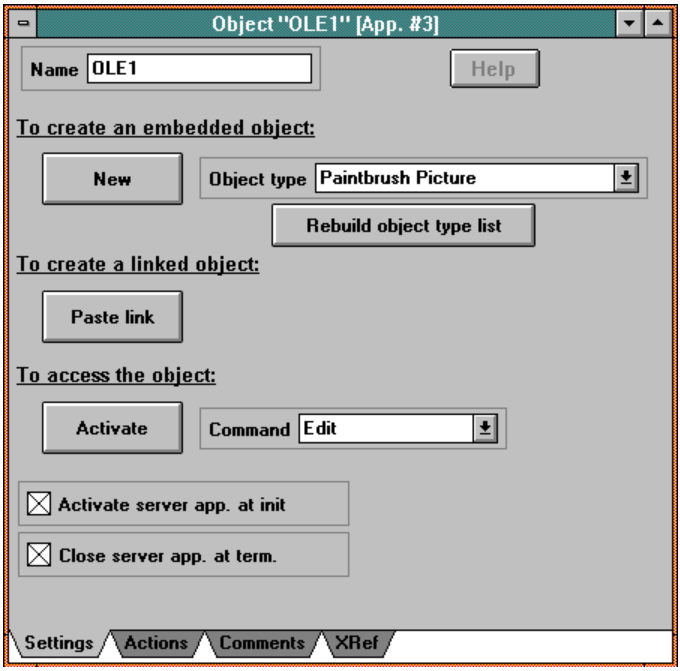
Lotus Freelance: Funktioniert OK.

Sound Recorder (in Windows): Funktioniert OK, spielt Sounds. Bidirektionale Verbindungen können nicht verwendet werden.

Media Player (in Windows, für Video, Audio...): Funktioniert OK. Bidirektionale Verbindungen können nicht verwendet werden.

Die Read-me Dateien sollten beachtet werden, um aktuelle Informationen über OLE zu erhalten.

Einstellungen (Settings)



New - Erzeugt ein neues, eingebettetes Objekt des Typs, der von dem "Objekt type" Setting angezeigt wird.

Objekt type - Auswahl des Objekttyps, das erzeugt werden soll, sobald "New" gedrückt wird.

Rebuild Objekt type list - Erstellt die Systemliste der an Ihrem Rechner möglichen OLE Objekttypen, die von den installierten Programmen abhängt. Dies benötigt einige Sekunden und fügt die gefundenen Programme in die Auswahlliste unter dem "Objekt type".

Paste link - Erzeugen eines verbundenen OLE Objekts aus den Daten in der Zwischenablage. Zuerst muß dafür in der anderen OLE-Applikation das Edit/Copy Kommando benutzt werde.

Activate / Command - Gibt eine Meldung an das "andere" (Server) Programm, dieses OLE Objekt zu aktivieren. Die möglichen Kommandos hängen von den Objekttypen ab, aber enthalten oft den "Edit" Befehl.

Activate server app. at init. - Falls angekreuzt, verwendet TestPoint automatisch Activate mit dem "Edit" Kommando, sobald in den Run Modus gewechselt wird (oder die Runtime startet). TestPoint versucht ebenfalls, den OLE Server zu verstecken und zu minimieren, so daß er nicht auf der TestPoint Oberfläche sichtbar ist. Dieses Setting ist nützlich im Fall von bidirektionalen Verbindungen, aber nicht für statische Objekte wie beispielsweise Paintbrush Bilder.

Close server app. at term. - Falls angekreuzt, (und Activate Server App At Init ebenfalls angeschaltet ist), versucht TestPoint, die Server Applikation automatisch zu beenden, sobald das TestPoint Programm beendet wird (falls in den Edit Modus gegangen, oder die Runtime geschlossen wird). Bitte beachten Sie, daß im Falle bidirektionaler Verbindungen die Server Applikation eine Dialog Box zeigt, die fragt, ob die modifizierten Daten gespeichert werden sollen.

Aktionen

Activate - Hat die selben Aktionen wie der Activate Knopf in den Settings. Damit kann das OLE Objekt von einer Action List aus aktiviert werden. Beispielsweise kann bei einem Sound Waveform Objekt das "Play" Kommando aktiviert werden. Oder man kann ein Paintbrush Bild mit dem "Edit" Kommando aktivieren.

Update - Erneuerung des Bildes. Dies wird normal nicht für verbundene Objekte benötigt, wenn die "andere" (Server) Applikation gerade läuft.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Action list

Die Action List des OLE Objekts wird ausgeführt, sobald der Bediener auf das Objekt doppelklickt. Jede beliebige Aktion, wie beispielsweise "Activate", kann nun das Editieren ermöglichen.

Panel Objekt 

Das Panel Objekt kann dazu benutzt werden, zusätzliche Panels der Applikation hinzuzufügen. Diese Panels können angezeigt oder versteckt werden, indem Aktionen des Objekts verwendet werden. Dies ermöglicht den Einsatz von Hilfsfenstern für eine bessere Übersicht.

Alle Panels können andere Objekte enthalten.

Panels können ebenso innerhalb von benutzerdefinierten Objekten benutzt werden, indem sie Settings des Objekts enthalten. Ebenfalls können Panels auch in benutzerdefinierte Objekte verwandelt werden, indem das Utilities/Package as User-Defined Menükommando verwendet wird (Menüzeile).

Benutzerschnittstelle

Ein Panel ist ein unabhängiges Fenster, dessen Größe im TestPoint Editor verändert werden kann.

Aktionen

Show - Macht das Panel sichtbar, und setzt es an die oberste Position aller Fenster.

Hide - Macht das Panel unsichtbar.

Show & Wait - Macht das Panel sichtbar und wartet dann, bis es unsichtbar wird (durch ein Schließen des Benutzers oder indem eine Aktion ausgeführt wird), bevor die Action List weiter abgearbeitet wird. Die Show & Wait Action ist sehr nützlich, wenn man ein sogenanntes ‘Prompting’ von dem Bediener erwartet, das bedeutet, TestPoint erwartet eine Antwort des Bedieners, bevor die Action List weiter ausgeführt wird.

Move Objekt in - Ändert die Position und Größe eines spezifizierten Objekts auf dem Panel, wie z. B. eines Knopfes, eines Displays, etc. Die Parameterwerte werden in Bildschirmpixeln (Punkten) angegeben. Falls ein Wert kleiner als 0 für einen Parameter benutzt wird, bleibt diese Größe unverändert (beispielsweise, x = 100, y = -1 läßt die y-Koordinate des Objekts unverändert).

Set focus to - Bewegt den momentanen "focus", den Platz, an dem die aktuelle Tastatureingabe stattfinden kann, auf ein spezifiziertes Objekt auf dem Panel (zum Beispiel ein Dateneingabefeld).

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten - keine

Initial value - keine

Einstellungen (Settings)

BG color - Wählt die Hintergrundfarbe des Panels. Helles Grau ist die Standardeinstellung.

X, Y - Die relative Position dieses Panels, gemessen von der Position des Hauptpanels. Bitte beachten Sie, daß dieses relative Positionieren nur in einer Runtimeversion ausgeführt wird, nicht im TestPoint Editor.

Bei dem obersten Panel einer Applikation, zeigen die X und Y Settings die Bildschirmposition an, an der das Hauptpanel beim Start der Applikation als Runtime liegen soll. Falls X und Y -1 sind, wird die Windows Standardpositionierung verwendet.

Multitasking mode - Dieses Setting ist nur für das Top-Level Panel der Applikation möglich. Es gibt an, ob mehrere TestPoint Action Lists gleichzeitg ausgeführt werden können. Weitere Details können dem

Multitasking Kapitel in der "Using TestPoint" Handbuchsektion entnommen werden.

Action List - keine

Picture Objekt 

Das Picture Objekt zeigt eine beliebige Bitmapgrafik auf dem Panel an. Die Bilder können dafür verwendet werden, um graphische Bezeichnungen zu erstellen, Schaltpläne, Bauteillistendarstellungen, Testaufbauskizzen und andere Bildinformationen für den Bediener darzustellen. Ein Bild kann auch hinter andere Objekte plaziert werden. Ein leeres Bild kann dazu benutzt werden, eine Hintergrundfarbe oder einen Rahmen darzustellen. Ebenfalls können zusammengehörige Objekte auf dem Panel mit einer gemeinsamen Caption (Beschriftung) versehen werden.

Benutzerschnittstelle

Zeigt ein Bild, das aus einer Bitmapdatei eingelesen wurde, und die zugehörige Caption (Beschriftung).

Aktionen - keine

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten - keine

Einstellungen (Settings)

Picture filename - Der Name der Bitmapdatei. Unterstützte Dateiformate sind BMP, PCX, TIFF.

BG color - Die Farbe, die benutzt wird, falls kein Bild spezifiziert wird, beziehungsweise für die Bereiche außerhalb des Bildes.

Auto-Size - Falls angekreuzt, wird das Bild automatisch auf die Größe des Orginalbitmap gesetzt.

Stretch - Falls angekreuzt, wird das Bitmapfile so skaliert, daß es in das Bild Fenster hineinpaßt. Dies kann aber zu einem verzerrten Bild führen.

Visible - Kontrolliert, ob das Bild auf dem Panel sichtbar ist.

Caption - Falls angekreuzt, wird der Objektname als eine Caption (Beschriftung) geschrieben.

Bezel - Falls angekreuzt, wird die Bezel (Rahmen) um das Bild gezeichnet. Falls nicht angekreuzt, wird das Bild ohne Rahmen gezeichnet.

Action List - keine

Port I/O Objekt 

Das Port I/O Objekt wird eingesetzt, um Kunden-spezifische oder nicht unterstütze Hardware über direkte I/O Operationen anzusprechen.

Benutzerschnittstelle - keine

Aktionen

Output byte to - Schreibt ein Byte an die angegebene Basisadresse. Der Offset Parameter wird zu der I/O Adresse hinzuaddiert.

Output word to - Schreibt ein Word an die angegebene Basisadresse. Der Offset Parameter wird zu der I/O Adresse hinzuaddiert.

Input byte from - Liest ein Byte von der angegebenen Basisadresse.

Input word from - Liest ein Word von der angegebenen Basisadresse.

Test bits of - Liest ein Byte von der angegebenen Basisadresse und führt eine UND Verknüpfung mit dem Mask Parameter durch. Diese Aktion ist gedacht, um ein oder mehrere Bits zu testen, ohne daß ein zusätzliches Math Objekt benötigt wird.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten

Der Datenwert des Port I/O Objektes ist das Ergebnis einer „Input“ oder „Test bit“ Aktion.

Einstellungen (Settings)

I/O address - Die I/O Basisadresse, auf die zugegriffen werden soll. Ein H am Ende spezifiziert einen hexadezimalen Wert (z. B. 300H).

Beachten Sie, daß Windows einige Basisadressen schützt, die zu systemkritischen Ressourcen oder Komponenten gehören. Ein Zugriff auf diese ist mit dem Port I/O Objekt nicht möglich.

Demo mode - Bei der Aktivierung wird auf den Port nicht wirklich zugegriffen, es werden statt dessen Demodaten bei der Input Action geliefert.

Initial value - keine

Action List - keine

Pushbutton Objekt 

Das Pushbutton (Druckschalter) Objekt bietet eine Benutzerschnittstelle, die über Anklicken eine Action List startet.

Benutzerschnittstelle

Ein Standard Windows Pushbutton (Taster), mit dem Objektnamen als Beschriftung.

Aktionen

Push - Anklicken des Schalters durch den Benutzer.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten - keine

Einstellungen (Settings)

Visible - Kontrolliert, ob der Pushbutton auf dem Panel sichtbar sein soll.

Enabled - Wenn nicht aktiviert, ist die Beschriftung hellgrau und der Pushbutton kann nicht betätigt werden.

Exec. Aktionen at initialize - Bei der Aktivierung der Option wird die Action List einmal bei der Initialisierung (Start) des Programms ausgeführt.

Action List

Die Action List wird immer bei der Betätigung des Pushbuttons ausgeführt (oder wenn die Push Aktion des Objektes in einer anderen Action List aufgerufen wird).

Report Objekt 

Das Report Objekt erzeugt gedruckte und formatierte Protokolle, die Text, Daten und Graphik enthalten können. Die Standard Windows Druckertreiber werden benutzt, so daß eine Unterstützung für alle Drucker und Schriftarten vorhanden ist.

Benutzerschnittstelle - keine

Aktionen

Start - Fängt ein neues Protokoll an.

Print - Druckt einen oder mehrere Einträge auf einem Protokoll. Diese Aktion druckt die momentane Position auf der Seite und verändert nach dem Drucken die Position normalerweise auf die nächste Zeile.

Die Print Aktion kann als Parameter jede Form von Daten verarbeiten, egal ob es sich um konstante Texte oder um TestPoint Objekte handelt, deren Daten gedruckt werden sollen. Das Drucken der Daten erfolgt sequentiell, normalerweise in einer Zeile, bis das Ende Zeile erreicht wird oder die Daten ein Wagenrücklauf oder Zeilenvorschub Zeichen enthalten. Am Ende der Print Action wird die momentane Position aktualisiert (auf die nächste Zeile wenn „auto-crlf“ in den Settings aktiviert ist). Wenn auf der momentanen Seite keine Platz mehr ist, wird automatisch eine neue Seite angefangen.

Die meisten Daten von Objekten werden als einfacher Text gedruckt. Eine Formatierung der Daten ist möglich. Als Beispiel, wenn Sie ein Math Objekt ausdrucken, das eine Zahl enthält, können Sie auf das Objekt doppelklicken und das Datenformat wählen, z. B. die Anzahl der angezeigten Nachkommastellen. Weitere Informationen finden Sie auch im Kapitel Daten Typen und Formatierung.

Wenn Sie ein Grid Objekt ausdrucken oder ein anderes Objekt, das eine Liste von Vektoren enthält, werden die einzelnen Spalten durch ein TAB getrennt und eine neue Zeile für jede Reihe. Die Einstellung der TAB Positionen erfolgt in den Settings des Report Objektes (siehe auch Settings).

Ein Graph, Picture oder Panel Objekt wird als Bild gedruckt. Die Skalierung erfolgt so, daß es zwischen den linken und rechten Rand paßt. Sie können eine exakte Größe und Position mit der „Print (at)“ Action festlegen (siehe unten).

End - Schließen des Protokolls und Start des Druckvorgangs.

New page - Setzt das Protokoll auf einer neuen Seite fort. Kopfzeile wird gedruckt, wenn aktiviert (siehe Settings). Eine neue Seite wird auch automatisch angefangen, wenn das Ende der aktuellen Seite erreicht wird und weiterhin Daten gedruckt werden sollen.

Set position in - Verschiebt die aktuelle Druckposition für die nächste Print Aktion der Daten. Die Positionsangabe ist relativ zur oberen linken Ecke der Seite, innerhalb der Seitenbegrenzungen. Die Angabe erfolgt in cm oder inches, abhängig von der Einstellung in den Settings des Report Objektes. Die Position ist begrenzt auf den Rand der Seite, auch wenn eine Position außerhalb des Randes spezifiziert wird.

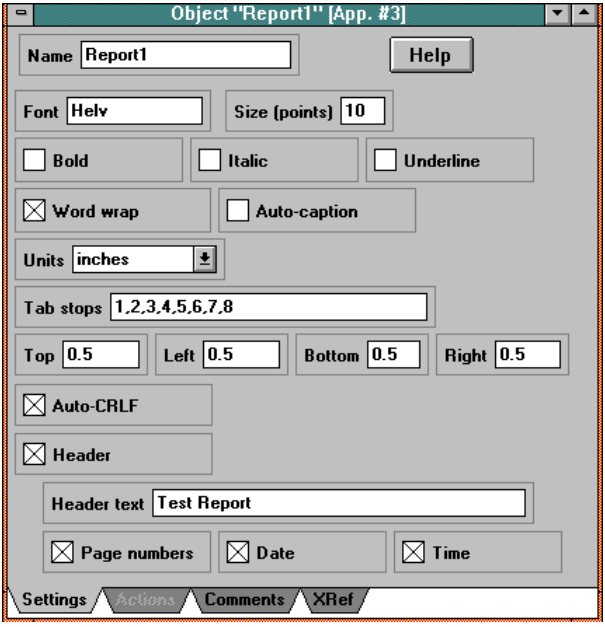
Print (at) - Druckt einen Parameter an einer spezifizierten Stelle der aktuellen Seite, in der angegebenen Größe. Wenn als x oder y Position ein Wert kleiner als Null eingegeben wird, wird an die momentan aktuelle Position gedruckt. Wenn bei der Höhe (height) oder Breite (width) ein Wert kleiner als Null angegeben wird, erfolgt eine Anpassung bis zum maximalen rechten bzw. unteren Rand. Beachten Sie, daß die Daten das spezifizierte Rechteck nicht ausfüllen müssen (obwohl Bilder, z. B. Grafik Objekte, so skaliert werden, daß sie es ausfüllen). Wie auch immer, die Daten werden so begrenzt, daß sie das Rechteck nicht überschreiten.

Cancel - Bricht das Protokoll ohne drucken ab.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

Daten - keine

Einstellungen (Settings)



Font - Der Name des Schriftsatzes, der zum Drucken benutzt werden soll. Für die meisten Anwendungen ist „Helv“ oder „Arial“ zu empfehlen. Auch dieses Setting können Sie, wie alle anderen auch, in eine Action List ziehen und dort während des Programmablaufs verändern.

Size (points) - Die Größe des Schriftsatzes in Punkten (1/72 eines Inch).

Bold, Italic, Underline - Zusätzliche Eigenschaften der Schriftart.

Word-wrap - Bei der Aktivierung dieser Option wird der Text, der den rechten Rand erreicht, automatisch in eine neue Zeile umgebrochen. Ansonsten wird Text, der über den rechten Rand hinaus geht, nicht gedruckt.

Auto-caption - Wenn aktiviert, wird jedes Mal, wenn ein TestObjekt gedruckt wird, der Name des Objektes mit einem Gleichheitszeichen vor dem Wert eingefügt. (z. B. „Display=3.45“).

Units - Einheit für die Tabulatoren, Ränder und Werte der „Print (at)“ und „Set position in“ Actions.

Tab stops - Position der Tabulatoren, gemessen vom linken Rand. Wenn TAB Zeichen oder Tabellen von Daten gedruckt werden, geben diese Tabulatoren den Spaltenabstand an.

Top, Left, Bottom, Right - Abstand des Seitenrandes.

Auto-CRLF - Bei Aktivierung dieser Option wird nach jeder Print Aktion eine neue Zeile angefangen. Ansonsten wird bei mehreren Print Aktionen immer in die gleiche Zeile gedruckt (Ausnahme siehe Word Wrap Setting), aber ein CRLF kann auch eingefügt werden, um definitiv zur nächsten Zeile zu wechseln.

Header - Aktiviert die Möglichkeit, einen konstanten Text im Kopf der Seite einzufügen. Der Text kann über eine oder zwei Zeilen gehen, je nach gewählter Einstellung.

Header Text - Text der Kopfzeile(n).

Page Numbers, Date, Time - Optionale Bestandteile der Kopfzeile (Seitenzahl, Datum, Uhrzeit).

Action List - keine

RS232 Objekt 

Das RS232 Objekt unterstützt die Kommunikation über die RS232 (COM) Schnittstellen des Rechners. Das Objekt kann natürlich auch mit RS-422 und RS-485 Geräten arbeiten.

Weitere Informationen zu RS232 Einstellungen und Fehlerbeseitigung finden Sie im entsprechenden Kapitel in diesem Handbuch.

Benutzerschnittstelle - keine

Aktionen

Open - Öffnet die RS232 Schnittstelle. Die Schnittstelle wird aber auch automatisch von Output- oder Enter-Actions geöffnet. Benötigt wird aber die Aktion zum Öffnen der Schnittstelle, wenn auf einlaufende Daten (Ereignisse) reagiert werden soll.

Output to - Sendet Daten zu einem an der RS232 Schnittstelle angeschlossenen Gerät. Die Action List Zeile erlaubt die Eingabe von beliebig vielen Parametern (Daten). Die Formatierung zum Senden der Daten kann durch Doppelklicken auf die jeweiligen Parameter verändert werden. Ein Abschlußzeichen (Terminator) zur Endeerkennung der Daten wird angehängt (CR, CRLF, LF, oder keines).

Der "wait for completion?" Parameter zeigt an, ob in der Zeile der Action List gewartet werden soll, bis die Daten gesendet sind oder sofort zur nächsten Zeile gegangen wird und eine Übertragung der Daten im Hintergrund erfolgt. Wenn Sie mehr Daten senden wollen, als Platz in der „output queue“ (siehe Settings) vorhanden ist, wird die Abarbeitung der Aktion solange verzögert, bis wieder genug Platz frei ist.

Enter from - Liest Daten von einem an der RS232 Schnittstelle angeschlossenen Gerät ein. Die Anzahl der angegebenen Bytes werden bis zu dem definierten Abschlußzeichen (Termintator) eingelesen. Bei

der Angabe keines Abschlußzeichens (none) wird exakt die spezifizierte Anzahl von Bytes eingelesen. Die Daten können von jedem Gerät auch im Hintergrund einlaufen, während TestPoint mit anderen Aufgaben beschäftigt ist, solange nur die Daten in die „input queue“ (siehe Settings) passen.

Close - Schließt eine RS232 Schnittstelle, so daß sie für andere Programme zur Verfügung steht. **Das Schließen einer Schnittstelle löscht alle einlaufenden Daten.**

Get # characters - Liest die Anzahl der Zeichen aus, die sich in der „input queue“ befinden.

Set mode of - Setzt die Übertragungsrate und andere Kommunikations-parameter der Schnittstelle. Wenn diese Aktion nicht eingesetzt wird, werden die Standard Parameter von Windows übernommen.

Set handshaking of - Schaltet das Hardware Handshaking (CTS und RTS Signale) bzw. Software Handshaking (XON und XOFF Zeichen) Ein oder Aus.

Set DTR - Setzt das DTR Signal auf Ein oder Aus. Das DTR Signal wird normalerweise beim Öffnen der Schnittstelle angeschaltet. Ist eine direkte Kontrolle der DTR Leitung aus Ihrem Programm erforderlich, kann das über diese Aktion geschehen.

Set RTS - Setze das RTS Signal auf An oder Aus.

Transmit break - Sendet ein „break“ Signal.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in die Action List.

Daten

Die Daten des RS233 Objektes sind das Ergebnis der letzten „Enter from„ einlesen Action.

Initial value - keine Daten

Einstellungen (Settings)

COM port # - Die Nummer der Schnittstelle, von 1 bis 9. Die Nummer entspricht der Einstellung im Windows Control Panel. (3.1x) oder in der Systemsteuerung (Win 95).

Default output term. - Das Endezeichen der Übertragung, das in jeder neuen „Output to“ Aktions Zeile eingefügt wird. Normalerweise das Wagenrücklauf Zeichen (CR). Wenn Sie Daten an ein Gerät senden möchten, das ein anderes Zeichen erfordert, können Sie dieses hier eintragen, so daß es standardmäßig in alle neuen Action List Zeilen übernommen wird.

Default input term. - Das Endezeichen der Übertragung, das in jeder neuen „Enter from“ Action Zeile eingefügt wird. Normalerweise das Wagenrücklauf Zeichen (CR).

Timeout (sec) - Diese Zeit wird maximal bei der Durchführung auf eine Ausgabe oder Eingabe Aktion gewartet. Bei der Ausgabe, wenn „wait for completion“ spezifiziert ist oder der Ausgabe Buffer voll ist, ist es die Zeit, die maximal gewartet wird. Beim Einlesen, wenn der Eingangs Buffer leer wird, bevor das Einlesen abgeschlossen ist, ist dieses die Zeitbegrenzung.

Event on receiving character: - Mit diesem Zeichen kann ein Event (Ereignis) beim Einlesen ausgelöst werden. Das Zeichen kann auch leer gelassen oder auf „none“ gesetzt werden, wenn kein Ereignis erforder-

lich ist. Normalerweise ist das Wagenrücklauf Zeichen eine gute Wahl, da dann der komplette empfangene Datensatz eingelesen wird.

Output queue size - Die Anzahl der Bytes, die zur Datenausgabe im Hintergrund reserviert werden (von 256 bis 8192).

Input queue size - Die Anzahl von Bytes, die bei der Dateneingabe für Daten reserviert werden, die vor einer "Enter from" Action auflaufen (von 256 bis 8192).

Action List

Die Action List des RS232 Objekts wird ausgeführt, sobald das Zeichen empfangen wird, das in den Settings spezifiziert ist. Falls kein Event (Ereignis) Zeichen angegeben ist, wird die Action List nicht ausgeführt.

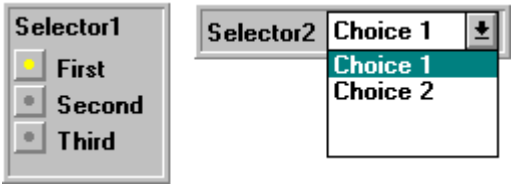
Wenn das Event (Ereignis) Zeichen beispielsweise auf CR gesetzt wird, wird das RS232 Objekt dazu veranlaßt, auf eingehende Daten automatisch zu reagieren.

Selector Objekt

Das Selector Objekt bietet eine Benutzerschnittstelle, die es dem Anwen-der erlaubt, eine Option aus einer Anzahl von Alternativen zu wählen.

Benutzerschnittstelle

Es gibt zwei Arten: Die Standard Windows Dropdown Liste, und eine Spalte von Knöpfen, von denen einer beleuchtet ist, wie es auf vielen Steuerungskontrollen verwendet wird. Diese Art von Selector Objekt versieht jeden Knopf mit einem Auswahl-Label und beleuchtet die getroffene Auswahl. Der Bediener kann auf jeden Knopf klicken, um unmittelbar eine neue Auswahl zu treffen.



Aktionen

Set - Setzt das Selector Objekt auf den Auswahlpunkt, der von der Parameterangabe gegeben ist. Falls der Parameterstring gleich einem der Auswahl-Labels ist, wird diese Auswahl getroffen. Andererseits wird der Parameter als eine Indexzahl interpretiert, von 0 bis N-1.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List Fenster.

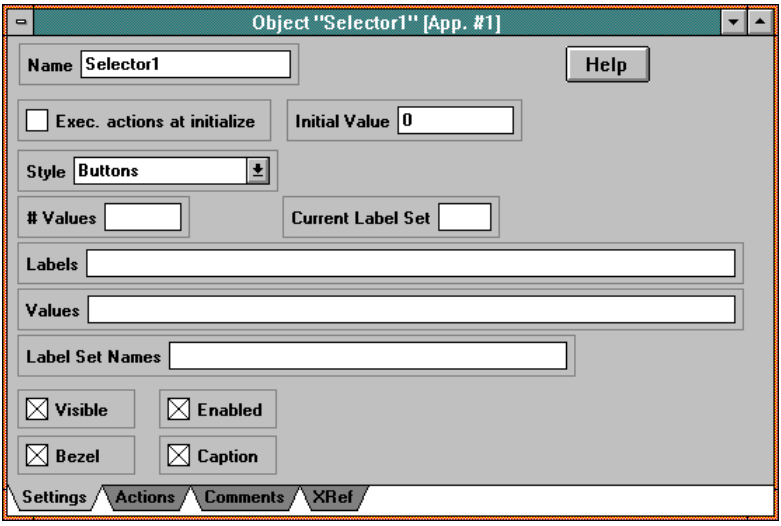
Daten

Der Datenwert hängt von den Settings ab. Falls keine "value" Settings spezifiziert wurden, entspricht der Datenwert dem Auswahlindex, von 0 bis N-1. Falls Value Settings gegeben sind, kann der Wert eine beliebige Zahl oder ein String sein.

Initial value

Ein Anfangswert kann in den Settings spezifiziert werden.

Einstellungen (Settings)



Style - Dropdown Liste oder Knöpfe. Siehe obige Beschreibung.

 # Values - Die Anzahl der möglichen Alternativen.

Current Label Set - Dieses Setting wird verwendet, wenn mehrere Label-Sets verwendet werden. Diese Option ist sinnvoll, falls ein

Selector Objekt für eine Funktion, aber für unterschiedliche Alternativen basierend, auf anderen Kontrollobjekten verwendet wird. Beispielsweise besitzt das ‘Range Setting’ eines Multimeters oftmals Alternativen, die von der gewählten Funktion abhängen. Mehrere Label Sets können in diesem Fall verwendet werden, wobei die Current Label Set und # Values Settings von der Action List des Funktionsobjekts kontrolliert werden können.
Das Current Label Set Setting ist immer eine Zahl, die von 0 beginnt.

Labels - Die Labels (Beschriftungen) für die Alternativen, die durch Kommata getrennt sind. Falls mehrere Label Sets verwendet werden, werden die einzelnen Sets durch Semicolons getrennt.

Values - Value Strings für jede Alternative, getrennt durch Kommata. Falls keine Value Strings gegeben sind, entspricht der Datenwert der gewählten Indexzahl von 0 bis N-1.

Label Set Names - Die Namen der einzelnen Label Sets, getrennt durch Kommata. Jeder Name eines Label Sets wird von einem Gleichheitszei-chen und einer zugehörigen Label Set Zahl gefolgt. Label Set Namen sind optional. Falls der Current Label Set gleich einer Zahl gesetzt wird, sind Namen nicht erforderlich. Label Set Namen sind dafür nützlich, Label Sets mit Hilfe von Strings auszuwählen.
Ein Beispiel eines gültigen Settings wäre: "DCV=0,ACV=1,Ohms=2". Siehe das Keithley 195 Multimeter Beispiel (K195.TST), in dem eine Beispielanwendung dieser Label Sets verwendet wurde.

Visible - Kontrolliert, ob das Auswahlobjekt sichtbar auf dem Panel erscheint.

Enabled - Falls nicht angekreuzt, werden die Caption und Labels hellgrau dargestellt und der Bediener kann keine Auswahl treffen.

Exec. Aktionen at initialize - Falls angekreuzt, wird die Action List bei der Programminitialisierung ausgeführt, nachdem der Initialwert des Selektors gesetzt wurde.

Initial value - Jeder Wert kann dem Selektor bei der Initialisierung des Programms zugewiesen werden.

Bezel - Falls angekreuzt, wird die Bezel (Rahmen) um das Objekt gezeichnet.

Caption - Falls angekreuzt, wird der Captiontext geschrieben.

Action List

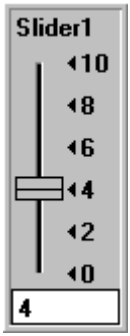
Die Action List wird ausgeführt, sobald die Auswahlalternative verändert wurde.

Slider Objekt

Das Slider Objekt ermöglicht eine kontinuierliche Variablenkontrolle, die interaktiv vom Bediener gesteuert werden kann. Jedesmal, wenn der Slider (Schieberegler) bewegt wird, wird seine Action List ausgeführt.

Benutzerschnittstelle

Der Schieberegler sieht wie ein Schiebepotentiometer aus. Er besitzt einen Balken, der mit der Maus gefaßt und auf- oder abbewegt werden kann. Mittels der Tastatur können die Auf- und Abpfeile dazu benutzt werden, den Schieberegler zu bewegen (falls die Shift Taste gleichzeitig gedrückt wird, bewegt sich der Schieber in größeren Schritten mit jedem Tastenanschlag).



Es gibt ebenfalls ein Eingabefeld unterhalb des Schiebereglers, in das der Benutzer den neuen Wert eintippen kann. In diesem Fall bewegt sich der Regler auf den neuen Wert in einem Schritt, ohne die entsprechenden Zwischenschritte, und erlaubt eine präzise Angabe ohne eine präzise Mauspositionierung.

Aktionen

Set - Setzt den Schieberegler auf den spezifizierten numerischen Wert. Falls der Wert außerhalb des Bereichs des Schiebereglers liegt, wird der Regler auf Anschlag gesetzt.

Bitte beachten Sie: : Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in die Action List.

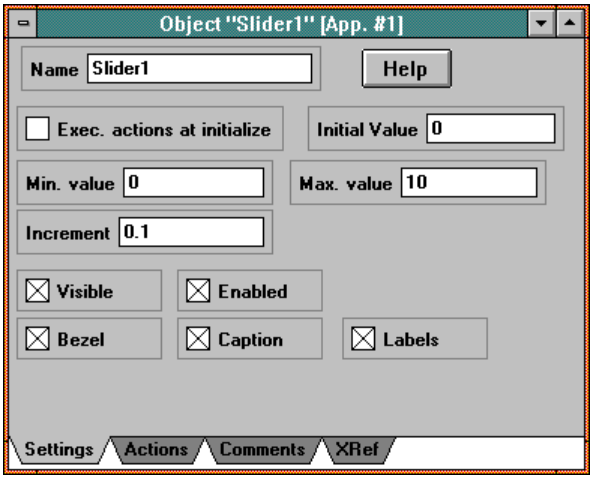
Daten

Der Datenwert des Sliders ist eine Zahl, die der Position des Reglers entspricht. Er kann verändert werden, indem die "Set" Action verwendet wird, oder der Benutzer mit der Maus bzw. der Tastatur eingreift.

Initial value

Der Anfangswert des Sliders hängt von den Settings ab.

Einstellungen (Settings)



Min. value - Der Wert für das untere Ende des Schiebereglers.

Max. value - Der Wert für das obere Ende des Schiebereglers.

Increment - Die Schrittweite zwischen gültigen Reglerschritten. Falls diese Schrittweite 0 ist, kann der Slider alle Werte annehmen. Falls sie größer als 0 ist, bewegt sich der Schieber in Schritten, das bedeutet, daß sein Wert immer ein Vielfaches des Increments ist.

Initial value - Der Startwert des Schiebreglers, sobald die Anwendung startet.

Exec. at initialization - Falls angekreuzt, wird die Action List ausgeführt, nachdem die Anwendung gestartet wurde.

Visible - Falls nicht angekreuzt, wird das Slider Objekt nicht auf dem Panel angezeigt.

Enabled - Falls nicht angekreuzt, reagiert der Schieberegler nicht auf Mausbewegungen oder Tastatureingaben.

Caption -Falls angekreuzt, wird der Objektname als Caption (Beschriftung) oberhalb des Sliders abgebildet.

Bezel - Falls angekreuzt, wird die Bezel (Rahmen) um das Objekt gezeichnet.

Labels - Falls angekreuzt, werden numerische Labels (Beschriftung) auf der rechten Seite des Sliders angezeigt. Die Anzahl der Labels und ihrer Werte wird automatisch anhand der Größe des Objekts auf dem Bildschirm und der Min/Max Werte bestimmt.

Action List

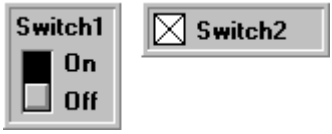
Die Action List des Sliders wird jedesmal ausgeführt, wenn sich sein Wert ändert. Die Update Rate unterschiedlicher Mausverschiebungen hängt davon ab, wie lange die Action List zu ihrer Ausführung benötigt.

Switch Objekt

Das Switch (Schalter) Objekt bietet eine Benutzerschnittstelle, die es dem Benutzer erlaubt, einen von zwei Zuständen auszuwählen.

Benutzerschnittstelle

Es gibt zwei verschiedene Formen: den Schiebeschalter, und die sogenannte Checkbox. Die Checkbox Form entspricht der Standard Windows Checkbox, mit dem Objektnamen als Caption (Beschriftung). Die Schiebeschalterform ist ein Auf-/Abschalter, mit dem Objektnamen als Caption und optionalen Labels neben den Auf- und Ab-Positionen.



Aktionen

Set - Setzt den Zustand des Schalters auf den Wert des Action Parameters. Falls der Parameter ein String ist, wird er mit den An- und Aus-Schaltern verglichen und der Schalter auf den korrespondierenden Zustand gesetzt. Falls es keinen passenden Label gibt, wird der Parameter in eine Zahl umgewandelt. Der Zahlen Parameter 0 setzt den Schalter auf aus, jeder andere Wert setzt ihn auf An.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in das Action List.

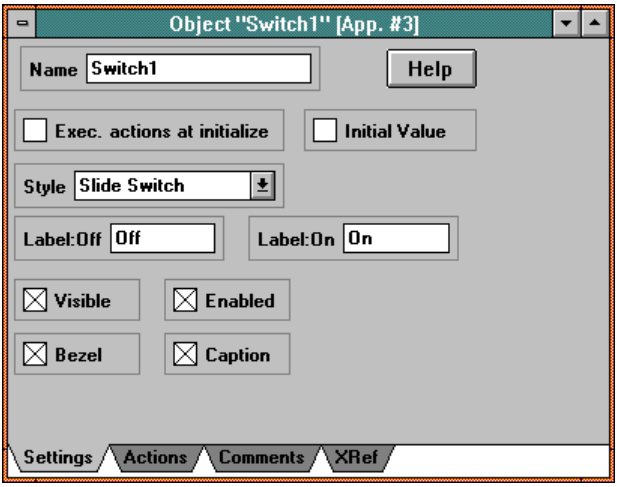
Daten

Der Datenwert des Switch Objekts entspricht einer Zahl, entweder 0 oder 1.

Initial value

Der Initialwert hängt von den Settings ab.

Einstellungen (Settings)



Style - Schiebeschalter oder Checkbox. Siehe obige Beschreibung.

Label: off and on - Text Labels für die Aus-/An Zustände des Schalters. Falls "Slide Switch" gewählt wurde, werden diese Labels neben dem Schalter angezeigt. Die Labels werden ebenso in der "Set" Action verwendet, um sie mit dem Action Parameter zu vergleichen.

Visible - Kontrolliert, ob der Schalter auf dem Panel sichtbar oder unsichtbar ist.

Enabled - Falls nicht angekreuzt, werden die Text Labels hellgrau dargestellt, und der Benutzer kann nicht den Schalterzustand verändern.

Exec. Aktionen at initialize - Falls angekreuzt, wird die Action List bei der Programminitialisierung ausgeführt, nachdem der Initialwert des Schalters gesetzt wurde.

Initial Value - Der Wert, auf den der Schalter bei der Programminitialisierung gesetzt wird. Falls die Checkbox angekreuzt ist, wird der Schalter auf der Stellung An initialisiert.

Bezel - Falls angekreuzt, wird die Bezel (Rahmen) um das Objekt gezeichnet.

Caption - Falls angekreuzt, wird der Caption (Beschriftung) Text gezeichnet.

Action List

Die Action List wird ausgeführt, sobald sich der Zustand des Schalters verändert, also der Benutzer oder das Programm mit der Set Action den Schalter schaltet.

Task Objekt 

Das Task Objekt erlaubt Ihnen eine erweiterte Kontrolle über die Multitasking-Fähigkeiten von TestPoint. Details und Beispiele finden Sie im Kapitel über Multitasking in diesem Handbuch. Das Task Object besitzt außerdem eine eigene Action List die beim Start/Verlassen der TestPoint Anwendung oder unabhängig davon ausgeführt werden kann.

Benutzerschnittstelle - keine

Aktionen

Die Aktionen für dieses Objekt sind in zwei Gruppen aufgegliedert: ***global*** und ***this task***. Die „global“ Aktion kann in jeder Action List eingesetzt werden. Die Aktionen unter dem Punkt „this task“ beeinflussen nur die Ausführung der Action List des Task Objekt.

Aktionen - global

Enter critical region - Die Abarbeitung des dadurch definierten Bereiches kann nur von einer Aktion gleichzeitig ausgeführt werden, eine zweite wird solange angehalten, bis die erste Aktion den Bereich verläßt. Das Ende des Bereichs wird durch „Exit critical region“ festgelegt.

Die sogenannten „critical regions“ erlaube den alleinigen Zugriff auf Daten oder Hardware, wenn Multitasking Probleme auftreten. „Critical regions“ werden in den meisten TestPoint Anwendungen nicht benötitgt.

Exit critical region - siehe "Enter critical region" oben.

Yield - Erlaubt die Bearbeitung und Ausführung anderer Action Lists und externer Windows-Programme, bevor die nächste Zeile der momentanen Action List abgearbeitet wird. Wird der Multitaking-Mode (die Einstellung erfolgt im Settings Fenster des Main Panels) auf „Disabled“ oder „Yield at end of loops“ gesetzt, erlaubt TestPoint nicht die Abarbeitung anderer Aktionen (Programme) nach jeder Zeile. Die Yield Action verhindert dieses Verhalten und räumt auch anderen Aktionen (Programmen) eine Ausführungszeit ein.

Halt application - Beendet eine Applikation. In einer runtime wird das TestPoint Programm geschlossen. In der TestPoint Entwicklungsumgebung wird zurück in den Edit Mode gewechselt. Ein Exit Code kann spezifiziert werden.

Return - Verursacht ein Verlassen der Action List, in der dieser Befehl ausgeführt wird, ohne daß der Rest abgearbeitet wird. Eine „Return“ Action verläßt die momentane Action List zur nächstübergeordneten. Als Beispiel, über einen Pushbutton wird eine Action List gestartet, hieraus wird wiederum die Action List eines Data-Entry Objektes aufgerufen. Wenn in der Action List des Data-Entry Objektes das „Return“ Kommando ausgeführt wird, erfolgt ein Verlassen dieser Action List zurück in die Action List des Pushbuttons, die dann fortgesetzt wird.

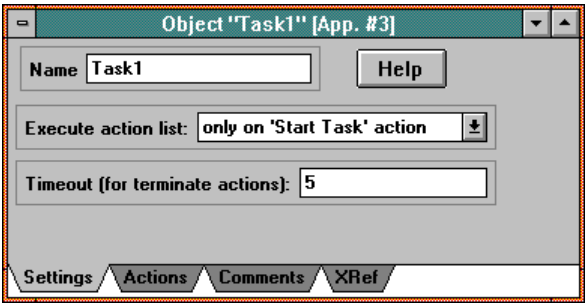
Aktionen - this task

Start - Startet die Action List des Task Objektes. Diese neue Task läuft parallel (Multitasking) mit der aufrufenden.

Stop - Stoppt die Abarbeitung der Action List des Task Objektes.

Daten - keine

Einstellungen (Settings)



Execute action list - Die Action List des Task Objektes kann bei der Initialisierung (wenn die Applikation startet), beim Beenden (wenn der Anwender die Applikation schließt) oder auf Anforderung (durch die „Start“ Action) ausgeführt werden.
Diese Einstellung wählt eine der drei Ausführungsarten.

Timeout - Wenn Sie die Einstellung „Execute at termination“ für das Task Objekt gewählt haben, können Sie hier die Zeit in Sekunden eingeben, die maximal der Action List des Task Objektes zur Ausführung zur Verfügung steht. Mit dieser zeitlichen Begrenzung wird verhindert, daß endlos Schleifen im Task Objekt ein Schließen der Anwendung unmöglich macht.

Action List

Wird je nach Einstellung des Task Objektes ausgeführt beim Start oder Beenden der Applikation, bzw. durch die „Start“ Action.

Text Objekt 

Das Text Objekt kann eingesetzt werden, um Schriftzüge oder Anweisungen an den Benutzer auf TestPoint Panels darzustellen. Wenn die Standard Beschriftung von Switch, Selector oder Display Objekten ausgeschaltet wird, kann mit dem Text Objekt ein eigener Text angezeigt werden.

Benutzerschnittstelle

Eine oder mehrere Text-Zeilen in einer veränderbaren Farbe und Schriftsatz:

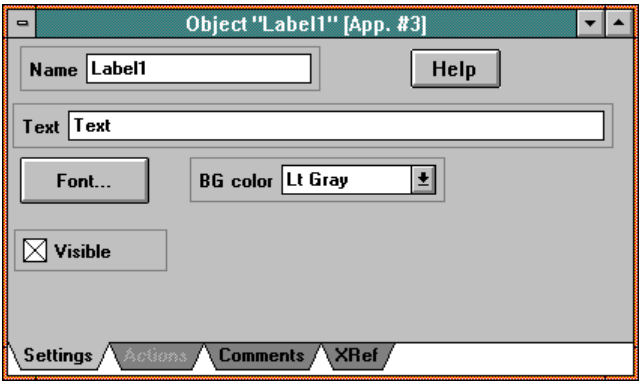
Um den Text wird kein Rahmen angezeigt, es sei denn, Sie aktivieren diese Einstellung in den Settings des Text Objektes. Das Objekt kann mit der Maus in der Windows üblichen Weise in der Größe skaliert werden.

Aktionen - keine

Achtung: Um die Eigenschaften (Settings) dieses Objektes aus einer Action List heraus zu verändern, ziehen Sie die entsprechende Einstellung einfach in die Action List.

Daten - keine

Einstellungen (Settings)



Text - Der Text, der angezeigt werden soll. Wenn mehrere Zeilen erforderlich sind, können Sie diese durch das Einfügen des „|“ Zeichen in den Text erzeugen.

Font - Das Betätigen dieses Knopfes zeigt Ihnen die Standard Dialog-Box zur Auswahl einer Schriftart, diese erlaubt Ihnen, Art, Größe, Attribut und Farbe der Schriftart auszuwählen.
Die voreingestellte Schriftart entspricht der, die TestPoint normalerweise für Objekt-Beschriftungen benutzt.

BG color - Hintergrundsfarbe des Text Objektes.

Visible - Ist diese Einstellung nicht ausgewählt, wird das Objekt nicht angezeigt. Mit dieser Einstellung können Sie aus einer Action List heraus das Objekt sichbar bzw. unsichtbar machen, um mit dem Anwender zu kommunizieren.

Action List - keine

Time (Zeit) Objekt 

Dieses Objekt ermöglicht Ihnen den Zugriff auf die momentane Zeit und Datum des Systems, um Meßwerte mit einer Zeitmarke zu versehen. Außerdem ist es mit diesem Objekt möglich, zeitgesteuerte Ereignisse zu generieren.

+ **Zeitgesteuerte Ereignisse können nur annähernd genau sein, da andere Windows Programme für eine unbestimmte Zeit die komplette Rechenleistung des Systems blockieren können.**
Diese Zeitbasis ist für Echtzeit-Systeme mit garantierten Antwortzeiten ungeeignet.
Benutzen Sie das A/D oder D/A Objekt für fest definierte Abtastzeiten, da diese auf einem Hardware-Zähler auf der Karte beruhen.

Benutzerschnittstelle - keine

Aktionen

Start timer - Startet die Erzeugung zeitbasierter Ereignisse mit der spezifizierten Intervallzeit in Sekunden.
Die Genauigkeit dieser Intervallzeit beträgt 100ms, optimal sind Intervallzeiten von 1 Sekunde und größer.
Bitte beachten Sie den Hinweis oben.

Stop timer - Beendet die Erzeugung von zeitbasierten Ereignissen.

Delay - Verzögert die Abarbeitung einer Action List um die angegebene Zeit in Sekunden, bevor mit der nächsten Zeile fortgefahren wird. Während dieser Verzögerung bekommen andere Action Lists und Windows Porgramme Zeit zur Ausführung. Die Genauigkeit der Verzögerung beträgt 50ms für große Zeiten, unter 50ms beträgt die Genauigkeit 1ms.

Achtung: Um die Eigenschaften (Settings) dieses Objektes aus einer Action List heraus zu verändern, ziehen Sie die entsprechende Einstellung einfach in die Action List.

Daten

Die Daten des Time (Zeit) Objektes ist die momentane Zeit, wenn der Wert gelesen wird, ensprechend den Settings des Time Objektes. Ist eine Änderung des Formats aus dem Ablauf heraus nötig, kann die Einstellung in eine Action List übernommen werden.

Initial value - Immer die momentane Zeit.

Einstellungen (Settings)

Time format - Das Format der zurückgegebenen Zeit, wenn auf die Daten des Objekts zugegriffen wird. Das können verschiedene Kombinationen aus Zeit und Datum sein, in unterschiedlichen String Formaten. Es kann auch die vergangene Zeit seit dem 01. Jan. 1970 sein, dieses dient zur Berechnung von Zeitdifferenzen (die Genauigkeit beträgt 55ms). Die Auswahl erfolgt in einem Dropdown Menü.

Action List

Die Action List des Time Objektes wird ausgeführt, wenn ein Zeitereignis ausgelöst wird. Die Zeitereignisse werden erst nach der „Start timer“ Action ausgelöst, zuzüglich der Zeit des angegebenen Intervals. Zeit-ereignisse sind sinnvoll, wenn periodisch Bedingungen zu überprüfen sind oder Daten von Geräten in bestimmten Intervallen zu erfassen sind. Beachten Sie aber, daß das Intervall nicht auf Sekundenbruchteile genau ist.

User-Defined Objekt USER

Das User-Defined (benutzer-definiertes) Objekt erlaubt jede beliebige Kombination von TestPoint Objekten und Action Lists zu einem einzigen TestPoint Objekt. Das kann hilfreich sein, um Details von Applikationen zu verbergen und die Objekt Liste des Haupt-Panels kurz zu halten. Außerdem ist es damit auch möglich, neue Objekte zu erstellen, die in verschiedenen Applikationen eingesetzt werden sollen. Ein User-Defined Objekt kann auch in das Stock Fenster gebracht werden und steht so immer wieder zur Verfügung.

Das User-Defined Objekt kann wie das Panel Objekt andere Objekte enthalten, sogenannte „child" Objekte. Das User-Defined Objekt erscheint im Objekt Fenster mit einem Pfeil nach Rechts, genau wie ein Panel, und der Pfeil kann dazu benutzt werden, die Objekte im User-Defined sichtbar zu machen. Diese „child" Objekte bestimmen die Funktion des User-Defined Objekts.

Achtung: Objekte zu einem User-Defined Objekt zusammenzufassen, ist kein Ersatz für den Einsatz von Sub-Panels (Panel-Objekt). Wenn Objekte zu einem User-Defined zusammengefügt werden, erscheinen Sie nicht mehr auf dem Haupt-Panel. Natürlich können Sie Panels in einem User-Defined Objekt haben, diese müssen Sie dann aber durch eine Show Action sichtbar machen.

Ein User-Defined Objekt kann durch den Befehl „Package as User-Defined" unter „Utilities" aus der Menüleiste erzeugt werden. Danach ist eine Bearbeitung wie in TestPoint üblich möglich.

Bestimmte Teile der Definition eines User-defined Objekts - seine Aktionen, die Benutzerschnittstelle, die Datenwerte, etc. - werden in speziellen Child Objekten gespeichert. Beispielsweise existiert ein Child Objekt namens "Data", das immer dann benutzt wird, wenn der Datenwert eines User-defined Objekts benötigt wird. Diese speziellen reservierten Objektnamen sind:

Data
Action Names
Action List
UI Name
Class Name
Settings
Initialize
Icon
Lock

Benutzerschnittstelle

Die Benutzerschnittstelle hängt von dem Child Object in dem User-defined Objekt ab. Die Benutzerschnittstelle kann von jedem ausgewählten Child Objekt gebildet werden. Die Auswahl wird getroffen, indem in dem "Edit User-defined" Dialog der Name des gewünschten Child Objekts spezifiziert wird.

Aktionen

Die möglichen Aktionen hängen von den Child Objekten in dem User-defined Objekt ab.

Diese Aktions Definitionen können normalerweise mit Hilfe des "Edit user-defind" Dialogfenster definiert werden (erreichbar über das Utilities/Edit User-defined Menükommando).

Die Implementation von Aktions Definitionen besteht darin, eine Tabelle der Zuordnungen innerhalb des Objekts in einem speziellen Child Objekt zu speichern. Falls es ein Child Objekt (Data Entry) namens "Action Names" (exakte Schreibweise und Groß-/Kleinschreibung sind wichtig), kontrolliert der Textwert dieses Objekts die möglichen Aktionen des User-defined Objekts. Der Text sollte aus einer Liste von Aktions Definitionen bestehen, die durch Semicolons getrennt werden. Jede Aktions Definition besteht aus einer Bezeichnung für die Aktion, gefolgt von einem Komma, dem Namen des Child Objekts, das zur Ausführung der Aktion benutzt wird, gefolgt von einem Doppelpunkt, und dem Aktions Namen innerhalb des Child Objekts. Beispielsweise wird für ein Child Pushbutton Objekt namens "Initialize" mit seiner Action List namens "Preset" folgender String in dem "Action Names" Objekt verwendet: "Preset,Initialize:Push".

Bitte beachten Sie: Sobald ein User-defined Objekt mittels dem Utilities Package aus dem User-Defined Menu Kommando erzeugt wird, wird das "Action Names" Objekt automatisch hinzugefügt, und es enthält die Aktionen für alle Benutzer Input Child Objekte.

Daten

Falls ein Child Objekt namens "Daten" existiert, wird der Datenwert dieses Objektes als Datenwert des User-defined Objekts verwendet. Falls es nicht existiert, wird dafür der Datenwert des ersten Child Objekts in der Objektliste verwendet.

Initial value

Es gibt keinen eingebauten Initialwert. Die Initialisierung kann jedoch vorgenommen werden (siehe weiter unten).

Initialization

Falls ein Child Objekt namens "Initialize" existiert, wird seine Action List bei der Programminitialisierung ausgeführt. Diese Action List kann jegliche Initialisierungsvorgänge vornehmen, die für das User-defined Objekt notwendig sind.

Settings (Einstellungen)

Falls ein Child Panel Objekt namens "Settings" existiert, wird dieses Panel für die Settings des User-defined Objekt verwendet. Das Child Panel kann beliebige Objekte beinhalten, die die Benutzerauswahl der Settings darstellen, und die Datenwerte dieser Objekte können wie jedes andere TestPoint Objekt in den Action Lists innerhalb des User-defined Objekts verwendet werden. Das Settings Panel sollte kein Eingabefeld für den Objektnamen enthalten - dieses wird automatisch von TestPoint hinzugefügt, sobald das Setting Fenster angezeigt wird.

Action List

Falls ein Child Action Objekt namens "Action List" existiert, gibt es auch eine Action List für das User-defined Objekt. Das Action List Objekt muß innerhalb einer anderen Action List eines Child-Objektes des User-defined Objekt ausgeführt werden. Die Bedingungen, die erfüllt sein müssen, wenn die Action List ausgeführt werden soll, können ganz von dem Designer des User-defined Objekts festgelegt werden. (Siehe Beispiele im Kapitel User-defined Objekte).

Icon

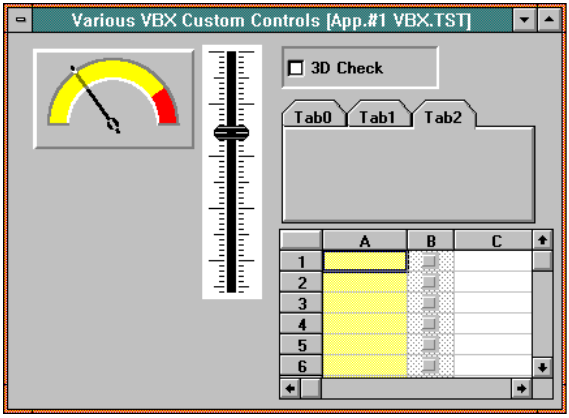
Falls ein Child Picture Objekt namens "Icon" existiert, benutzt das User-defined Objekt sein Bitmap als Icon. Bitte beachten Sie, daß TestPoint Objekt Icons immer aus 24 mal 24 Bit bestehen müssen. Falls es kein solches Child Objekt gibt, wird ein Standard User-defined Icon verwendet. Falls ein User-defined Objekt erzeugt wird, um es später wieder zu benutzen, oder um es auch an andere Programmierer zu verteilen, wird empfohlen, solch ein Bitmap anzufertigen. Die Bitmapdatei muß nicht mit dem Objekt weitergegeben werden (das Bitmap wird in das User-defined Objekt automatisch kopiert).

Class Name

Der Klassen-, oder Kategoriename für ein Objekt, wenn automatisch generierte Namen für Objekte vergeben und aus dem Stock kopiert werden. Beispiele für solche Class Namen sind "Pushbutton" oder "Switch". Die Class Namen für das User-defined Objekt können bestimmt werden, indem ein Child Data Entry Objekt namens "Class Name" implementiert wird. So ein Objekt wird automatisch erzeugt, sobald ein User-defined Objekt mit Hilfe des Utilities Package Kommando aus dem User-Defined Menu erzeugt wird.

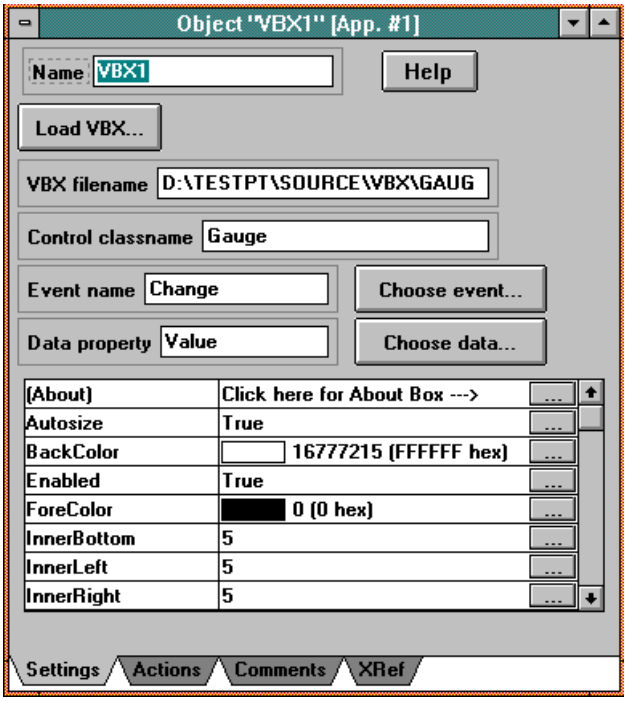
VBX Objekt

Das VBX Objekt ermöglicht es, vielerlei Arten neuer Kontrollelemente und Funktionen in TestPoint Anwendungen und auf die Panels zu implementieren:



VBX Controls sind ein Industrie Standard, der ursprünglich für Microsoft Visual Basic geschaffen wurde. Es gibt viele Software-hersteller, die VBX Control Softwarepakete produzieren, und hunderte dieser Controls sind auf dem Markt erhältlich. Dies beinhaltet Knöpfe, Schieberegler, analoge Anzeigeinstrumente, 3D Checkboxen, Tabellenkalkulationen, Datenbanken, etc.

Einstellungen (Settings)



Load VBX - Dieser Knopf macht eine Dialogbox sichtbar, in der das VBX Control File ausgewählt und geladen werden kann. Falls die Datei mehr als ein Control enthält, wird der Programmierer gefragt, welcher Typ verwendet werden soll.

VBX filename - Der Name der VBX Datei. Normalerweise muß dieses Setting nicht direkt verändert werden. Es ändert sich als Resultat des Load VBX Knopfes.

Control classname - Der Name der VBX Control Klasse, die aus der VBX Datei entnommen wird. Normalerweise muß dieses Setting nicht direkt verändert werden. Es ändert sich als Resultat des Load VBX Knopfes.

Choose event - Mit Hilfe dieses Knopfes kann das VBX Event (Ereignis) gewählt werden, das für die Triggerung der Ausführung der Action List von TestPoint verantwortlich ist. VBX Controls können mehrere verschiedene Ereignisse definieren. Die möglichen Ereignisse hängen von dem jeweiligen VBX Control ab. Man kann ebenso **all events** wählen, in diesem Fall wird man bei allen VBX Ereignissen benachrichtigt, und man kann die "Get last event" Action (siehe unten) verwenden, um zu bestimmen, welches Ereignis aufgetreten ist.

Event name - Der gewählte Event Name (siehe oben).

Choose Daten - Dieser Knopf wird verwendet, um die VBX "property" auszuwählen, die für den Datenwert des VBX Objekts benutzt wird. VBX Controls können verschiedene Properties (Eigenschaften) definieren, die wie TestPoint Settings verwendet werden können.

Daten property - (siehe oben)

Other settings (VBX properties)

Das Scroll Fenster am Ende der Settings listet alle Properties (Eigenschaften) des VBX Control auf. Die möglichen Properties hängen von dem jeweiligen VBX Control ab, und sollten in der Beschreibung des VBX dokumentiert sein. Diese Properties (Eigenschaften) können in Action Lists wie jedes andere TestPoint Setting gezogen werden.

Die Properties können modifiziert werden, indem der "..." Knopf auf der rechten Seite des Property Namens gedrückt wird.

Bitte beachten Sie, daß einige Properties als Runtime-only markiert sind. Sie können nur in Action Lists angesprochen werden und können nicht direkt verändert werden. Sie weisen deshalb auch keinen "..." Knopf auf.

Aktionen

Set property of - Mit dieser Aktion kann jede Property (Eigenschaft) des Controls per Name geändert werden. Für die meisten Properties ist dies nur eine Alternative für das Ziehen der Property direkt von dem Settings Panel. Für jene Properties allerdings, die Arrays beinhalten, erlaubt die Set Property Action, einen Indexwert zu spezifizieren.

Get property of - Ähnlich zu "Set Property", aber liest den Wert der Property. Der Datenwert des VBX Objekts wird zu dem Property Wert **eines** Zugriffs (siehe unter Datenwert).

Get last event - Diese Aktion liest die Information des letzten Events (Ereignisses), das das VBX Control geliefert hat. Man kann auswählen, welches Event TestPoint dazu veranlassen soll, die Action List auszuführen, oder man verwendet die Einstellung "all events". Im letzten Fall kann diese Aktion in Zusammenhang mit einer IF Entscheidung dazu verwendet werden, um zu entscheiden, welche Aktion ausgeführt werden soll. Die zurückgegebene Information besteht aus einem **LIST** Datentyp, der einige Items (Einträge) enthält, die von dem Event (Ereignis) abhängen: Das erste Item ist immer ein **STRING**, der dem Namen des Events (Ereignisses) entspricht. Die verbleibenden Listen Items hängen von dem Event (Ereignis) ab - die mitgelieferte Dokumentation zu dem VBX Control sollte Aussagen über die Verwendung der Event (Ereignis) Argumente enthalten. Beispielsweise hat ein Tastendruck Event (Ereignis) ein Argument: die Taste, die gedrückt wurde. Ebenso stellt die Event (Ereignis) Information den Datenwert des VBX Objekts für **einen** Zugriff dar (siehe unter Datenwert).

Add item to - Einige VBX Controls erlauben diese sogenannte "method". Beispielsweise können einige List Box Controls Strings zu der Liste mit "Add Item" hinzugefügt werden. Die Dokumentation des VBX Controls sollte diese Action beschreiben, sofern sie implementiert ist.

Remove item from - Ähnlich der "Add item" Action, aber löscht das Item.

Move - Ändert die Position und/oder Größe des Controls auf dem Panel. x,y,w, und h werden in Bildschirmpixeln angegeben.

Refresh - Zwingt das Control, neugezeichnet zu werden. Die meisten Controls erneuern sich selbst automatisch und benötigen diese Aktion nicht, dennoch kann sie in einigen Fällen nützlich sein.

Bitte beachten Sie: Um ein Setting dieses Objekts innerhalb einer Action List zu ändern, ziehen Sie das Setting aus dem Settingsfenster in die Action List.

Datenwert

Normalerweise ist der Datenwert des VBX Objekts definiert durch eine der "Properties". Beispielsweise ist er für einen Schieberegler oder ein Analoganzeigegerät die „Value Property“-Eigenschaft.

Bestimmte TestPoint Aktionen jedoch können **temporär** den Datenwert verändern. Die "get property of" und "get last event" Aktionen ändern den Datenwert des Objekts, auf diese Weise kann das Resultat dieser Aktionen in der nächsten Action Line verwendet werden (beispielsweise, um es in einem Display anzuzeigen oder in einem IF Statement zu verwenden). Eine solche Änderung ist **temporär**, da sie nur für **einen** Zugriff auf die Daten gilt. Das nächste Mal, wenn das Objekt in einer Action verwendet wird, wird der Datenwert erneut von der VBX Property bestimmt.

Falls nun der Property Array Wert in ein Display geschrieben werden soll, und er **daraufhin ebenso** in einem IF Statement verwendet werden soll, muß die "get property" Action ein zweites Mal ausgeführt werden.

Action List

Die Action List eines VBX Objekts wird ausgeführt, sobald das ausgewählte Event (Ereignis) auftritt. Die meisten VBX Controls besitzen mehr als ein Event (Ereignis), das sie verarbeiten können, beispielsweise MouseDown, MouseUp, etc. Der "Choose event" Knopf in dem Settings Fenster erlaubt dem Benutzer, zu wählen, welches Ereignis die TestPoint Action List auslösen soll, bzw. "all events" auszulösen.

VBX Software Lizenzen

VBX Controls stellen eine separate Software Komponente dar, die von dem Hersteller bezogen wird und dann mit TestPoint dazu verwendet wird, eigene Applikationen zu erstellen. Die Lizenzpolitik der unterschiedlichen Hersteller unterscheidet sich, dennoch erlauben es die meisten VBX Anbieter, kostenlose Runtimes wie in TestPoint zu erstellen. Dennoch muß die VBX Software erworben werden, um diese Applikationen zu erstellen.

Generell wird die Lizenzierung von Dateien mit der Endung .LIC im Windows Verzeichnis kontrolliert, die Lizenz erstreckt sich nicht auf die Weitergabe dieser Dateien, nur der .VBX Dateien.

Sobald ein VBX in den Editor geladen wird, oder eine Datei mit einem VBX geöffnet wird, werden in TestPoint die Lizenzbestimmungen angewendet, und der Programmierer bekommt typischerweise eine Warnung, falls nur die Runtime Lizenz des VBX Controls vorliegt. Sobald diese Controls in einer TestPoint Runtime vorliegen, ist keine Lizenzdatei erforderlich.

Ein Beispiel ist TestPoint beigefügt: VBX.TST. Dieses zeigt einige erhältliche Control Objekte. Bitte beachten Sie, daß wir die Runtime Version dieser VBX Controls mitliefern, das bedeutet, falls Sie versuchen, VBX.TST in den TestPoint Editor zu laden, wird dies nicht

gelingen. VBX Controls sind Software, dessen Copyright bei den Autoren liegt, meistens ist es aber erlaubt, Runtimes frei weiterzugeben.

Sobald Runtimes mit VBX Controls erstellt werden, muß die "Add Files" Option dazu verwendet werden, die .VBX Files der Runtime Disk beizufügen!

Wichtige Informationen

Einige wenige VBX Custom Controls erfordern Visual Basic V2.0 oder V3.0 und arbeiten nicht mit TestPoint zusammen. Es gibt jedoch davon nicht viele. Sie bekommen eine Nachricht, wenn TestPoint versucht, solche Dateien zu laden.

VBX Controls sind externer Code, der zu TestPoint hinzugefügt wurde. Einige wenige dieser Controls haben bekannte Fehler in ihrer Software. Beispielsweise erzeugt das 3D Menu Control in den VBTools (die sowieso nicht für TestPoint sinnvoll sind) eine allgemeine Schutzverletzung, falls seine "popupcreate" Property ohne die Initialisierung anderer Werte vorgenommen wird. Derselbe Zusammenbruch passiert, sobald das Control in Visual Basic verwendet wird.

Falls irgendein Problem mit VBX Controls auftritt, und Sie Zugang zu Visual Basic oder Borland oder Microsoft C++ haben, versuchen Sie bitte das Problem mit diesen Programmiersprachen zu duplizieren. Somit kann das Problem als VBX-Problem identifiziert werden.

Mathematische Funktions-Referenz

+	-	*	/	^	
mod	( )	[ ]			
not	and	or	xor		
<	<=	>	>=	=	<>
strEqual	strCompare				
floor	ceil	round	int	abs	sgn
rnd	rndNormal				

Trig, Log, usw. Funktionen

sin	cos	tan
asin	acos	atan
	atan2	
sinh	cosh	tanh
sqr		
log	log10	exp
pow2	pow10	
factorial		

&	asc	chr
	instr	substr
length	strEqual	strCompare
upcase	lowcase	
strtrim		

Konvertierungs-Funktionen

hex
bin
str
convertToNumber
convertToVector
convertToArray

valhex
valbin

convertToString
convertToList

sum

min
max
rms

avg
median
minindex
maxindex

mode
stddev

dim
vectorReplace
zero

reverse
determinant
inverse
appendVector
generateSin
generateTriangle
generateSquare
generateSteps
decimate
reshapeArray

index subarray
arrayReplace
one ramp
idn
rotate

transpose matMultiply
vector
generateCos
generateRamp
generate
generateRepeat
interleave

list sublist select

Kurvenanpassungs-Funktionen

fitLinear

fitPolynomial
interpolate
polynomial

fitExponential
fitLogarithmic

lowpass

highpass bandpass
notch

smoothAvg
smoothAvgCentered
smoothMedian
firFilter
iirFilter

Signalanalyse Funktionen

FFT
hamming

blackmanharris3
integrate
histogram
convolve
solve
sort

IFFT
hanning blackman
poisson
blackmanharris4
derivative derivative2

cliplower	clipupper	cliprange
type		
if		
pi	e	
length		
previous		

Strings zusammenfügen

x & y

Verbindet zwei Strings zu einem. Falls erforderlich, werden numerische Operanden in Strings konvertiert.

Beispiele

"ABC" & "def"	"ABCdef"
str1 & "\r\n"	dabei ist str1 "XYZ", Ergebnis ist "XYZ\r\n"

Absolut Wert

abs(x)

Wenn das Argument negativ ist, wird es positiv. Andernfalls wird das Argument unverändert zurückgegeben.

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der gleichen Dimension wie das Argument.

Beispiel

abs(3.2)	3.2
abs(-2.4)	2.4
abs(0)	0
abs(vector(1,-2,0))	vector(1,2,0)
abs(list(3,-2))	list(3,2)

Zusammenfügen von Vektoren oder Listen von Vektoren

appendVector(x,y,...)

Verbindet null oder mehr Vektoren zu einem längeren Vektor (Synonym für **vector()**). Wenn kein Argument angegeben ist, wird ein Vektor der Länge Null erzeugt. Beliebige Zahlen von Vektoren oder Skalare können in dieser Funktion zusammengefügt werden. Sie ist ebenfalls nützlich bei der Erzeugung von konstanten Vektoren (siehe Beispiel unten).

Diese Funktion kann ebenfalls Listen mit mehreren Vektoren an andere Listen anhängen. Wenn ein Argument eine Liste von Vektoren ist, dann müssen alle Argumente Listen von Vektoren sein. Die Listen werden Punkt für Punkt kombiniert und brauchen somit nicht die gleiche Anzahl von Punkten zu beinhalten. Die Anzahl der Punkte in der Ergebnis-Liste ist die gleiche, wie die Anzahl des ersten Arguments.

Beispiele

appendVector(x,y)

mit x=vector(1,2)
und y=vector(3,4),
Ergebnis ist vector(1,2,3,4)

appendVector(1,2,3,4)

vector(1,2,3,4)

appendVector(l1,l2)
l1=list(vector(1,2),vector(10,20)

mit

und l2=list(vector(3,4),vector(8))
Ergebnis:

list(vector(1,2,3,4),vector(10,20,8))

Ersetzen eines Elementes in einem Feld (Array)

arrayReplace (array,row,col,value)

Ersetzt das gegebene Element in dem Feld mit dem angegebenen Wert. Rückgabewert ist das modifizierte Feld.

Bitte beachten Sie, daß das Math Objekt das modifizierte Array enthält, sobald diese Aktion ausgeführt wurde. Das originale, in das Parameterfeld 'array' eingesetzte, Array wird **nicht** modifiziert.

Eine Möglichkeit, mit nur einem Array vor und nach der Aktion zu operieren, stellt die Verwendung des **previous** Befehls dar. Hier wird das Ursprungsarray, das im Datenwert des Math Objektes gespeichert sein muß, auch wieder modifiziert im Math Objekt gespeichert:

```
arrayReplace( previous(), i, newvalue)
```

Arithmetische Operatoren

x+y	**Addition**
x-y	**Subtraktion**
x*y	**Multiplikation**
x/y	**Division**
x^y	**Exponentiation**
x mod y	**Modulus**

Die bekannten arithmetischen Operatoren. Sie können auf Vektoren, Zahlen, Feldern oder Listen angewendet werden.

Beispiele

3+4	7
2-6	-4
3*2.1	6.3
7/4	1.75
2^3	8
11 mod 3	2
vector(1,2)*2	vector(2,4)
vector(2,3)*vector(4,8)	vector(8,24)
list(1,2,3)-1	list(0,1,2)
list(1,2,3)+list(0,2,5)	list(1,4,8)

ASCII Code eines Zeichens

asc(str)

Gibt den ASCII-Zeichen-Code für das erste Zeichen des übergebenen Strings zurück.

Beispiele

asc("ABC")	65
asc("\r")	13

Durchschnitt oder arithmetisches Mittel

avg(v)

Ermittelt das arithmetische Mittel der Werte in einem Vektor oder Feld. Diese Funktion kann sowohl auf Vektoren als auch auf Felder angewendet werden. Bei letzteren wird eine Liste mit Mittelwerten zurückgegeben.

Beispiele

avg(vector(1,2,3,4))	2.5
avg(list(vector(1,2),vector(3,4))	list(1.5,3.5)

Bandpass Filter

bandpass(Kurve,Abtastrate,CutLow,CutHigh)

Filtert einen Vektor (Kurvenform) mit Hilfe der Bandpass Frequenz Funktion. Dabei wird die Abtastfrequenz sowie das untere und obere Ende des Frequenzbereiches des Bandpasses angegeben.

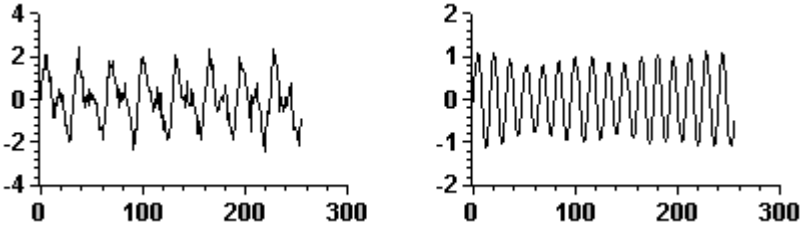
Der Filter Algorithmus ist:

1) Berechnung der FFT der Kurvenform und Ermittlung dessen Spektrums
2) Nullsetzung der Frequenzen außerhalb des Bandpasses
3) Berechnung der inversen FFT

siehe ebenfalls: lowpass(), highpass(), notch()

Beispiele

Anwendung der Bandpassfilter-Funktion **bandpass(wave,1000,40,80)** auf die linke Kurven ergibt die rechte Kurvenform:



Konvertierung einer Zahl in einen Binärstring

bin(x)
oder bin(x,n)

Konvertiert eine Zahl in einen String aus Einsen und Nullen. Dabei ist die Anzahl der Digits ein optionaler Parameter zwischen 1 und 32 (Bit).

Beispiele

bin(5)	"101"
bin(5,4)	"0101"
bin(0,4)	"0000"

Aufrunden

ceil(x)

Rundet das Argument auf die nächsthöhere ganze Zahl auf.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs wie das Argument.

Beispiele

ceil(4.2)	5
ceil(3.8)	4
ceil(0)	0
ceil(-2.2)	-2
ceil(vector(1.2,-2.3))	vector(2,-2)

Zeichen zu einem ASCII Code

chr(x)

Konvertiert eine ASCII-Code-Zahl in das zugehörige Zeichen.

Beispiele

chr(65)	"A"
chr(48)	"0"
chr(13)	"\r" (das „Carriage Return“ Zeichen.)

Abschneiden an der unteren Grenze

cliplower(x,Grenzwert)

Schneidet das Argument an dem gegebenen unteren Grenzwert ab. Wenn das Argument über dem Grenzwert ist, wird das Argument unverändert zurückgegeben; ist es unter dem Grenzwert, wird der Grenzwert zurückgegeben.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs wie das Argument.

Beispiele

clipLower(3.4,3)	3.4
clipLower(3.4,4)	4
clipLower(vector(1,2,3,4),3)	vector(3,3,3,4)

Abschneiden innerhalb eines Bereiches

clipRange(x,UntereGrenze,ObereGrenze)

Schneidet das Argument innerhalb seiner oberen und unteren Grenze aus. Wenn das Argument innerhalb der Grenzwerte liegt, wird es unverändert zurückgegeben. Außerhalb der Grenzen wird der zugehörige Grenzwert zurückgegeben.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs wie das Argument.

Beispiele

clipRange(3.4,3,5)	3.4
clipRange(3.4,4,5)	4
clipRange(vector(1,2,3,4),2,3)	vector(2,2,3,3)

Abschneiden am oberen Grenzwert

clipupper(x,Grenzwert)

Schneidet das Argument an dem gegebenen oberen Grenzwert ab. Wenn das Argument unter dem Grenzwert ist, wird das Argument unverändert zurückgegeben; ist es über dem Grenzwert, wird der Grenzwert zurückgegeben.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs wie das Argument.

Beispiele

clipUpper(3.4,3)	3
clipUpper(3.4,4)	3.4
clipUpper(vector(1,2,3,4),3)	vector(1,2,3,3)

Vergleichsoperatoren

< **kleiner als**
> **größer als**
<= **kleiner als oder gleich**
>= **größer als oder gleich**
= **gleich**
<> **ungleich**

Die Familie der Vergleichsoperatoren. Das Ergebnis ist 0 für falsch und 1 für wahr.

Diese Funktionen können auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das Ergebnis ist in jedem Fall eine einzige Zahl.

Werden die Funktionen auf eine Liste angewendet, wird der Vergleich mit jedem einzelnen Punkt der Liste durchgeführt. Das Ergebnis ist eine Liste von Ergebnissen, die mit den einzelnen Positionen in der Liste korrespondiert.

Wenn die Funktionen auf Strings angewendet werden, werden die Strings nach ASCII Zeichen-Code der Reihe nach verglichen.

Beispiele

3<4	1
5<3	0
vector(1,2,3)<4	1
vector(1,2,5)<4	0
list(3,4,5)<=4	list(1,1,0)

convertToNumber(x)

Wenn x eine Zahl ist, wird der Wert unverändert zurückgegeben. Wenn x ein String ist, wird er konvertiert (z.B. „3.45E2“ wird zu 345). Wenn x ein Vektor oder Feld ist, wird das erste Element zurückgegeben. Wenn x eine Liste ist, wird dessen erster Punkt nach diesen Regeln konvertiert.

Wenn x „nodata“ ist; wird 0 zurückgegeben.

convertToString(x)

Wenn x eine Zahl ist, wird sie unter Benutzung der „Auto-Formatierung“ konvertiert (z. B. aus 3.45 wird „3.45“).

Wenn x ein String ist, wird er unverändert zurückgegeben. Andere Datentypen werden konvertiert, indem die Standard Ausgabeformatierung verwendet wird (z. B. wird ein Vektor konvertiert, indem zwischen den Elementen ein CRLF eingefügt wird).

convertToVector(x)

Wenn x eine Zahl oder ein String ist, wird ein Ein-Element-Vektor zurückgegeben.

Wenn x ein Vektor ist, wird er unverändert zurückgegeben.

Wenn x ein Feld ist, wird es in einen eindimensionalen Vektor umgewandelt.

Wenn x eine Liste ist, werden die einzelnen Punkte als Vektor zusammengehängt.

convertToArray(x)

Wenn x eine Zahl oder ein String ist, wird ein 1 mal 1 Feld zurückgegeben.

Wenn x ein Vektor ist, wird ein n mal 1 Feld zurückgeben.
Wenn x ein Feld ist, ist der Rückgabewert unverändert.
Wenn x eine Liste ist, werden die Elemente Zeilen im Feld. Die Anzahl der Spalten ist gleich dem längsten Vektor. Sollte ein Feld in der Liste enthalten sein, wird eine Fehlermeldung erzeugt.

convertToList(x)

Wenn x ein Vektor oder ein Feld ist, wird eine Liste aus diesen Elementen geformt.

Andere Datentypen werden als Liste mit Einzelelementen zurückgegeben.

Beispiele

convertToString(vector(1,2))	"1\r\n2\r\n"
convertToVector(list(1,2,vector(3,4)))	vector(1,2,3,4)
convertToList(vector(1,2,3))	list(1,2,3)

Konvolution von Vektoren

convolve(x,y)

Berechnet eine Konvolution der vollen Länge der beiden Vektor-Argumente. Wenn x ein Vektor der Länge M ist und y ein Vektor der Länge N, dann ist das Ergebnis ein Vektor der Länge M+N-1.

Diese Funktion kann auch auf Listen von Vektoren angewendet werden. In diesem Fall werden die Vektoren in jeder Liste konvolutiert. Das Ergebnis ist eine Liste der Einzelergebnisse.

Die **Kreuz-Korrelation** von zwei Vektoren x und y wird berechnet:

```
convolve(x,reverse(y))
```

Die **Auto-Korrelation** eines Vektors x wird berechnet:

```
convolve(x,reverse(x))
```

Dezimierung eines Vektors (nehme jeden n-ten Punkt)

decimate (vect, n)

Verringert die Länge eines Vektors dadurch, das nur jeder n-te Punkt verwendet wird, beginnend mit dem ersten Element.

Die Funktion kann auch auf Listen von Vektoren angewendet werden und liefert dann als Rückgabewert eine Liste mit den dezimierten Ergebnissen.

Diese Funktion ist nützlich bei der Reduktion der effektiven Abtastrate für A/D-Werte zu Anzeigezwecken, während mit hoher Datenrate gespeichert oder Berechnungen durchgeführt werden können.

Numerische Differentiation

derivative(yvect,dx)
oder derivative(yvect,xvect)

Berechnet die numerische Differentiation einer gegebenen Kurve (Vektor von y-Werten). Die x-Werte haben jeweils den gleichen Abstand, Schrittweite dx, wenn das zweite Argument eine Zahl ist. Wird ein Vektor als zweites Argument angegeben, wird die Schrittweite durch dessen Werte definiert. Die zweite Form dieser Funktion erlaubt ungleiche Intervalle zwischen den Daten.

Der verwendete Algorithmus ist eine Kurvenanpassung an ein Polynom 4. Ordnung mit jeder Gruppe aus fünf Punkten, die dann berechnet wird als erste Ableitung dieses Polynoms. Das Ergebnis der Kurvenanpassung wird benutzt für die Ableitung am Mittelpunkt der Fünf-Punkte-Gruppe. Der Vektor muß mindestens fünf Punkte groß sein.

derivative2 (yvect,dx)
oder derivative2 (yvect,xvect)

Diese Variante der Ableitungsfunktion benutzt Polynome 2. Ordnung, die angepaßt werden an jeweils 3 Punkte der Kurve. Dadurch ist diese Funktion deutlich schneller als **derivative**.

Determinante einer Matrix

determinant(x)

Berechnet die Determinante eines quadratischen Zahlenfeldes. Wenn das Argument kein quadratisches Feld ist, wird eine Fehlermeldung ausgegeben. Eine Fehlermeldung wird ebenfalls angezeigt, wenn die Matrix „singulär“ ist (Determinantenberechnung dabei unmöglich).

Beispiele

```
determinant(idn(3,3))                1
determinant(a)          dabei ist a:  1   3
                                      2   2
                        Ergebnis ist -4
```

Dimension (Größe) eines Vektors oder Feldes

dim(v)
oder dim(a,n)

Gibt als Ergebnis die Größe eines Vektors oder die Dimension eines Feldes zurück (n sollte 0 oder 1 sein, um #Zeilen oder #Spalten zu erhalten).

Beispiel

dim(vector(2,3,4))	3

e (Basis des Natürlichen Logarithmus)

e()

Diese Funktion gibt den genauen Wert der Konstante „e“ zurück.

e hoch x (e^x)

exp(x)

Berechnet e hoch x.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs wie das Argument.

Beispiele

exp(1)	2.71828...
exp(vector(0,1))	vector(1,2.71828)

Fakultät einer Ganzzahl (Integer)

factorial(x)

Berechnet x!, was definiert ist als x*(x-1)*(x-2)...*2*1, für beliebige ganzzahlige Werte größer oder gleich 1. Wenn x keine Ganzzahl ist, wird nach dem Dezimalpunkt angeschnitten. Ist x negativ, wird automatisch der Absolutwert (abs(x)!) berechnet und das Ergebnis mal (-1) genommen.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs und der gleichen Dimension wie das Argument.

Beispiele

factorial(3)	6
factorial(vector(1,2,3))	vector(1,2,6)

Schnelle Fourier Transformation

FFT(Abtastrate,Kurvenform)

Berechnet die FFT (Fast Fourier Transformation) einer Kurvenform (eines Vektors).

Die Rückgabewerte bestehen aus einer Liste von drei Vektoren:
die Frequenzen jeder Komponente des Spektrums
die Amplituden jeder Komponente
und die Phase (im Bogenmaß) jeder Komponente.

Das Ergebnis ist sehr einfach als x-y-Grafik darstellbar, weil die Frequenzen an der x-Achse und Amplitude und Phase an der y-Achse dargestellt werden können.

Auf die einzelnen Teile des Ergebnisses kann über die **select()**-Funktion oder durch die Benutzung der TestPoint Daten Formatierungsfunktion (Doppelklick auf das Objekt in der Action List) zugegriffen werden.

Beispiel

```
FFT(1000,wave)
select(FFT(1000,wave),1)          (um nur die Amplituden zu erhalten)
```

Oder in einer Action List:

```
1) Calculate    FFT*      with freq=1000, wave=A/D1
2) Set          Grid1     to FFT:magnitudes
```

wobei der FFT-Datentyp definiert wurde als Liste von drei Vektoren durch Doppelklick auf die Buchstaben FFT in der Zeile 1 der Action

List und der Definition im Menü. FFT* wird dann als Parameter in der Zeile 2 benutzt. Wieder ein Doppelklick und die Menüauswahl ist da.

Finite Impulse Response filter

FIRfilter(Kurvenform,fir)

Berechnet die FIR Filterfunktion einer gegebenen Kurvenform. Das **Kurvenform** Argument muß ein Vektor aus Zahlen sein. Das **fir** Argument ist ebenfalls ein Zahlenvektor, der die FIR Kurve darstellt. Beispielsweise hat ein idealer Tiefpassfilter (Rechteckimpuls im Frequenzbereich) die Funktion sin(x)/x als Sprungantwort.

Die Filterung wird durch Konvolution der beiden Vektoren durchgeführt. Angemessene Sprungantwort-Vektoren können mit verschiedensten Mitteln berechnet werden, wie z. B. der McClellan-Parks Algorithmus, der in den meisten Büchern über Digitale Signalverarbeitung beschrieben ist.

Siehe auch: lowpass(), highpass(), bandpass(), notch(), IIRfilter().

Referenzen:
McClellan, Parks, Rabiner. A computer program for designing optimum FIR linear phase digital filters. *IEEE Transactions on Audio and Electroacoustics,* AU-21(6):506-526, 1973.
Rabiner and Gold. *Theory and Application of Digital Signal Processing.* Prentice Hall, 1975.

Die TestPoint FFT-basierenden Filterfunktionen: **lowpass, highpass, bandpass,** und **notch** sind generell bessere Filterfunktionen, außer Sie benötigen eine spezielle FIR-Filter Implementation.

fitLinear(xvect,yvect)
fitExponential(xvect,yvect)
fitLogarithmic(xvect,yvect)
fitPolynomial(xvect,yvect,order)

Paßt eine Kurve den gegebenen x- und y-Datensätzen an. Die Parameter xvect und yvect müssen Vektoren aus Zahlen sein und die gleiche Größe haben. Die Anpassung wird mittels der Methode der kleinsten Summenquadrate berechnet. Der Rückgabewert ist ein Vektor, der die Koeffizienten enthält.

Benutzte Funktionen und Rückgabewerte:

Linear	bx+a
Rückgabe	vector(a,b)
Exponential	exp(bx)+a
Rückgabe	vector(a,b)
Logarithmisch	b*log(x)+a
Rückgabe	vector(a,b)
Polynomial	a+b*x+c*x^2+...
Rückgabe	vector(a,b,c,...)

Abrunden

floor(x)

Rundet das Argument ab.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs und der gleichen Dimension wie das Argument.

Beispiele

floor(4.2)	4
floorl(3.8)	3
floorl(0)	0
floor(-2.2)	-3
floor(vector(1.2,-2.3))	vector(1,-3)

Kurvenformen erzeugen

generateSin(n,period)
generateCos(n,period)
generateTriangle(n,period)
generateRamp(n,period)
generateSquare(n,period,duty)
generate(n,period,function)

Erzeugt einen Vektor mit Kurvenformwerten der Länge n mit der angegebenen Anzahl an Punkten pro Periode. Die Rechteck-Funktion hat einen optionalen Parameter duty, der die Symmetrie der Kurve bestimmt (duty=0.5 ergibt eine symmetrische Rechteck-Kurve).

Die **generate(n,x,Funktion)** Funktion enthält als dritten Parameter eine Index-Nummer (siehe unten) der ausgewählten Kurve. Die Indizes sind:

0	Sinus
1	Cosinus
2	Dreieck
3	Rampe
4	Rechteck (0.5 duty cycle)

Die Amplitudenwerte und der Offset können eingestellt werden, indem auf das Ergebnis der „generate“-Funktionen arithmetische Funktionen angewendet werden.
Wenn die Phase der Kurvenform nicht bei Null beginnen soll, erzeugen Sie eine längere Kurve als benötigt, benutzen Sie die **subarray** Funktion und überspringen Sie die Punkte in dem Anfangsbereich des Vektors.

Beispiel

3*generateSin(128,64)+1

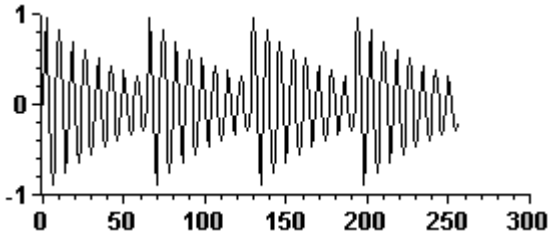
Kurvenformen erzeugen - Wiederholungen eines Vektors

generateRepeat(vect,n)

Wiederholung des gegebenen Vektors n mal, um die Länge zu vergrößern.

Beispiel

generateRepeat(generateSin(64,8)*exp(ramp(64)*-0.02),4)



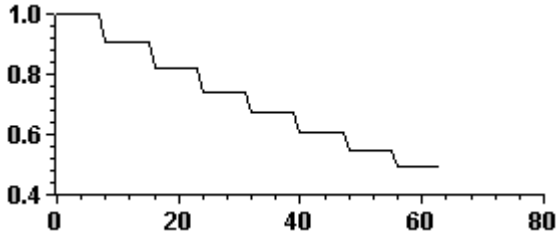
Kurvenformen erzeugen - Vektorelemente wiederholen

generateSteps(vect,n)

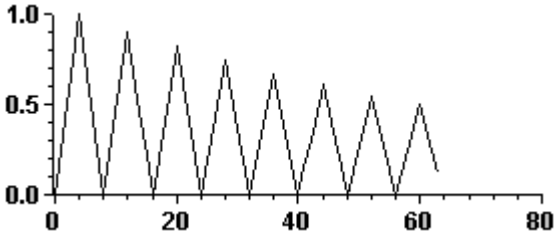
Wiederholung jedes Elements eines Vektors n mal. Das Ergebnis ist n mal länger als das Argument.

Beispiele

generateSteps(exp(ramp(8)*-0.1),8)



generateSteps(exp(ramp(8)*-0.1),8)*generateTriangle(64,8)



Konvertierung einer Dezimalzahl in einen Hex-String

hex(x)
oder hex(x,n)

Konvertiert eine Dezimalzahl in einen String aus Hex-Digits. Die Anzahl der zu verwendenen Digits ist ein optionaler Parameter, der maximal 8 Digits zuläßt.

Beispiele

hex(13)	"D"
hex(19)	"13"
hex(12,4)	"000C"

Hinweise

Hex Konstanten - um hexadezimale Konstanten in Aktions-Zeilen und mathematischen Ausdrücken verwenden zu können, beginnen Sie mit der Zahl und enden mit „H“ (z.B. 12A9H).

Nicht druckbare Zeichen in String Konstanten - benutzen Sie \x und den zwei Zeichen Hex-Code für die Zeichen (z.B. „Das ist ESC: \x1B“).

Hochpass Filter

highpass(wave,fsamp,fcutoff)

Filtert einen Vektor (Kurvenform) mit Hilfe der Hochpass Frequenz Funktion. Dabei wird die Abtastfrequenz sowie das untere Ende des Frequenzbereiches des Hochpasses angegeben.

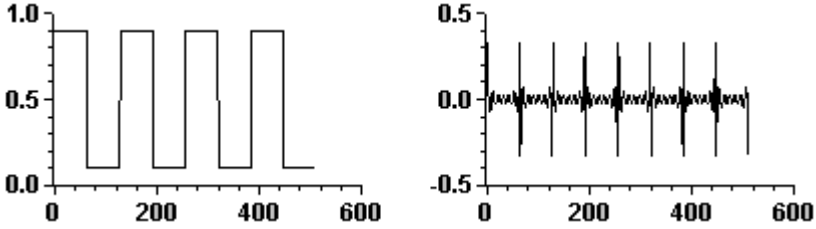
Der Filter Algorithmus ist:

1) Berechnung der FFT der Kurvenform und Ermittlung dessen Spektrums
2) Nullsetzung der Frequenzen unterhalb des Hochpasses
3) Berechnung der inversen FFT dieses Ergebnisses

siehe ebenfalls: bandpass(), highpass(), notch()

Beispiele

Anwendung der Funktion highpass(wave,1000,100) auf die links dargestellt Kurvenform ergibt die rechts abgebildete Kurvenform:



Erzeugt ein Histogramm der Verteilung aus dem Datensatz

histogram(vect,nintervals)

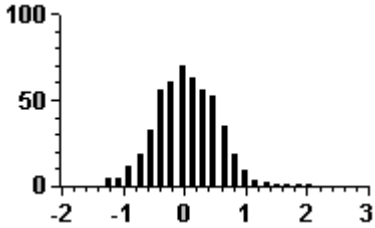
Nimmt die Maximal- und Minimalwerte aus dem gegebenen Vektorargument und teilt den Bereich auf in gleiche Intervalle. Anschließend werden die Datenelemente, die in die Bereiche hineinpassen, gezählt.

Das Ergebnis besteht aus einer Liste von zwei Vektoren:
Dem mittleren Wert für jedes Intervall und der Anzahl der Elemente, die in dieses Intervall hineinfallen.

Die graphische Darstellung kann direkt in einer TestPoint x-y-Balkengrafik erfolgen und ergibt eine Standard-Histogram Anzeige.

Beispiele

histogram(rndNormal(0,0.5,500),20)



Um ein Histogramm zu erzeugen, das einen festen Bereich A und B mit N Intervallen abdeckt, auch, wenn die aktuellen Daten nicht in dieses Inter-vall passen, benutzen Sie die **ramp** Funktion. Dadurch fällt ein Punkt auf jedes Intervall. Subtrahieren Sie eins vom Ergebnis mit der folgenden Formel:

histogram(appendvector(input,ramp(N)*(B-A+1)/N+A),N)-list(0,1)

Identitäts Matrix

idn(n,n)

Erzeugt eine Indetitätsmatrix (diagonale Elemente sind 1), der gegebenen Größe. Die beiden Argumente geben die Anzahl der Zeilen und Spalten an und müssen gleich groß sein.

Beispiel

idn(3,3)

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Bedingung (Auswahl) Ausdruck

if(Bedingung,wennwahr,wennfalsch)

Wenn das erste Argument (Bedingung) wahr (ungleich 0) ist, wird das zweite Argument (wennwahr) zurückgegeben, ansonsten das dritte (wennfalsch).

Beispiele

if(2<3,"wahr","falsch")	"true"
if(x<4,x*2,x-1)	mit x=vector(1,2,3), ergibt vector(2,4,6)

Bitte beachten Sie: Es werden in jedem Fall beide Ausdrücke (wennwahr, wennfalsch) ausgerechnet, auch wenn nur eine Alternative gewählt wird.

Inverse Schnelle Fourier Transformation

IFFT(fftlist)
oder IFFT(Amplitude,Phase)

Berechnet die inverse FFT und gibt einen Vektor mit reellen Zahlen zurück. Das Argument kann eine Liste von Vektoren sein, die in dem selben Format aufgebaut ist, wie das Ergebnis der **FFT** Funktion oder zwei separate Vektoren für die Amplitude und die Phase.

Infinite Impulse Response Filter

IIRfilter(daten,alpha,beta,fCutoff,Abtastfrequenz)

Filtert die gegebene Kurvenform (Vektor aus Zahlen) unter Benutzung der Infiniten Sprung-Antwort-Funktion. Das Filter ist spezifiziert durch alpha und beta S-Flächen-Koeffizienten für N kaskadierte direkte Form II Filter Sektionen. Die Abtastfrequenz und die gewünschte Grenzfrequenz müssen angegeben werden.

Alpha und **beta** müssen N mal 3 Felder sein. Die N Zeilen korrespondieren mit den N Filterstufen, die gewünscht werden. Die 3 Spalten sind die 0-te, 1-te und 2-te Potenz S-Flächen-Koeffizienten des Filters zweiter Ordnung. Die Koeffizienten Werte für die verschiedensten Filter finden Sie in Fachbüchern zur digitalen Filterentwicklung.

Als erstes werden die S-Flächen-Koeffizienten in Z-Flächen-Koeffizienten transformiert. Dann wird die Z Bereichstransformation berechnet.

Die TestPoint FFT-basierenden Filterfunktionen: **lowpass, highpass, bandpass,** und **notch** sind genererell bessere Filterfunktionen, außer Sie benötigen eine spezielle IIR-Filter Implementation.

Vektor oder Feld Indizierung (Element Zugriff)

index(vect,i)
or index(arr,i,j)

Gibt einen einzigen Wert aus einem Vektor oder Feld zurück, der definiert ist durch die Position i im Vektor bzw. i,j im Feld, beginnend bei der Position 0.

index(vect,i) ist ein Äquivalent zu **vect[i]**, mit der Ausnahme, daß die erste Funktion auch mit Vektor Werten berechnet werden kann, die zweite Funktion kann nur mit Variablen arbeiten.

Beispiele

index(vector(0,1,2,3),2)	2
index(ramp(8),3)	3

Suche einen Teil-String

instr(großerstring,kleinerstring)

Findet die Position des ersten Auftretens des Parameters kleinerstring in großerstring. Der Rückgabewert enthält die Position der ersten Übereinstimmung in großerstring. Wenn keine Übereinstimmung vorhanden ist, wird Null zurückgegeben.

Beispiele

instr("ABCDE","x")	0
instr("the red hen is red","red")	5

Integer (Rundung)

int(x)

Rundet das Argument. Für positives Argument wird immer ab-, für negatives Argument aufgerundet.

Die Funktion kann angewendet werden auf Zahlen, Vektoren oder Listen. Der Rückgabewert ist des gleichen Typs wie das Argument.

Beispiele

int(4.2)	4
int(3.8)	3
int(0)	0
int(-2.2)	-2
int(vector(1.2,-2.3))	vector(1,-2)

Numerische Integration

integrate(yvect,dx)
oder integrate(yvect,xvect)

Führt eine numerische Integration nach der Simpson Regel durch. Die Argumente können aus einem y-Vektor und einem x-Intervall oder aus Vektoren für y und x, für ungleiche x-Intervalle bestehen.

Das Ergebnis ist ein Vektor des Integrals zu jedem Punkt der Kurve. Das erste Element ist Null. Das zweite enthält die Fläche unter der Kurve von x[0] bis x[1]. Das nächste Element enthält die Fläche unter der Kurve von x[0] bis x[2], usw.

Wenn nur eine einzige Zahl als Ergebnis für die Fläche unter der Kurve erforderlich ist, benutzen Sie die **index**-Funktion, um den letzten Wert des Feldes zu erhalten:

```
index(integrate(yvect,dx),dim(yvect)-1)
```

Punktweise Verbindung mehrerer Vektoren

interleave (v1, v2, ...)

Erzeugt aus mehreren Vektoren einen einzigen, unter abwechselnder Verwendung der einzelnen Elemente aus den vorgegebenen Vektoren. Damit wird das erste Element des Ergebnisvektors v1[0], das nächste ist v2[0] bis zu vn[0]. Danach geht es weiter mit v1[1], v2[1] usw. Eine beliebige Anzahl von Vektoren kann an diese Funktion übergeben werden.

Es können ebenfalls Listen, bestehend aus einem oder mehreren Vektoren übergeben werden. Diese werden wie normale Vektoren behandelt.

Die Länge des Ergebnisvektors wird vom kürzesten Vektor Argument bestimmt. Dadurch wird die Länge des Ergebnisvektors zu n mal dem Minimum von dim(v1), dim(v2),...dim(vn).

Diese Funktion ist nützlich bei der Kombinierung für Vektorvorgaben für mehrkanalige D/A Ausgaben, da D/A Boards die Ausgabewerte in dieser Reihenfolge erwarten (Kanal 0 erster Wert, Kanal 1 erster Wert, usw. ...).

Beispiel

```
interleave(vector(1,2), vector(3,4)        vector(1,3,2,4)
```

Lineare Interpolation (Vergleichstabelle)

interpolate(x,xytabelle)

Berechnet einen y-Wert aus einem gegebenen x-Wert und einer Tabelle, die aus xy-Wertepaaren besteht. Die lineare Interpolation wird benutzt, um Werte zu bestimmen, die zwischen zwei Tabelleneinträgen liegen.

Diese Funktion kann auf Zahlen, Vektoren, Felder und Listen angewendet werden. Das zurückgegebene Ergebnis ist des gleichen Typs und der gleichen Dimension wie das x-Argument.

Die xy-Tabelle kann mit jeder gewünschten Art und Weise erzeugt werden, aber eine der allgemeinsten Methoden ist das Einlesen einer Tabelle aus einer Datei, unter Benutzung des TestPoint File-Objektes. Dadurch sind Sie in der Lage verschiedene Tabellen einfach auszuwählen und die Tabellen einfach zu verändern oder anzupassen.

Beispiele

interpolate(3.5,tabelle) mit tabelle: 0 0
1 5
2 7
4
12

=(list(vector(0,1,2,4),vector(0,5,7,12))

Ergebnis: 10.75

Matrix Umkehrung

inverse(x)

Kehrt die gegebene Matrix um. Das Argument muß ein quadratisches Zahlenfeld sein.

Die Funktion benutzt die LU-Faktorisierung der Matrix, um die Umkehrung zu berechnen. Der Algorithmus ist von der Standard Linpack Methode hergeleitet.

Beispiele

inverse(x)	mit x:	1 0 1 2 2 0 0 0 3		
	ergibt	1	0	-0.3333
		-1	0.5	0.3333
		0	0	0.3333

Länge eines Strings, Vektors oder Liste

length(x)

Wenn x ein **String** ist, ist das Ergebnis die String-Länge.

Wenn x ein **Vektor** ist, stellt das Ergebnis die Anzahl der Elemente dar (ebenso wie **dim**)

Wenn x eine Liste ist, besteht das Ergebnis aus der Anzahl der Positionen in der Liste.

Beispiele

length("ABCD")	4
length(ramp(8))	8
length(list(0,1,"x"))	3

Erzeugt eine Liste oder hängt an Listen an

list(a,b,...)

Erzeugt eine Liste aus den gegebenen Punkten, die aus beliebigen Datentypen bestehen können. Wenn ein Argument eine Liste ist, werden alle Punkte der Liste an die Ergebnisliste angehängt. Es kann eine beliebige Anzahl von Argumenten übergeben werden, einschließlich Null (die leere Liste).

Beispiele

```
list(1,2,8)
list(1,4,"xyz",vector(3,4))
list(1,2,list(3,4))                    list(1,2,3,4)
```

Logarithmus

log(x) or ln(x)
log10(x)

Berechnet den natürlichen oder den dekadischen Logarithmus einer Zahl x. Synonyme sind **log** oder **ln**.

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Siehe ebenfalls **pow10()** und **exp()**.

Beispiele

log(e())	1
log(exp(ramp(3)))	vector(0,1,2)
log10(100)	2

Logische und Bit Funktionen

x and y **bitweises und**
x or y **bitweises oder**
not x **logische Negation oder Einerkomplement**
x xor y **bitweise exklusives oder**

Diese Funktionen können als logische Operationen mit wahr/falsch (0 oder 1) Argumenten benutzt werden, oder als Bit Operation.

Hinweis: Die **not** Operation erzeugt nur wahr/unwahr (0/1) und erzeugt kein bitweises Komplement. Wenn Sie die Bits eines Wertes invertieren wollen, benutzen Sie einen der folgenden Ausdrücke, abhängig von der Anzahl der Bits, die Sie wünschen:

```
x xor 0FFH
x xor 0FFFFH
x xor 0FFFFFFFFH
```

(Die Konstanten beginnen mit einer Null und enden mit einem H, um auf Hexadezimalwerte hinzuweisen. Ein exklusives oder invertiert die Bits des Originalwertes.)

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Beispiele

(3<4) and (5<6)	1
(3<4) or (7<3)	1
13 and 7	5

Konvertiert einen String zu Kleinbuchstaben

lowcase(string)

Wandelt die alphabetischen Zeichen in einem String in Kleinbuchstaben um.

Beispiel

lowcase("ABcdE")	"abcde"

Tiefpassfilter

lowpass(wave,fsamp,fcutoff)

Filtert einen Vektor (Kurvenform) mit Hilfe der Tiefpass Frequenz Funktion. Dabei wird die Abtastfrequenz sowie das untere Ende des Frequenzbereiches des Tiefpasses angegeben.

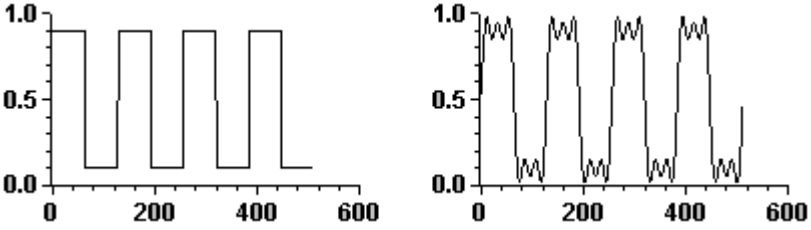
Der Filter Algorithmus ist:

1) Berechnung der FFT der Kurvenform und Ermittlung dessen Spektrums
2) Nullsetzung der Frequenzen oberhalb des Tiefpasses
3) Berechnung der inversen FFT dieses Ergebnisses

siehe ebenfalls: bandpass(), highpass(), notch()

Beispiele

Anwendung der Funktion lowpass(wave,1000,50) auf die links dargestellt Kurvenform ergibt die rechts abgebildete Kurvenform:



Matrizen Multiplikation

matMultiply(x,y)

Multipliziert zwei Felder. Die Anzahl der Spalten in x muß gleich der Anzahl der Zeilen in y sein. Ansonsten wird eine Fehlermeldung angezeigt.

Beispiele

matMultiply(x,y)	mit x:	1 2
		3 4
		5 6
	und y:	2 0
		1 1
	ergibt:	4 2
		10 4
		16 6

Maximalwert und Position

max(x)

Findet den Maximalwert in einem Vektor oder Feld.

Die Funktion kann ebenfalls auf Listen von Vektoren angewendet werden. Das Ergebnis besteht dann aus einer Liste von Maximalwerten.

maxindex(x)

Findet die Position des Maximalwertes in einem Feld. Wenn der gleiche Maximalwert mehrfach auftritt, wird das erste Vorkommen gefunden.

Die Funktion kann ebenfalls auf Listen von Vektoren angewendet werden. Das Ergebnis besteht dann aus einer Liste von Indizes.

Beispiele

max(vector(3,4,2,5,1,2))	5
maxindex(vector(3,4,2,5,1,2))	3
max(list(vector(4,3),vector(1,2,4,2))	list(4,4)

Medianwert eines Datensatzes (Vektor)

median(x)

Gibt den mittleren Wert eines gegebenen Vektors zurück. Der mittlere Wert wird gebildet, indem das Feld sortiert wird, und der Wert in der Mitte dieses sortierten Feldes als Ergebnis verwendet wird. Wenn der Vektor eine gerade Anzahl von Elementen hat, wird der Mittelwert (Average) der beiden zentralen Werte verwendet.

Die Funktion kann auch auf Listen von Vektoren angewendet werden. Das Ergebnis ist dann eine Liste der Medianwerte.

Beispiel

median(vector(2,4,3,3,4,5))	3.5

Minimalwert und Position

min(x)

Findet den Minimalwert in einem Vektor oder Feld.

Die Funktion kann ebenfalls auf Listen von Vektoren angewendet werden. Das Ergebnis besteht dann aus einer Liste von Minimalwerten.

minindex(x)

Findet die Position des Minimalwertes in einem Feld. Wenn der gleiche Minimalwert mehrfach auftritt, wird das erste Vorkommen gefunden.

Die Funktion kann ebenfalls auf Listen von Vektoren angewendet werden. Das Ergebnis besteht dann aus einer Liste von Indizes.

Beispiele

min(vector(3,4,2,5,1,2))	1
minindex(vector(3,4,2,5,1,2))	4
min(list(vector(4,3),vector(1,2,4,2))	list(3,1)

Mode eines Datensatzes

mode(v)

Findet den mode (häufigsten Wert) eines Vektors oder Feldes. Die Funktion kann ebenfalls auf Listen von Vektoren angewendet werden. Das Ergebnis besteht dann aus einer Liste von Modi.

Wenn ein Datensatz mehrere Modi hat (mehrere Werte mit der gleichen Häufigkeit), dann wird der kleinste Mode-Wert zurückgegeben.

Beispiele

mode(vector(1,2,3,2,4))	2
mode(list(vector(1,1,2),vector(4,3,4))	list(1,4)
mode(vector(1,2,3,2,3,4))	2
mode(vector(4,3,1,2))	1

Notch (Bandausschnitt) Filter

notch(wave,fsamp,flow,fhigh)

Filtert einen Vektor (Kurvenform) mit Hilfe der Bandausschnitt Frequenz Funktion. Dabei wird die Abtastfrequenz sowie das untere und obere Ende des Frequenzbereiches des Bandausschnittes angegeben.

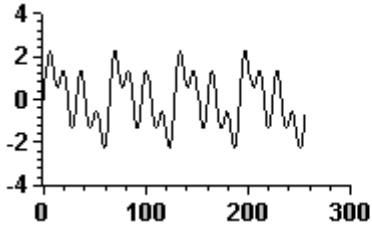
Der Filter Algorithmus ist:

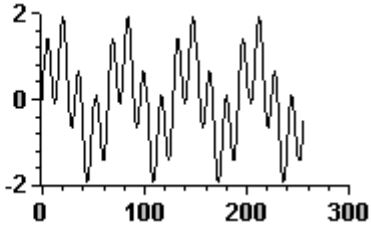
1) Berechnung der FFT der Kurvenform und Ermittlung dessen Spektrums
2) Nullsetzung der Frequenzen innerhalb des Notch-Filters
3) Berechnung der inversen FFT dieses Ergebnisses

siehe ebenfalls: bandpass(), highpass(), lowpass()

Beispiele

Anwendung der Funktion notch(wave,1000,20,40) auf die links dargestellt Kurvenform ergibt die rechts abgebildete Kurvenform:





Erzeugt einen Vektor oder ein Feld aus Einsen

one(n)
oder one(m,n)

Gibt einen Vektor oder ein Feld aus Einsen zurück. Die Größe des Vektors oder Feldes wird durch die Argumente vorgegeben. Wenn zwei Argumente angegeben sind, wird ein zweidimensionales Feld zurückgegeben.

Das berechnete Feld kann mit beliebigen Werten multipliziert werden, um Werte ungleich Eins zu erhalten.

Beispiele

```
one(3)                    vector(1,1,1)
one(5)*2                  vector(2,2,2,2,2)
one(2,3)                  Ergebnis ist:    1    1
                                           1    1
                                           1    1
```

pi (Mathematische Konstante)

pi()

Diese Funktion gibt den genauen Wert für Pi zurück.

Löst ein Polynom aus einem Vektor in Koeffizienten auf

polynomial(x,coef)

Löst ein Polynom aus einem gegebenen Vektor in ein Polynom auf. Der Koeffizientenvektor beginnt mit dem Konstanten Term in coef[0]:

```
coef[0] + coef[1] * x + coef[2] * x^2 + ...
```

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Diese Funktion kann in Verbindung mit der Funktion **fitPolynomial** verwendet werden. Nach der Anpassung der Polynomkurve an dieVersuchsdaten, kann die Funktion benutzt werden, um die Kurve mit verschiedenen x-Werten zu skalieren, beispielsweise zu Grafikzwecken. Sehen Sie sich hierzu das Beispiel FIT.TST an, das sich in dem Beispielverzeichnis (EXAMPLES) von TestPoint befindet.

Beispiele

polynomial(3,vector(1,2,3))	34
polynomial(ramp(3),vector(1,1,1))	vector(0,3,7)

Potenz von 10

pow10(x)

Erhebt 10 zur x-ten Potenz. Wenn x eine Ganzzahl ist, dann ist die Funktion genauer als 10^x.

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Beispiele

pow10(2)	100
pow10(ramp(4))	vector(1,10,100,1000)
pow10(list(3,-1))	list(1000,0.1)

Potenz von 2

pow2(x)

Erhebt 2 zur x-ten Potenz. Wenn x eine Ganzzahl ist, dann ist die Funktion genauer als 2^x.

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Beispiele

pow2(2)	4
pow2(ramp(4))	vector(1,2,4,8)
pow2(list(3,-1))	list(8,0.5)

Vorhergehender Wert dieser Math-Funktion

previous()

Diese Funktion gibt das Ergebnis des Math-Objektes zurück, bevor der Ausdruck berechnet wurde (und weist den neuen Wert dem Math-Objekt zu). Sie ist nützlich, wenn Zähler und iterative Berechnungen durchgeführt werden müssen.

Da das Math-Objekt mit dem Datentyp **Nodata** startet, wenn die Anwendung gestartet wird, muß das Math-Objekt zu Beginn, bevor die **previous()** Funktion aufgerufen wird, initialisiert werden. Das Math-Objekt kennt dazu die **Set**-Aktion.

Beispiele

Diese Formel stellt einen Zähler dar:

```
previous()+1
```

Um den Zähler zu initialisieren, benutzen Sie folgende Zeile

```
1) Set          Counter          to 0
```

Um den Zähler zu erhöhen, benutzen Sie folgende Zeile:

```
1) Calculate    Counter
```

Erzeugt einen Vektor mit ansteigenden Werten

ramp(n)
oder ramp(m,n)

Gibt einen Vektor oder ein Feld mit Ganzzahlen zurück, die bei 0 beginnen. Wenn zwei Argumente benutzt werden, wird ein Feld zurückgegeben. Die Zahlen steigen in Zeilen und Spalten.

Wenn die Rampenfunktion andere Werte zurückgeben soll, muß sie per Faktor und Offset skaliert werden.

Beispiele

```
ramp(4)                          vector(0,1,2,3)
ramp(2,3)                        Ergebnis:  0  1  2
                                            1  2  3
ramp(3)*2+1                      vector(1,3,5)
```

Gleichmäßig- oder Normalverteilte Zufallszahlen

rnd()
oder rnd(n)
oder rnd(m,n)

rndNormal()
oder rndNormal(mean)
oder rndNormal(mean,sigma)
oder rndNormal(mean,sigma,n)
oder rndNormal(mean,sigma,m,n)

Gibt eine oder mehrere Zufallszahlen zurück. Die **rnd()** Funktion ergibt gleichmäßig verteilte Zahlen zwischen 0 und 1. Die **rndNormal()** Funktion gibt normal-verteilte Werte, mit wählbarem Mittelwert und Standardabweichung, zurück.

Die Vorgabewerte für die **rndNormal()** Funktion sind für den Mittelwert 0 und für die Standardabweichung 1.

Beide Zufallsfunktionen können einen Vektor oder ein Feld der gegebenen Dimensionen erzeugen.

Beispiele

```
rnd()
rnd(5)
rndNormal()
rndNormal(0,1.5)
rndNormal(0,1,100)
```

Ändert die Dimensionen eines Feldes

reshapeArray (Feld, Zeilen)

Ändert die Anzahl der Zeilen in einem Feld und gibt ein Feld zurück, das die gleichen Werte beinhaltet, aber eine andere „Gestalt“ hat.

Wenn die Anzahl der Werte in einem Feld kein Vielfaches der Spalten ist, werden die übrigen gelöscht.

Beispiel

Ausgangspunkt ist ein Feld als Argument wie folgt:

1	2	3
4	5	6
7	8	9

Die Umformung in n Zeilen ergibt:

n=4

1	2
3	4
5	6
7	8

n=2

1	2	3	4
5	6	7	8

Kehrt einen Vektor um

reverse(x)

Kehrt die Reihenfolge der Elemente in einem Vektor um.

Die Funktion kann auch auf Listen von Vektoren angewendet werden und gibt dann ebenfalls Listen zurück.

Beispiele

```
reverse(vector(1,2,3,4))                    vector(4,3,2,1)
reverse(list(ramp(3),vector(4,2,5)))
                    list(vector(2,1,0),vector(5,2,4))
```

Effektivwert Berechnung (RMS)

rms(vect)

Gibt den Effektivwert (RMS=Root Mean Square) zurück. Dieser ist definiert als sqrt(avg(vect^2)).

Die Funktion kann ebenfalls auf Listen angewendet werden und hat dann als Rückgabewert eine Liste der Effektivwerte.

Rotiert die Elemente in einem Vektor

rotate(x,n)

Bewegt die Elemente in einem Vektor um n Plätze kreisförmig nach links. Werte vom Anfang des Vektors werden ans Ende bewegt.

Wenn n negativ ist, werden die Bewegungen nach rechts durchgeführt.

Die Funktion kann auch auf Listen von Vektoren angewendet werden, und gibt eine Liste von gedrehten Vektoren zurück.

Beispiele

rotate(vector(1,2,3,4,5),2)	vector(3,4,5,1,2)
rotate(vector(1,2,3,4,5),-1)	vector(5,1,2,3,4)

Rundet zur nächsten Ganzzahl

round(x)

Rundet das Argument zur nächsten Ganzzahl. Werte die in .5 enden, werden aufgerundet (Equivalent zu **floor(x+0.5)**).

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Beispiele

round(4.2)	4
round(3.8)	4
round(0)	0
round(-2.2)	-2
round(vector(1.2,-2.3))	vector(1,-2)

Wählt ein Element der Liste aus

select(list,i)

Gibt ein ausgewähltes Element der gegebenen Liste zurück. Die Numerierung der Elemente in der Liste beginnt bei 0.

Wenn der Index des Elementes i kleiner als 0 ist oder über das Ende der Liste hinausgeht, ist der Rückgabewert ein **Nodata** Wert. Wenn das Listenargument keine Liste repräsentiert, wird es als 1-Punkt Liste interpretiert.

Beispiele

select(list(0,1,2,3,4),3)	3
select(list("a","b","c"),1)	"b"
select(list(0,1,2,3,4),5)	**nodata**
select(list(ramp(3),2,"x"),0)	vector(0,1,2)

Vorzeichen (0,1 oder -1) einer Zahl

sgn(x)

Gibt 0,1 oder -1, basierend auf dem Wert des Arguments, zurück. Ist das Argument größer als 0, ist der Rückgabewert 1. Wenn es gleich Null ist, ändert sich der Wert nicht. Alle Argumente, die kleiner als Null sind, werden zu -1.

Anwendungshinweis:

Eine Anwendung dieser Funktion ist der Vergleich eines Vektors mit einer Zahl, um individuelle Vergleichsergebnisse für jedes Element zu erhalten. Die regulären Vergleichsoperatoren (größer als, kleiner als,...) prüfen, ob alle Elemente im Vektor dem Vergleich entsprechen.

Um einen Vektor zu erhalten, der anzeigt, ob jedes Element über oder unter 3 ist, benutzt man folgende Formel:

```
sgn(input-3)
```

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Beispiele

sgn(4)	1
sgn(-3)	-1
sgn(0)	0
sgn(vector(3,0,2,-2))	vector(1,0,1,-1)
sgn(vector(3,4,5,4,2)-3)	vector(0,1,1,1,-1)

Kurvenform Glättung (Filterung)

smoothAvg(wave,n)
smoothAvgCentered(wave,n)
smoothMedian(wave,n)

Diese Funktionen glätten eine Kurvenform durch die Verbindung der nächstliegenden Kurvenelemente. Der n Parameter kann zwischen 1 und 1000 liegen.

Die **SmoothAvg** Funktion ermittelt den Mittelwert aus dem aktuellen und der vorherigen n-1 Punkte. Der aktuelle Punkt wird durch das Ergebnis der Berechnung ersetzt.

Die **SmoothAvgCentered** Funktion mittelt n Punkte. Dabei wird der aktuelle Punkt als Mitte für diese Berechnung verwendet.

Die **SmoothMedian** Funktion berechnet den Median von n Punkten zentriert auf den aktuellen Punkt. Dies ist ein nicht-lineares Filter, das Spitzen eliminiert.

Alle drei Funktionen belassen einige Punkte am Ende des Vektors unverändert. Beispielsweise überspringt die SmoothAvg Funktion die ersten n-1 Punkte, bis genug (n) Punkte vorhanden sind, um die Mittelung vorzunehmen.

Diese Funktionen können ebenso auf Listen von Vektoren angewendet werden und geben dann eine Liste geglätteter Vektoren zurück.

Löst ein System linearer Gleichungen

solve(coef,rhs)

Löst eine n lineare Gleichungen mit n Unbekannten. Die **Coeff** Parameter müssen aus einem n mal n Feld bestehen. Der **Rhs** Parameter besteht aus einem Vektor mit n Werten. Die Gleichungen haben folgendes Aussehen:

```
coef[0,0] * x0 + coef[0,1] * x1 + ...     = rhs[0]
coef[1,0] * x0 + coef[1,1] * x1 + ...     = rhs[1]
coef[2,0] * x0 + coef[2,1] * x1 + ...     = rhs[2]
usw...
```

Ein Vektor mit Lösungswerten wird zurückgegeben.

Beispiel

solve(coef,rhs) mit coef:

```
1   3
2   2
```

und rhs = vector(11,10),
Repräsentieren die Gleichungen:

```
x + 3y = 11
2x + 2y = 10
```

ergibt = vector(2,3), also x=2 und y=3!

Sortiert einen Vektor oder eine Liste von Vektoren

sort(x)
oder sort(list,field)

Sortiert einen gegebenen Vektor in aufsteigender Ordnung. Der Vektor kann Zahlen oder Zeichenketten enthalten.

Bei einer Liste von Vektoren kann durch einen optionalen Field-Parameter (Listenpunkt Nummer), die zur Sortierung zu verwendende Liste ausgewählt werden. Alle Vektoren in der Liste werden basierend auf dem ausgewählten Vektor sortiert.

Beispiele

```
sort(vector(1,4,3,2))
    ergibt:          vector(1,2,3,4)
sort(list(vector(3,5,2),vector("a","c","b")))
    ergibt:          list(vector(2,3,5),vector("b","a","c"))
sort(list(vector(3,5,2),vector("a","c","b")),1)
    ergibt:          list(vector(3,2,5),vector("a","b","c"))
```

Quadratwurzel

sqr(x)

Gibt die Quadratwurzel des Arguments zurück.

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Beispiele

sqr(16)	4
sqr(vector(25,9,16))	vector(5,3,4)

Standardabweichung

stddev(v)

Gibt die Standardabweichung der Werte des gegebenen Vektors zurück.

Die Funktion kann auch auf Listen von Vektoren angewendet werden und gibt dann eine Liste von Standardabweichungen zurück.

Die benutzte Formel ist:

sqr(sum(v[i] - avg(v))^2) / n

Formatiert eine Zahl als String

str(x)
oder str(x,fmt)

Wandelt die gegebene Zahl in einen String um. Ein optionaler Argumentenparameter kann angegeben werden.

Das Format Argument **fmt** ist das gleiche, das in der Programmiersprache „C“ im Zusammenhang mit der „printf“ Funktion verwendet wird. Es beginnt immer mit einem Prozentzeichen, gefolgt von einem Minuszeichen für die Linkszentrierung, dann die optionale Feldbreite, einem optionalen Dezimalpunkt mit Angabe der Nachkommastellen und einem „l“ für „Long“-Werte. Die folgenden Parameter spezifizieren die Darstellungsweise:

x oder X	Hex-Format
f	Floating Point
e oder E	Wissenschaftliche (Exponentielle) Darstellung
g oder G	f oder e Format, automatisch, basierend auf dem Wert.

Der voreingestellt Wert ist „%.15G“. Dieser Wert wird normalerweise für die TestPoint Formatierung benutzt.

Beispiele

str(3.45)	"3.45"
str(34.56,"%d")	"34"
str(34.56,"%8.2f")	" 34.56"
str(34.56,"%-8.2f")	"34.56 "
str(34.56,"%-0.2f")	"34.56"
str(34.56,"%E")	" 3.456000E+01"

str(3.45)	"3.45"
str(34.56,"%d")	"34"
str(34.56,"%8.2f")	" 34.56"
str(34.56,"%-8.2f")	"34.56 "
str(34.56,"%-0.2f")	"34.56"
str(34.56,"%E")	" 3.456000E+01"

Vergleicht zwei Strings

strCompare(s1,s2)

Vergleicht zwei Strings. Der Rückgabewert ist -1, wenn s1 kleiner als s2 ist, Null, wenn beide Strings gleich sind und +1, wenn s1 größer ist als s2.

Überprüft zwei Strings auf Gleichheit

strEqual(s1,s2)

Gibt 1 (wahr) zurück, wenn die Strings gleich sind und Null, wenn nicht.

Entfernen von führenden oder abschließenden Leerzeichen

strtrim(str)
oder strtrim(str,chars)

Entfernt führende oder abschließende Leerzeichen, Returns, Linefeeds oder Tabs aus einem String. Ein optionaler Parameter kann angegeben werden, um andere Zeichen als die Voreingestellten zu entfernen.

Beispiele

strtrim(" abc\r\n")	"abc"
strtrim("aabbacdcaa","ac")	"bbacd"

Wählt einen Teil aus einem Vektor oder Feld aus

subarray(v,start)
oder subarray(v,start,end)
oder subarray(a,startrow,endrow,startcol)
oder subarray(a,startrow,endrow,startcol,endcol)

Extrahiert einen Teil aus einem Vektor oder Feld, der definiert wird durch einen Bereich von Index-Werten. Wenn ein optionaler Parameter wie Endindex nicht angegeben wird, wird ein voreingestellter Wert benutzt. Der voreingestellte Wert ist das Ende des Vektors oder Feldes.

Die Funktion kann auch auf Listen von Vektoren oder Felder angewendet werden, wobei das Ergebnis dann Listen von Sub-Feldern ist.

Beispiele

subarray(vector(0,1,2,3,4,5),2,4)	vector(2,3,4)
subarray(vector(0,1,2,3,4,5),3)	vector(3,4,5)
subarray(vector(0,1,2,3,4,5),-1,2)	vector(0,1,2)
subarray(vector(0,1,2,3,4,5),6,8)	vector()

Auswahl eines Teils aus einer Liste

sublist(list,start)
oder sublist(list,start,end)

Extrahiert einen Teil einer gegebenen Liste zwischen spezifizierten Listenelementindexes. Das erste Element der Liste hat die Nummer 0.

Beispiele

sublist(list(0,1,2,3,4),2,3)	list(2,3)
sublist(list("a",2,ramp(3),1),1)	list(2,ramp(3),1)

Auswahl eines Teilstrings

substr(str,start)
oder substr(str,start,end)

Extrahiert einen Teil des gegebenen Strings zwischen den gegebenen Start- und Endpositionen. Die Position des ersten Zeichens in dem String ist 1.

Beispiele

substr("abcdef",4)	"def"
substr("abcdef",2,3)	"bc"
substr("abcdef",7)	""

Summiert die Werte eines Vektors oder Feldes auf

sum(x)

Gibt die Summe der Werte in einem gegebenen Feld oder Vektor zurück.

Die Funktion kann auch auf Listen von Vektoren angewendet werden und gibt dann eine Liste von Summen zurück.

Beispiele

sum(vector(1,3,4))	8
sum(list(ramp(3),vector(5,2)))	list(3,7)

Matrizen umstellen

transpose(x)

Stellt das gegebene Zahlenfeld um, indem Zeilen und Spalten vertauscht werden.

Beispiele

transpose(x)	mit x:	1 2 3 4 5 6
	ergibt:	1 4 2 5 3 6

Trigonometrische und Hyperbolische Funktionen

sin(x)	**Sinus**
cos(x)	**Cosinus**
tan(x)	**Tangens**
asin(x)	**Arcus Sinus**
acos(x)	**Arcus Cosinus**
atan(x)	**Arc Tangens**
atan2(y,x)	**Arc Tangens, gegebener y und x Komponenten**
sinh(x)	**Hyperbolic Sinus**
cosh(x)	**Hyperbolic Cosinus**
tanh(x)	**Hyperbolic Tangens**

Die Familie der trigonometrischen Funktionen mit Argumenten im Bogenmaß.

Die **Atan2** Funktion ist nützlich, wenn die x und y Anteile bekannt sind und x Null sein könnte, sodaß die Division y durch x nicht sicher ist.

Diese Funktion kann auf Zahlen, Vektoren, Felder oder Listen angewendet werden. Das Ergebnis ist des selben Typs und der selben Dimension wie das Argument.

Beispiele

sin(pi()/2)	1
atan2(1,0)	pi/2

Datentyp ermitteln

type(x)

Gibt eine Zahl zurück, die den Datenstyp des Arguments spezifiziert:

0	keine Daten (Uninitialisiert oder Leer)
1	Zahl
2	String
3	Vektor
4	Feld
5	Liste

Konvertiert einen String in Großbuchstaben

upcase(str)

Wandelt alle alphabetischen Zeichen in Großbuchstaben um.

Beispiele

upcase("TestPoint")	"TESTPOINT"

Konvertiert einen binären String in eine Zahl

valbin(str)

(Maximal 32 Bits, führende Leerzeichen werden übersprungen)

Beispiele

valbin("101")	5

Konvertiert einen Hex-String in eine Zahl

valhex(str)

(Maximal 8 Digits, führende Leerzeichen werden übersprungen)

Beispiele

valhex("C")	12
valhex("13")	19

Anhängen von Vektoren oder Listen von Vektoren

vector(x,y,...)

Kombiniert Null oder mehr Vektoren in einen längeren Vektor (Synonym für **AppendVector()**). Werden keine Argumente gegeben, ist das Ergebnis ein Null-Längen Vektor. Jede beliebige Anzahl von Vektoren oder Skalaren können mit dieser Funktion zusammengefaßt werden. Die Funktion ist ebenfalls nützlich für die Erzeugung von kurzen konstanten Vektoren (siehe Beispiele unten).

Diese Funktion kann ebenfalls Listen mit mehreren Vektoren an andere Listen anhängen. Wenn ein Argument eine Liste von Vektoren ist, müssen alle anderen Argumente ebenfalls Listen von Vektoren sein. Die Listen werden Punkt für Punkt zusammengefügt und brauchen nicht die gleiche Anzahl von Punkten zu enthalten.

Beispiele

vector(x,y)	mit x=vector(1,2) und y=vector(3,4), Ergebnis ist vector(1,2,3,4)
vector(1,2,3,4)	vector(1,2,3,4)
vector(l1,l2)	mit l1=list(vector(1,2),vector(10,20) und l2=list(vector(3,4),vector(8)) Ergebnis:

list(vector(1,2,3,4),vector(10,20,8))

Modifiziert einen Vektor durch Ersetzen eines Elements

vectorReplace(vect,i,newvalue)

Ersetzt ein Element in einem gegebenen Vektor. Ist der angegebene Index „i“ größer als die Länge des Vektors, wird der Vektor automatisch verlängert.

Bitte beachten Sie, daß das Math Objekt das modifzierte Array enthält, sobald diese Aktion ausgeführt wurde. Das originale, in das Parameterfeld ‘array’ eingesetzte, Array wird **nicht** modifiziert.

Eine Möglichkeit, mit nur einem Array vor und nach der Aktion zu operieren, stellt die Verwendung des **previous** Befehls dar. Hier wird das Ursprungsarray, das im Datenwert des Math Objektes gespeichert sein muß, auch wieder modifziert im Math Objekt gespeichert:

```
vectorReplace( previous(), i, newvalue)
```

Fenster Funktionen zum Gebrauch vor der FFT

Multipliziert jedes Element des Argumentes mit einem Gewichtungsfaktor. Mit einem gegebenen Vektor der Größe N läuft der Parameter k der folgenden Gleichungen von -N/2 bis N/2.

hamming(x)

0.543748+0.456252*cos(2*pi*k/N)

hanning(x)

0.5+0.5*cos(2*pi*k/N)

blackman(x)

0.426590+0.496560*cos(2*pi*k/N)+0.076848*cos(4*pi*k/N)

blackmanharris3(x)

0.42323-0.49755*cos(2*pi*k/N)+0.07922*cos(4*pi*k/N)

blackmanharris4(x)

0.35875-0.48829*cos(2*pi*k/N)+0.14128*cos(4*pi*k/N)
-0.01168*cos(6*pi*k/N)

poisson(x,alpha)

exp(-alpha * abs(k) * 2 / N)

Erzeugt einen Vektor oder ein Feld aus Nullen

zero(n)
oder zero(m,n)

Gibt einen Vektor oder ein Feld mit Nullen zurück. Die Größe des Vektors oder des Feldes wird durch die Argumente bestimmt. Wenn zwei Argumente gegeben sind, wird ein 2-dimensionales Feld erzeugt.

Beispiele

zero(3)		vector(0,0,0)
zero(2,3)	Ergebnis:	0 0
		0 0
		0 0

Anhang A
TESTPT.INI Datei Format

TESTPT.INI beinhaltet Informationen zur Konfiguration des TestPoint Editors und der Run-Time Versionen. Sie besteht aus Sektionen, die mit „[Sektionsnamen]“ beginnen und in der Form „Einstellung=Wert“ definiert sind.

[GPIBn]

Hardware Konfigurationseinstellungen für GPIB. Siehe auch Hardware Konfigurationsanhang.

[ADn], [DAn]

Hardware Konfigurationseinstellungen für A/D und D/A Karten. **Hinweis:** Für kombinierte A/D und D/A Karten ist es **nicht** erforderlich, diese Einstellungen zu duplizieren.

[ADDRIVERS]

TestPoint Treiber Bibliotheken für A/D-Karten. Sind im Lieferumfang von TestPoint enthalten.

[DIOn]

Hardware Konfigurationseinstellungen für Digitale Ein-/Ausgaben. Siehe auch Hardware Konfigurationsanhang.

[Config]

GraphBuffer=yes Benutzung des Bilschirmbuffers zum sanften Grafikaufbau.

MaxPendingEvents=32 Anzahl der Ereignisse, die auf jedes Objekt warten kann. Ein Ereignis wartet, wenn das Ereignis eintritt, aber die Aktionsliste wird bereits weiter ausgeführt. Wenn die Aktionliste beendet ist und ein weiteres Ereignis wartet, dann wird die Aktionsliste wieder ausgeführt. Wenn das hier angegebene Limit überschritten wird, wird eine Fehlermeldung ausgegeben. Der Wert für diese Einstellung kann jede Ganzzahl zeischen 0 und 30 000 sein.

UIMaxPendingEvents=2 Die selbe wie MaxPendingEvents, aber bezieht sich auf User-interface Objekte wie Pushbuttons, Data-entry Felder, Schalter, etc. Dieses Limit dient normalerweise dazu, Probleme zu umgehen, wenn ein Benutzer Klick auf ein Objekt öfter durchgeführt wird, während andere längere Operationen noch ausgeführt werden müssen.

Stack=64 Das Maximum der Action List Stack Tiefe, bevor ein Fehler angezeigt wird. Dieser Stack wird dazu verwendet, wenn eine Action List eine andere ausführen läßt - beispielsweise, indem ein Knopf einen Schalter schaltet, dessen Action List dann ausgeführt wird (Stack Tiefe 2). Kann von 16 bis 512 gesetzt werden.

MaxObjectWindows=2 Maximale Anzahl von Objekt Detail Fenstern (mit Tabs für die Action List, Settings, etc.). Enthält

nicht die Settings Fenster, die temporär offen sind. Sobald mehr Fenster vernwendet werden, wird das älteste wieder geschlossen. Kann auf einen Wert zwischen 1 und 32 gesetzt werden.

DragPauseDistance=8 Maximale Distanz, die die Maus sich bewegen darf, und für TestPoint immer noch auf einem Platz stehen bleibt für Drag-and-drop. Sobald die Maus auf einem Fensterteil steht, das nicht im Vordergrund ist, wird das Fenster nach vorne gesetzt, um die Vollendung der Drag-and-drop Operation zu gewährleisten. Kann von 4 bis 50 gesetzt werden.

DragPause=1000 Zeit in Millisekunden, die die Maus stationär auf einem Platz gehalten werden muß, um als pausierend zu gelten. (siehe oben). Kann von 500 bis 10000 gesetzt werden.

DragScroll=100 Zeit in Millisekunden für die Auto-scroll Wiederholungsrate.

nobackups=Falls dieses Setting Yes ist, werden keine .BAK Files angelegt, sobald ein TestPoint Applikationsfile gespeichert wird. Falls dieses Setting No, oder nicht vorhanden ist, werden Backup Files angelegt.

[Intl]

sDecimal=. Dezimalseparator für Zahlen, für internationalen Einsatz. Nur für den Einsatz des Clipboard oder durch DDE. Kann **.** oder **,** oder **windows** sein. Falls sie **windows** ist, werden die internationalen Settings im Windows Control Panel verwendet (dies kann nützlich für den Datenaustausch mit Microsoft Excel sein).

[Print.SkipSettings]

Liste der Objekt Settings, die während des Ausdrucks ausgelassen werden, wenn das **some** Setting im Print Dialog gewählt wurde.

[Keys]

The "Keys", oder Passwörter, die verwendet wurden, um User-defined Objekte oder Passwörter zu speichern. Diese Section wird automatisch von TestPoint verwaltet. Die Keys werden zur eigenen Bequemlichkeit hier gespeichert, so muß man nicht immer die eigenen Passwörter verwalten. Sobald eine Runtime Diskette erstellt wird, werden die Keys **nicht** hinzugefügt.

[Workspace]

Information über das Layout der Fenster in TestPoint und anderer Settings, wie beispiesweise Snap to grid, so daß diese Settings bei der nächsten Sitzung von TestPoint verwendet werden können. Diese Werte werden immer gespeichert, wenn TestPoint verlassen wird.

[Math]

Listet Add-on mathematische Funktionspakete auf. TestPoint erlaubt, daß die eingebauten mathematischen Funktionen mit weiteren Add-ons erweitert werden. Diese Add-ons müssen in einem eigenen File Format vorliegen. Damit ein Add-on erkannt wird, muß sein Filename (package.DLL=) hier aufgelistet sein. Diese Sektion sollte normalerweise nicht verändert werden.

[Objects]

Listet Add-on Objektpakete auf. TestPoint erlaubt es, daß neue Objekt Kategorien in Form von separaten Support Files hinzugefügt werden. Der interne Codename muß in folgender Weise aufgelistet werden: interner Codename gefolgt von einem Gleichheitszeichen und dem Support File Namen. Diese Add-ons

müssen in einem speziellen File Format vorliegen. Diese Sektion sollte normalerweise nicht modifiziert werden.

Anhang B.
Demo- oder simulierte Daten

TestPoint kann simulierte Daten anstatt von den Daten aktuell vorliegender Hardware benutzen, wie beispielsweise Daten des GPIB und DIO Objekts. Dieses Feature kann während der Lernphase von TestPoint, für den Einsatz in der Debugging Phase, oder falls ein bestimmter Teil des Test Equipments temporär nicht vorhanden ist.

Objekte, die Demodaten verwenden können, besitzen ein Setting, das kontrolliert, ob Demodaten oder echte Hardwaredaten verwendet werden sollen.

Standardmäßig sind die Demo Daten immer ein Wert, die zufällig verteilt um den Wert 3 herum liegt.

Jedes Objekt kann jedoch eigene simulierte Daten verwenden, indem die gewünschten Daten in einer Datei namens DEMO.DAT im TestPoint Verzeichnis abgelegt werden. Diese Datei enthält Zeilen, wie folgende:

```
#GPIB:Meter Enter from NUMBER 4.25
#GPIB:Meter2 Enter from STRING "this is a string"
#GPIB:Scope Enter from VECTOR OF NUMBER 8
1
2
3
4
5
6
7
8
#RS232:Remote Enter from LIST 3
NUMBER 3.5
STRING "a string"
VECTOR OF NUMBER 3
10
20
30
```

Das "#" Zeichen zeigt einen Demodaten Eintrag an. Er wird gefolgt von dem Class Namen des Objekts, einem Doppelpunkt (:), dann dem Objektnamen, einem Leerzeichen, und dem Action Namen. Groß- und Kleinschreibung sind dabei überall wichtig. Nach dem Action Namen kommt der Datentyp, der zurückgegeben werden soll, der NUMBER, STRING, VECTOR OF NUMBER, etc sein darf. Für Vektoren muß die Vektorlänge von dem Typnamen gefolgt sein.
Als nächstes sind die Daten enthalten. Für Zahlen und Strings wird auf der selben Zeile weitergeschrieben. Vektoren müssen in separate Zeilen unterteilt werden, ein Wert pro Zeile.
In einer LIST muß die Anzahl der Items gegeben sein, und dann die verschiedenen Datentypen in separaten Zeilen.

Falls die selbe Action unterschiedliche Datenwerte auf verschiedene aufeinanderfolgende Ausführungen liefern soll, wird eine Sequenznummer in dem Demodatenfile angegeben, wie beispielsweise:

```
#GPIB:Meter Enter from(0) NUMBER 3.5
#GPIB:Meter Enter from(1) NUMBER 3.6
#GPIB:Meter Enter from(2) NUMBER 3.7
#GPIB:Meter Enter from(*) NUMBER 4.0
```

Eine Sequenznummer von * zeigt an, daß ein Wert für alle Ausführungen nach der explizit gegebenen verwendet wird.

Bitte beachten Sie: Das A/D Objekt ist eine Ausnahme: seine Demodaten, wenn sie gesmaplet werden, entsprechen einem 1KHz Rechtecksignal auf Kanal 0, und einem 1KHz Sägezahnsignal auf den anderen Kanälen. Die DEMO.DAT Datei wird nicht für A/D benutzt.

Anhang C.
Hardware Konfiguration

Die Hardware Konfigurationseinstellungen, die für TestPoint verwendet werden, sind in der TESTPT.INI Datei angegeben. Diese Datei kann mit ASCII Text Editor bearbeitet werden, wie z.B. das Windows Notepad Programm.

GPIB

TestPoint arbeitet mit jeder CEC GPIB (IEEE-488) Karte. Installieren Sie die Karte, wie es im Handbuch, das mit der Karte geliefert wird angegeben ist.

TestPoint unterstützt auch GPIB Karten anderer Hersteller, mit optionalen Zusatztreibern (kostenpflichtig). Zu diesen Treibern werden Instruktionen für die Installation und Konfiguration geliefert.

Die Hardware Einstellungen für die entsprechende GPIB Karte muß in der TESTPT.INI Datei eingegeben werden.

[GPIB0]

Der [GPIBn] Abschnitt enthält Hardwareeinstellungen für dir Karte n. Es können bis zu 4 Karten, numeriert von 0 bis 3, verwendet werden.

myaddr=21 Setzt die GPIB Adresse, die für die Controller Karte benutzt wird. Sie muß sich von den anderen Adressen der verwendeten Geräte unterscheiden und kann zwischen 0 und 30 liegen. Voreinstellung, auch wenn kein Eintrag vorhanden ist, ist 21.

io=0x2B8 Die I/O Adresse der GPIB Karte. Das Prefix „0x“ bedeutet, daß die Angabe Hexadezimal ist. Voreinstellung, auch wenn kein Eintrag vorhanden ist, ist 0x2B8.

irq=5 Der Interrupt Level der GPIB Karte. Er kann negartv sein (-1), wenn kein Interrupt verwendet wird, kann aber auch ausgelassen werden. Interrupts sind nicht erforderlich.

dma=1 Der DMA Kanal für die GPIB Karte. Kann negativ sein (-1), wenn kein DMA verwendet wird, kann aber auch ausgelassen werden. Voreinstellung ist kein DMA.

A/D,D/A

TestPoint arbeitet mit einer Vielzahl von verschiedenen A/D und D/A Interface Karten zusammen. Einige der Settings sind herstellerspezifisch. Generell erfordern alle A/D Karten einen Interrupt für zeitlich korrekte Datenaufnahme, und einen DMA Kanal für eine hohe Datenübertragungsrate.

WICHTIGE INFORMATION: Bitte lesen Sie die A/D Setup Datei, die als Icon in der TestPoint Gruppe beigefügt ist, um weitere detaillierte Informationen zu bekommen.

[ADn] oder [DAn]

Diese Sektion bietet Hardware Settings für A/D oder D/A Board n. Es können bis zu 4 A/D und/oder D/A Boards, nummeriert von 0 bis 3, verwendet werden. Falls ein kombiniertes A/D und D/A Board verwendet wird, müssen die Settings nicht dupliziert werden.

manufacturer=AUTO Der Manufacturer des Boards. Dieser String muß mit einem der eingebauten Strings in der ADDRIVERS Sektion übereinstimmen. Die Standardeinstellung ist "AUTO", die das automatische Boardsuchverfahren benutzt. Einige Hersteller unterstützen dieses Feature, beispielsweise Keithley, Data Translation, und NI. Andere Boards unterstützen es nicht, dann muß das TESTPT.INI File manuell editiert werden.

DIO

TestPoint arbeitet mit jedem digitalem I/O Board, das den Standardbaustein Intel 8255 Digital I/O Controller Chip benutzt. Die meisten DIO Cards verwenden diesen Chip.

Ein DIO Board kann bis zu 10 digitale I/O Chips auf dem Board enthalten, damit ist ein Maximum von 240 DIO Bits möglich. TestPoint unterstützt ein Maximum von 4 DIO Boards.

Die Hardware Settings des DIO Boards müssen in der TESTPT.INI datei eingetragen sein.

[DIO0]

Die [DIOn] Sektion bietet Hardware Settings für DIO Board n. Es ist ein Maximum von 4 Boards möglich, nummeriert von 0 bis 3.

io=0x300 Die Basis I/O Addresse für das DIO Board. Das "0x" bedeutet hexadezimal. Erforderlich.

channels=1 Die Anzahl der DIO Chips auf dem Board. Standardeinstellung ist 1, falls nicht spezifiziert.

iostep=4 Das I/O Addressinkrement zwischen aufeinanderfolgenden DIO Chips, falls mehrere DIO Chips auf dem Board verwendet werden. Default Einstellung ist 4.

Windows erlaubt normalerweise den Gebrauch von bis zu 4 COM Ports, aber kann für bis zu 9 Ports konfiguriert werden. TestPoint arbeitet mit jedem seriellem Port zusammen, der via Windows Treiber COMM.DRV unterstützt wird. Einige Mehrfach seriellen Interfacekarten benötigen spezielle Treiber, die den COMM.DRV Treiber ersetzen, somit sehen sie wie reguläre Windows Ports aus. **Der Test ist: Funktioniert der COM Port mit einem Standard Windows Accessory, wie beispielsweise dem Terminal Programm?** Falls ja, dann sollte es auch mit TestPoint funktionieren.

Falls serielle Verbindungen mit hoher Übertragungsrate, oberhalb von 4800 Baud, verwendet werden, sollte sichergestellt sein, daß der serielle Port die geeignete Hardware bietet. Einige ältere Hardwarebausteine benutzen noch den 8250 UART Chip, der nicht zuverlässig mit den hohen Raten unter Windows laufen kann. Neuere Komponenten wie beispielsweise der 16550 UART Chip unterstützen diese hohen Übertragungsraten.

Serielle Verbindungen besitzen immer eine gewisse Anzahl an Parametern, die an beiden Enden der Kommunikation gesetzt werden müssen, damit die Übertragung korrekt funktioniert. Die Einstellungen sind: Baudrate, Anzahl der Datenbits, Parity, und Anzahl der Stopbits. Beispielsweise: 9600 Baud, 8 Datenbits, keine Parity, 1 Stopbit. Diese Parameter können im Windows Control Panel gesetzt werden, und TestPoint wird diese benutzen. Der Vorteil dieses Ansatzes ist, daß falls die Device Settings später geändert werden, oder die Applikation auf einem anderen Rechner gestartet wird, das Control Panel dafür verwendet werden kann, die Settings zu ändern anstatt daß die TestPoint Applikation geändert werden muß.

Falls die Übertragungsparameter in der eigenen Applikation gesetzt werden sollen, hängen sie nicht von den Einstellungen des Control Panels ab. In diesem Fall kann die "Set mode" Action des RS232 Objekts verwendet werden, bevor der Port geöffnet wird.

RS232 definiert viele elektrische Signale. Einige Geräte benutzen nur einige wenige dieser Signale, andere benötigen alle. Gewöhnlicherweise wird ein Standard serielles Kabel funktionieren. Gelegentliche jedoch muß ein spezielles Kabel verwendet werden. Die bekannteste Art eines Kabels ist das sogenannte "**Null modem**". RS232 Geräte können konfiguriert werden, die Daten entweder auf Pin 2 oder Pin 3 der konventionellen 25 Pins anzubieten. Falls beide Enden der Verbindung auf den selben Pin übertragen wollen, muß ein Null modem vorgeschaltet werden, das die notwendigen Verbindungen umkehrt.

Einige Geräte unterstützen, einige erfordern, entweder einen Hardware oder einen Software Handshake, um Übertragungsfehler zu vermeiden, bei denen die Daten zu schnell ankommen. Hardware Handshake benutzt die Signale RTS und CTS auf dem RS232 Kabel. Sobald ein Gerät beschäftigt ist, und sein Input Buffer zu voll wird, setzt es ein Signal auf diesen Leitungen, und das andere Ende wartet. Software Handshake benutzt spezielle Zeichen genannt XON und XOFF, die über die regulären Datenleitungen gesendet werden, um den Transmitter zu stoppen, wenn der Empfäänger beschäftigt ist. Diese Methode arbeitet OK, solange man nicht binäre Daten sendet. Um die Verbindung korrekt funktionieren zu lassen, **muß** man wissen, welcher Handshake Typ von dem Gerät verwendet wird. Handshake kann in dem Windows Control Panel, oder mit der "Set handshaking" Action des RS232 Objekts eingestellt werden.

Falls Daten gesendet werden, und keine Daten ausgegeben werden, versuchen Sie bitte, den Handshake auszuschalten.

Es ist ebenfalls möglich, zu experimentieren, in dem das Windows Accessory Programm Terminal verwendet wird, das mit Windows ausgeliefert wird. Es beinhaltet komplette settings für alle Kommunikations- und Handshakeparameter. Sobald das Terminal läuft, können die selben Optionen in TestPoint verwendet werden.

Falls bei der RS232 Kommunikation Fehler auftreten, sollte sichergestellt werden, daß jedes Windows "Shell" Programm wie z.B. HP Dashboard, abgeschaltet ist.

Anhang D: Unterschiede zwischen TestPoint v1 und v2

Dieser Abschnitt beschreibt die Grundsätzlichen Änderungen zwischen TestPoint v1 und v2.

Hinweis: Das Dateiformat hat sich geändert! In TestPoint v2 können v1 Dateien eingelesenwerden, *nicht* jedoch umgekehrt. Das Dateifor-mat der v2 ist wesentlich kompakter als das der v1. **Wenn Sie mehrere TestPoint Anwender haben, die Dateien austauschen müssen, sollten alle auf Version 2.0 umsteigen. Andernfalls ist es ratsam beide Ver-sionen auf dem Computer zu haben.**

Benutzerdefinierte Objekte	Die Möglichkeit hinzugefügt Benutzerschnittstellen zu definieren.
Neue Editier-Funktionen	Neues tabelliertes Objekt Fenster mit „Settings“, „Action List“, usw. Erweiterte „Ziehen&Ablegen“ Funktion mit automatischem Umblättern. XRef (Kreuz-Referenzen). Erweiterte Ausdrucke von Anwendungen (ausgewählte Objekte, nur teilweise Einstellungen). Ausgedrucktes Listing von gelöschten Objekt-Referenzen. „Print Cross-Reference“ (Drucke Kreuz-Referenz). Drucke Objekt und drucke Panel Kommandos. Aktivieren und deaktivieren von Aktions Listen Zeilen. Fangen auf frei definierbare Raster. Passwort für Dateien.
VBX Custom Controls	VBX Objekt zugefügt.
OLE	OLE Objekt zugefügt.
Erweiterte Programmierung	Objektgruppen mit Indizierung (Objektfelder) Indirekte Objekte.
Berichtserzeugung	Report Objekt zugefügt.
A/D	Rohdaten Modus - Höhere Geschwindigkeit bei der Digitalisierung von Meßdaten. Einschließlich der nachträglichen Konvertierung auf

	Spannungen Weitere Aktionen zugefügt.
Daten Formatierung	ASCII Formatierung zugefügt (Anzahl der Stellen, usw.) einschließlich Vektoren.
Objekt Erweiterungen	Datei: Neue Aktionen zugefügt („Get file size“, „Test EOF“...) Code: Den „hwnd“ Argumententyp für Windows zugefügt. Graph: XY Datenpaar-Modus zugefügt. Math: Neue Funktionen zugefügt. Error-Handler: Die Möglichkeit benutzerdefinierte Fehler zu erzeugen zugefügt. Task: Start/Stop Aktionen´zugefügt. Die „Return“-Aktion zugefügt. Die „Action list at termination“ Aktion zugefügt. Panel: Setzt Fokus Aktion zugefügt. Bewege Objekt Aktion zugefügt. DIO: Ereignis auf eine bestimmte Bit-Maske zugefügt.
Case	„Case“ Objekt zugefügt.

Notizen

Notizen