LabVIEW™

Benutzerhandbuch

NATIONAL INSTRUMENTS™

Auflage Juli 1998
Artikelnummer 321200B-01

Deutschsprechende Zweigstellen

National Instruments
Germany GmbH
Konrad-Celtis-Straße 79
D-81369 München

National Instruments
Ges.m.b.H.
Plainbachstraße 12
A-5101 Salzburg-Bergheim

National Instruments
Switzerland
Sonnenbergstraße 53
CH-5408 Ennetbaden

Internet-Unterstützung

E-Mail: ni.germany@natinst.com
FTP Site: ftp.natinst.com
Web Adresse: www.natinst.com

Mailbox Unterstützung–Bulletin Board Support (BBS)

Deutschland: 089/71 25 61
USA: 512 794 5422

Telefon–Unterstützung

Deutschland:	Tel: 089/741 31 30	Fax: 089/714 60 35	
Österreich:	Tel: 0662/45 79 90 0	Fax: 0662/45 79 90 19	
Schweiz:	Tel: 056/200 51 51	Fax: 056/200 51 55	Tel: 022/980 05 11 (Genf)

Internationale Zweigstellen

Australien 03 9879 5166, Belgien 02 757 00 20, Brasilien 011 288 3336, Dänemark 45 76 26 00,
Finnland 09 725 725 11, Frankreich 01 48 14 24 24, Großbritannien 01635 523545, Holland 0348 433466,
Hongkong 2645 3186, Israel 03 6120092, Italien 02 413091, Japan 03 5472 2970,
Kanada (Ontario) 905 785 0085, Kanada (Québec) 514 694 8521, Korea 02 596 7456, Mexiko 5 520 2635,
Norwegen 32 84 84 00, Singapur 2265886, Spanien 91 640 0085, Schweden 08 730 49 70, Taiwan 02 377 1200

National Instruments - Firmenhauptsitz

11500 North Mopac Expressway Austin, Texas 78759 USA Tel: (+1) 512 794 0100

Wichtige Informationen

Garantie

Für die Datenträger, auf denen Sie die Software von National Instruments erhalten, wird für den Zeitraum von 90 Tagen nach dem Versand (nachweisbar durch Quittungen oder andere Dokumente) garantiert, daß sie keine Defekte in Material und Verarbeitung aufweisen, durch die die Ausführung der Programmanweisungen behindert wird. Sofern National Instruments während der Garantiezeit über bestehende Schäden informiert wird, entscheidet National Instruments nach eigenem Ermessen, ob Software-Datenträger, auf denen die Ausführung der Programmanweisungen nicht möglich ist, entweder repariert oder ersetzt werden. National Instruments leistet keine Gewähr dafür, daß die Ausführung der Software unbehindert und fehlerfrei erfolgen kann.

Es können keine Einsendungen zur Garantiebearbeitung entgegengenommen werden, die nicht deutlich auf der Außenseite durch eine Autorisierungsnummer für die Rücksendung, eine sogenannte RMA-Nummer (Return Material Authorization), gekennzeichnet sind. Lassen Sie sich vom Werk eine solche Autorisierungsnummer zuweisen. National Instruments übernimmt die Versandkosten für Teile, die im Rahmen einer Garantieleistung an den Kunden zurückgesendet werden.

National Instruments geht davon aus, daß die Informationen in diesem Handbuch korrekt sind. Die technischen Angaben in diesem Dokument wurden sorgfältig überprüft. Falls trotzdem technische oder typographische Fehler vorhanden sind, behält sich National Instruments das Recht vor, in nachfolgenden Auflagen dieses Dokuments Änderungen ohne weitere Mitteilung an die Benutzer dieser Auflage durchzuführen. Leser, die der Meinung sind, daß ein Fehler vorliegt, sollten sich direkt an National Instruments wenden. Unter keinen Umständen übernimmt National Instruments eine Haftung für Schäden, die eventuell aufgrund dieses Dokuments bzw. der darin enthaltenen Informationen oder im Zusammenhang damit entstehen.

NEBEN DER HIER BESCHRIEBENEN GARANTIE ÜBERNIMMT NATIONAL INSTRUMENTS WEDER AUSDRÜCKLICHE NOCH STILLSCHWEIGENDE GEWÄHRLEISTUNGEN. INSBESONDERE WIRD KEINE GARANTIE FÜR MARKTGÄNGIGKEIT ODER DIE EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ÜBERNOMMEN. DIE SCHADENSERSATZANSPRÜCHE FÜR SCHÄDEN, DIE DURCH VERSCHULDEN ODER FAHRLÄSSIGKEIT VON NATIONAL INSTRUMENTS VERURSACHT WERDEN, SIND AUF DIE HÖHE DES KAUFPREISES BESCHRÄNKT, DEN DER KUNDE FÜR DAS PRODUKT BEZAHLT HAT. NATIONAL INSTRUMENTS IST NICHT HAFTBAR FÜR SCHÄDEN, DIE DURCH DEN VERLUST VON DATEN, GEWINNEINBUSSEN, DURCH DIE EINSCHRÄNKUNG DER VERWENDBARKEIT DER PRODUKTE ODER DURCH NEBEN- ODER FOLGESCHÄDEN ENTSTEHEN. DIES GILT AUCH DANN, WENN NATIONAL INSTRUMENTS ÜBER DIE MÖGLICHKEIT SOLCHER SCHÄDEN UNTERRICHTET WURDE. Diese Einschränkung der Haftung von National Instruments gilt unabhängig von der Art der jeweiligen Handlung, sei es im Rahmen eines Vertrags oder als unerlaubte Handlung, und gilt auch in Fällen von Fahrlässigkeit. Gerichtliche Schritte gegen National Instruments müssen innerhalb von einem Jahr nach Eintreten des Grundes für diese Schritte eingeleitet werden. National Instruments ist nicht haftbar für die Verzögerung bestimmter Leistungen, die durch Vorgänge verursacht wird, über die National Instrument bei einer vernünftigen Abwägung keine Kontrolle ausüben kann. Die hiermit gegebene Garantie erstreckt sich nicht auf Schäden, Defekte, Fehlfunktionen oder Funktionsausfälle, die dadurch verursacht werden, daß der Benutzer die Anleitungen von National Instruments für die Installation, den Betrieb und die Wartung nicht einhält. Dieser Garantieausschluß gilt ebenso für Schäden, die durch Modifikationen, durch Mißbrauch oder fahrlässiges Verhalten auf seiten des Benutzers, durch Stromausfälle oder Spannungsstöße, durch Brand, Überschwemmungen, Unfälle, Handlungen einer dritten Partei oder andere Vorfälle verursacht werden, die nicht in einem vernünftigen Rahmen kontrolliert werden können.

Copyright

Warenzeichen

WARNHINWEIS BEZÜGLICH DES EINSATZES VON PRODUKTEN VON NATIONAL INSTRUMENTS FÜR MEDIZINISCHE UND KLINISCHE ZWECKE

Die Produkte von National Instruments sind aufgrund der verwendeten Komponenten und der angewendeten Testverfahren nicht dazu geeignet, die notwendige Zuverlässigkeit für den Einsatz in der Behandlung und Diagnose von Patienten zu gewährleisten. Durch den Einsatz von National Instruments-Produkten in Anwendungen, die für medizinische oder klinische Zwecke gedacht sind, können mögliche Verletzungen durch das Versagen eines Produkts oder durch Fehler des Benutzers oder Anwendungsentwicklers entstehen. Alle Einsätze und Anwendungen von Produkten von National Instruments, die im Rahmen einer medizinischen oder klinischen Behandlung verwendet werden, dürfen nur durch ordnungsgemäß ausgebildetes und qualifiziertes medizinisches Personal erfolgen. Alle gebräuchlichen medizinischen Sicherheitsmaßnahmen, Geräte und Verfahren, die im jeweiligen Einzelfall zum Schutz vor ernsten

Verletzungen mit möglicher Todesfolge angemessen sind, müssen ununterbrochen angewendet werden, solange Produkte von National Instruments eingesetzt werden. Die Produkte von National Instruments sind NICHT als Ersatz für etablierte Verfahren, Vorgehensweisen oder Geräte gedacht, die zur Überwachung oder zum Schutz der Gesundheit und zur allgemeinen Sicherheit von Patienten und Personal in der medizinischen oder klinischen Behandlung eingesetzt werden.

Inhaltsverzeichnis

Über dieses Handbuch

Kapitel 1
Einleitung

Teil I
Einführung in das Programmieren in G

Kapitel 2
VIs erstellen

Kapitel 3
Schleifen und Diagramme

Kapitel 4
Case- und Sequenzstrukturen und der Formel-Knoten

Kapitel 5
Arrays, Cluster und Graphen

Kapitel 6
Strings und Datei I/O

TEIL II
I/O-Schnittstellen

Kapitel 7
Erste Schritte mit einem LabVIEW-Gerätetreiber

Kapitel 8
Tutorial für LabVIEW VISA

Kapitel 9
Einführung in die LabVIEW GPIB-Funktionen

Kapitel 10
VIs für den seriellen Anschluß

TEIL III
Analyse

Kapitel 11
Einführung in die Analyse in LabVIEW

Kapitel 12
Signalerzeugung

Kapitel 13
Digitale Signalverarbeitung

Kapitel 14
Fensterglättung

Kapitel 15
Spektralanalyse und -messung

Kapitel 16
Filterung

Kapitel 17
Kurvenanpassung

Kapitel 18
Lineare Algebra

Kapitel 21
TCP und UDP

Kapitel 22
ActiveX-Unterstützung

Kapitel 23
DDE verwenden

Kapitel 24
AppleEvents

Kapitel 25
Programm-zu-Programm-Kommunikation

Teil V
Fortgeschrittenes Programmieren in G

Kapitel 26
VIs anpassen

Kapitel 27
Attribute der Frontpanel-Objekte

Kapitel 28
Programmdesign

Kapitel 29
Weiteres Vorgehen

Anhang A
Analyse-Referenzen

Anhang B
Häufig gestellte Fragen

Anhang C
Kundenbetreuung

Glossar

Stichwörterverzeichnis

Abbildungen

Tabellen

Übungen

Über dieses Handbuch

Das *LabVIEW Benutzerhandbuch* enthält Informationen zum Erstellen von virtuellen Instrumenten (VIs). Außerdem bietet dieses Handbuch Erläuterungen zu den Schnittstellen für die Eingabe und Ausgabe von Daten, Hinweise zum Einsatz von LabVIEW-VIs für die Durchführung von Analyseoperationen sowie Informationen darüber, wie LabVIEW die Kommunikation in einem Netzwerk und zwischen den Anwendungen behandelt. Lesen Sie bitte die *LabVIEW Versionshinweise*, bevor Sie das *LabVIEW Benutzerhandbuch* benutzen.

Gliederung dieses Handbuchs

Das *LabVIEW Benutzerhandbuch* ist folgendermaßen gegliedert:

- Kapitel 1, *Einleitung*, enthält eine Einführung in den einzigartigen Ansatz von LabVIEW bei der Programmierung. Außerdem wird in diesem Kapitel erklärt, wie Sie damit beginnen, LabVIEW zur Entwicklung von Programmen zu verwenden.

Teil I, Einführung in das Programmieren in G

Dieser Teil des Handbuchs enthält grundsätzliche Informationen über das Erstellen von virtuellen Instrumenten (VIs), über die Verwendung von VIs in anderen VIs, über Programmierstrukturen, wie z.B. Schleifen, und über Datenstrukturen, wie z.B. Arrays und Strings.

Teil I, *Einführung in das Programmieren in G*, enthält die folgenden Kapitel:

- Kapitel 2, *VIs erstellen*, erläutert, wie ein VI mit einem Frontpanel, d.h. der eigentlichen Benutzeroberfläche, und einem Blockdiagramm, d.h. dem Quellcode, erstellt wird. Nachdem Sie ein VI erstellt haben, können Sie es in anderen VIs einsetzen.
- Kapitel 3, *Schleifen und Diagramme*, macht Sie damit vertraut, wie Sie Teile des Blockdiagramms mit Hilfe einer While-Schleife bzw. einer For-Schleife wiederholen können. In diesem Kapitel wird außerdem erklärt, wie Sie mehrere Punkte nacheinander grafisch auf einem Diagramm anzeigen können.

- Kapitel 4, *Case- und Sequenzstrukturen und der Formel-Knoten*, erläutert die Verwendung der Case-Struktur, die eine Bedingungsstruktur darstellt, der Sequenzstruktur, die zum Festlegen einer Ausführreihenfolge nützlich ist, und des Formelknotens, der das Ausführen mathematischer Formeln unterstützt.
- Kapitel 5, *Arrays, Cluster und Graphen*, beschreibt, wie eine Gruppe oder ein Array von Datenpunkten auf einem Graphen angezeigt wird. Sie können sowohl Skalierungsparameter als auch ein Array von Datenpunkten an einen Graphen übergeben, indem Sie ein Cluster erstellen, das aus einer Gruppe unterschiedlicher Datentypen besteht.
- Kapitel 6, *Strings und Datei I/O*, beschreibt String-Bedienelemente und String-Anzeigen sowie Eingabe- und Ausgabeoperationen.

Teil II, I/O-Schnittstellen

Dieser Teil des Handbuchs enthält grundsätzliche Informationen zu den Schnittstellen, über die Daten ein- und ausgegeben werden können. Dazu gehören: Datenerfassung, GPIB, Seriell und VXI. Im Handbuch *Grundlagen der Datenerfassung mit LabVIEW* finden Sie einführende Informationen zur Echtzeit-Datenerfassung. VISA (Virtual Instrument Software Architecture) bietet die Möglichkeit, über eine einzige Software eine Schnittstelle für GPIB-, serielle und VXI-Instrumente herzustellen. Die LabVIEW-Anwendungen, die für ein spezielles Instrument entwickelt wurden, werden als Instrumententreiber bezeichnet. National Instruments stellt mehrere Instrumententreiber zur Verfügung, die die VISA-Bibliothek benutzen. Sie können aber ebenso Ihre eigenen Instrumententreiber anfertigen.

Teil II, *I/O-Schnittstellen*, enthält die folgenden Kapitel:

- Kapitel 7, *Erste Schritte mit einem LabVIEW-Gerätetreiber*, erläutert, wie Sie die Instrumententreiber von National Instruments erstellen und verwenden können.
- Kapitel 8, *Tutorial für LabVIEW VISA*, zeigt Ihnen, wie Sie gebräuchliche VISA-Anwendungen durch nachrichten- und registerorientierte Kommunikation implementieren können und wie Sie mit Ereignissen und der Sperrfunktion umgehen.
- Kapitel 9, *Einführung in die LabVIEW GPIB-Funktionen*, erläutert, wie GPIB arbeitet und welche Unterschiede zwischen den Schnittstellen IEEE 488 und IEEE 488.2 bestehen.
- Kapitel 10, *VIs für den seriellen Anschluß*, erläutert wichtige Faktoren, die die serielle Kommunikation beeinflussen.

Teil III, Analyse

Dieser Teil des Handbuchs enthält grundsätzliche Informationen zur Analyse von Daten, zur Signalverarbeitung und Signalerzeugung, zu linearer Algebra, zur Kurvenanpassung, zu Wahrscheinlichkeitsaspekten und zur Statistik.

Teil III, *Analyse*, enthält die folgenden Kapitel:

- Kapitel 11, *Einführung in die Analyse in LabVIEW*, bietet eine Einführung in die Konzepte, die allen Analyseanwendungen zugrundeliegen, einschließlich unterstützter Funktionalitäten, Notations- und Benennungskonventionen und Abtastsignalmethoden.
- Kapitel 12, *Signalerzeugung*, erläutert, wie Signale unter Verwendung der normalisierten Frequenz erzeugt werden und wie ein simulierter Funktionsgenerator angefertigt wird.
- Kapitel 13, *Digitale Signalverarbeitung*, zeigt die Unterschiede zwischen Fast Fourier Transform (FFT/Schnelle Fourier-Transformation) und Discrete Fourier Transform (DFT/Diskrete Fourier-Transformation).
- Kapitel 14, *Fensterglättung*, erläutert, wie durch Verwendung von Fenstern Leckeffekte verhindert werden und die Analyse erfaßter Signale verbessert wird.
- Kapitel 15, *Spektralanalyse und -messung*, zeigt, wie Sie das Amplituden- und Phasenspektrum bestimmen, einen Spektrumanalysator entwickeln und die harmonische Gesamtverzerrung THD (Total Harmonic Distortion) bestimmen.
- Kapitel 16, *Filterung*, erläutert, wie Sie mit IIR (Infinite Impulse Response)-, FIR (Finite Impulse Response)- und nicht-linearen Filtern unerwünschte Frequenzen aus Signalen herausfiltern können.
- Kapitel 17, *Kurvenanpassung*, zeigt, wie Informationen aus einem Datensatz extrahiert werden, um Beschreibungen von Datentrends zu erstellen.
- Kapitel 18, *Lineare Algebra*, erläutert, wie Matrixberechnungen und -analysen durchgeführt werden.
- Kapitel 19, *Wahrscheinlichkeit und Statistik*, erläutert einige grundsätzliche Konzepte der Wahrscheinlichkeitsrechnung und Statistik und zeigt, wie diese Konzepte zur Lösung realer Probleme verwendet werden können.

Teil IV, Kommunikation im Netzwerk und zwischen den Anwendungen

Dieser Teil des Handbuchs enthält grundsätzliche Informationen zur Kommunikation im Netzwerk und zwischen den Anwendungen.

Teil IV, *Kommunikation im Netzwerk und zwischen den Anwendungen*, enthält die folgenden Kapitel:

- Kapitel 20, *Einführung in die Kommunikation*, bietet eine Einführung in die Art und Weise, wie LabVIEW die Kommunikation im Netzwerk und zwischen den Anwendungen bewältigt.
- Kapitel 21, *TCP und UDP*, beschreibt die Protokolle Transmission Control Protocol (TCP) und Internet Protocol (IP) sowie Internet-Adressen.
- Kapitel 22, *ActiveX-Unterstützung*, erläutert, wie LabVIEW als ActiveX-Server und -Client verwendet wird. ActiveX entspricht der OLE Automation-Kommunikation.
- Kapitel 23, *DDE verwenden*, erläutert, wie der dynamische Datenaustausch DDE (Dynamic Data Exchange) zur Kommunikation zwischen Windows-Anwendungen eingesetzt wird. DDE kann auf einem Client, auf einem Server oder über ein Netzwerk verwendet werden.
- Kapitel 24, *AppleEvents*, zeigt, wie AppleEvents zur Kommunikation zwischen LabVIEW und anderen Anwendungen für den Macintosh eingesetzt werden. LabVIEW kann als AppleEvents-Server und -Client verwendet werden.
- Kapitel 25, *Programm-zu-Programm-Kommunikation*, erläutert, wie LabVIEW über die Programm-zu-Programm-Kommunikation (PPC/Program-to-Program Communication) mit anderen Anwendungen auf dem Macintosh kommunizieren kann.

Teil V, Fortgeschrittenes Programmieren in G

Dieser Teil des Handbuchs enthält Informationen über das Anpassen von VIs, das programmatische Steuern von Frontpanel-Objekten, VIs und LabVIEW sowie Tips für das Design komplexer Anwendungen.

Teil V, *Fortgeschrittenes Programmieren in G*, enthält die folgenden Kapitel.

- Kapitel 26, *VIs anpassen*, zeigt, wie das Aussehen und das Ausführverhalten eines laufenden VIs mit den Optionen **VI-Einstellungen…** und **Einstellungen SubVI Knoten…** angepaßt werden kann.

- Kapitel 27, *Attribute der Frontpanel-Objekte*, beschreibt die als Attributknoten bezeichneten Objekte, die spezielle Blockdiagrammknoten zur Steuerung des Aussehens und der funktionalen Charakteristiken von Bedien- und Anzeigeelementen darstellen.
- Kapitel 28, *Programmdesign*, erläutert Methoden, die zum Erstellen von Programmen verwendet werden können und beschreibt Richtlinien für den Programmierstil.
- Kapitel 29, *Weiteres Vorgehen*, bietet Informationen über Ressourcen, die Ihnen das Erstellen gelungener Anwendungen erleichtern können.

Anhänge, Glossar und Stichwörtverzeichnis

- Anhang A, *Analyse-Referenzen*, führt die Referenzmaterialien auf, die in LabVIEW beim Herstellen der Analyse-VIs verwendet werden können. Diese Materialien enthalten zusätzliche Informationen zu den Theorien und Algorithmen, die in der Analysebibliothek implementiert sind.
- Anhang B, *Häufig gestellte Fragen*, enthält Antworten auf häufig gestellte Fragen zur Netzwerkkommunikation von LabVIEW und zum Instrument-I/O, insbesondere GPIB- und serielle I/O-Vorgänge.
- Anhang C, *Kundenbetreuung*, enthält Formulare, die Ihnen das Sammeln der technischen Informationen erleichtern sollen, die von unseren Mitarbeitern zur Lösung Ihrer technischen Probleme benötigt werden. Der Anhang enthält außerdem ein Formular, auf dem Sie Kommentare zur Produktdokumentation einreichen können.
- Das *Glossar* enthält eine alphabetische Liste der Begriffe, die in diesem Handbuch verwendet werden, einschließlich Abkürzungen, Initialwörter, metrische Präfixe, Gedächtnishilfen und Symbole.
- Der *Stichwörterverzeichnis* enthält eine alphabetische Liste mit Seitenangaben für die wichtigen Begriffe und Themen in diesem Handbuch.

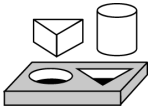
Schreibkonventionen in diesem Handbuch

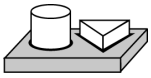
In diesem Handbuch werden die folgenden Schreibkonventionen verwendet:

< >
Spitze Klammern umgeben den Namen einer Taste auf der Tastatur, z.B. <Umschalt>. Spitze Klammern, die durch eine Ellipse getrennte Zahlen enthalten, geben einen Wertebereich für ein Bit oder einen Signalnamen an, z.B. DBIO<3...0>.

-
Ein Bindestrich zwischen zwei oder mehreren Tastennamen in spitzen Klammern weist darauf hin, daß Sie die angegebenen Tasten gleichzeitig drücken müssen, z.B. <Strg-Alt-Entf>.

»
Das Symbol **»** kennzeichnet die Reihenfolge, in der Sie verschachtelte Menübefehle und Dialogfeldoptionen ausführen müssen. Durch die Folge **Datei»Seite einrichten»Optionen»Schriftarten ersetzen** werden Sie aufgefordert, das Menü **Datei** zu öffnen, danach den Menübefehl **Seite einrichten** aufzurufen, dann **Optionen** und im letzten Dialogfeld schließlich die Option **Schriftarten ersetzen** zu wählen.


Wenn sich dieses Symbol links neben Text in fetter Schrift befindet, weist es auf den Anfang einer Übung hin, die Sie Schritt für Schritt mit einem bestimmten Aspekt von LabVIEW vertraut macht.


Wenn sich dieses Symbol links neben Text in fetter Schrift befindet, weist es auf das Ende einer Übung hin, die Sie Schritt für Schritt mit einem bestimmten Aspekt von LabVIEW vertraut macht.


Wenn sich dieses Symbol links neben Text in fetter, kursiver Schrift befindet, kennzeichnet es einen Hinweis, der wichtige Informationen enthält.


Wenn sich dieses Symbol links neben Text in fetter, kursiver Schrift befindet, weist es auf einen Text hin, der Vorsichtsmaßnahmen beschreibt, durch die Verletzungen, Datenverluste oder Systemabstürze vermieden werden können.

fett
Text in fetter Schrift kennzeichnet die Namen von Menüs, Menüoptionen, Parametern, Dialogfeldern, Dialogfeldschaltflächen oder -optionen, Symbolen, Fenstern, Registern in Windows 95 oder LEDs.

fett kursiv
Text in fetter, kursiver Schrift kennzeichnet ein Übungsziel, einen Hinweis, eine Vorsichtsmaßnahme oder eine Warnung.

fett gesperrt	Text in fetter, gesperrter Schrift kennzeichnet Meldungen und Antworten, die vom Computer automatisch auf dem Bildschirm ausgegeben werden.
kursiv	Text in kursiver Schrift kennzeichnet Variablen, Hervorhebungen, Querverweise oder Einführungen in wichtige Konzepte. Dieser Schriftstil kennzeichnet außerdem auch Textstellen, an denen Sie das entsprechende Wort oder den korrekten Wert einsetzen müssen, wie z.B. Windows 3.*x*.
`gesperrt`	Durch diesen Schriftstil werden sowohl Codeabschnitte, Programmierbeispiele und Syntaxbeispiele als auch Texte oder Zeichen gekennzeichnet, die Sie exakt in der angegebenen Weise über die Tastatur eingeben müssen. Dieser Schriftstil wird auch für die Namen von Diskettenlaufwerken, für Pfade, Verzeichnisse, Programme, Unterprogramme, Subroutinen, Gerätenamen, Funktionen, Operationen, Variablen, Dateinamen und -erweiterungen und für Mitteilungen und Kommentare verwendet, die einem Programm entnommen wurden.
Pfade	Bei den in diesem Handbuch angegebenen Pfaden werden Laufwerksnamen, Verzeichnisse, Ordner und Dateien durch einen Backslash (\) voneinander getrennt.
Plattform	Ein Text in dieser Schrift kennzeichnet Informationen, die für eine spezielle Plattform gelten.

Verwandte Dokumentationsmaterialien

Die folgenden Unterlagen könnten sich bei der Lektüre dieses Handbuchs als nützlich erweisen:

- *Referenzhandbuch zur Programmierung in G*
- *Grundlagen der Datenerfassung mit LabVIEW*
- *LabVIEW Funktionen und VI Referenzhandbuch*
- *LabVIEW Kurzanleitung*
- *LabVIEW Online-Referenz*—verfügbar durch Wahl von **Hilfe»Online-Referenz**
- *LabVIEW Online Tutorial* **(nur Windows)**—wird über das LabVIEW-Dialogfeld gestartet
- *Referenzkarte G-Programmierung*
- *Erste Schritte mit LabVIEW*
- LabVIEW *Versionshinweise*
- LabVIEW *Update-Hinweise*

Mitteilungen von Kunden

National Instruments begrüßt Ihre Kommentare zu unseren Produkten und Handbüchern. Wir sind besonders auch an den Anwendungen interessiert, die Sie mit unseren Produkten entwickeln, und wir möchten Ihnen behilflich sein, falls Sie auf Probleme stoßen. Um Ihnen den Kontakt mit uns so einfach wie möglich zu machen, enthält dieses Handbuch Formulare, in denen Sie Ihre Kommentare und Konfigurationsdaten eintragen können. Diese Formulare befinden sich in Anhang C, *Kundenbetreuung*, am Ende dieses Handbuchs.

1

Einleitung

In diesem Kapitel wird der einzigartige Programmierungsansatz von LabVIEW besprochen. Außerdem finden Sie hier Informationen, die die Entwicklung von Programmen mit Hilfe von LabVIEW erklären. In Querverweisen wird auf weitere Informationen in anderen Kapiteln und Handbüchern verwiesen.

Was ist LabVIEW?

LabVIEW ist eine in der Programmentwicklung eingesetzte Anwendung, die ähnlich wie C oder BASIC eine moderne Umgebung zur Programmentwicklung schafft und mit LabWindows/CVI von National Instruments vergleichbar ist. Von diesen Anwendungen unterscheidet sich LabVIEW jedoch in einem wichtigen Aspekt. Andere Programmiersysteme verwenden Programmiersprachen auf *Textbasis* zum Erstellen von Codezeilen, während LabVIEW die *grafische* Programmiersprache *G* benutzt, um Programme in Blockdiagrammform zu erstellen.

Ähnlich wie C oder BASIC ist auch LabVIEW ein Programmiersystem für allgemeine Zwecke, das umfangreiche Bibliotheken mit Funktionen für alle Programmieraufgaben umfaßt. LabVIEW verfügt über Bibliotheken für die Datenerfassung, für GPIB und serielle Instrumentensteuerung sowie für die Analyse, Darstellung und Speicherung von Daten. Darüber hinaus bietet LabVIEW auch konventionelle Werkzeuge für die Programmentwicklung. Sie sind damit also nicht nur in der Lage, Breakpoints zu setzen und die Ausführung zu animieren, wenn Sie sehen möchten, wie die Daten das Programm durchlaufen. Zur Erleichterung des Debugging-Verfahrens und der allgemeinen Programmentwicklung können Sie außerdem das Programm in Einzelschritten durcharbeiten.

Wie arbeitet LabVIEW?

Obwohl LabVIEW ein Programmiersystem für allgemeine Zwecke ist, stehen auch spezielle Bibliotheken mit Funktionen und Entwicklungswerkzeugen zur Verfügung, die für die Datenerfassung und Instrumentensteuerung bestimmt sind. Die LabVIEW-Programme werden als *virtuelle Instrumente* (*VI*) bezeichnet, da sie im Erscheinungsbild und in der Funktionsweise tatsächliche Instrumente imitieren können. Diese VIs sind jedoch mit den Funktionen herkömmlicher Sprachprogramme vergleichbar.

Ein solches VI zeichnet sich durch eine interaktive Benutzeroberfläche, ein als Quellcode dienendes Datenflußdiagramm und Icon-Verbindungen aus, über die das VI von anderen VIs auf höheren Ebenen aufgerufen werden kann. Im einzelnen weisen VIs folgende Struktur auf:

- Die interaktive Benutzeroberfläche eines VIs wird als *Frontpanel* bezeichnet, weil es die Schalttafel eines realen Instruments nachbildet. Das Frontpanel kann mit Drehknöpfen, Drucktasten, Graphen und anderen Bedien- und Anzeigeelementen ausgestattet sein. Die Dateneingabe erfolgt über Maus oder Tastatur, und die Ergebnisse können anschließend auf dem Computerbildschirm eingesehen werden.
- Das VI erhält Anweisungen von einem *Blockdiagramm*, das in G erstellt wird. Bei dem Blockdiagramm handelt es sich um eine verbildlichte Lösung für eine Programmierungsaufgabe. Das Blockdiagramm stellt außerdem den Quellcode für das VI dar.
- VIs sind in hierarchischer Anordnung und modular einsetzbar. Sie können also sowohl als Top-Level-Programme als auch als Unterprogramme im Rahmen anderer Programme verwendet werden. Ein VI, das einem anderen VI untergeordnet ist, wird als *SubVI* bezeichnet. *Icon und Anschluß* eines VIs arbeiten wie eine grafische Parameterliste, so daß andere VIs Daten an ein SubVI weiterleiten können.

Diese Funktionen machen deutlich, daß LabVIEW das Konzept des *modularen Programmierens* verfolgt und unterstützt. Sie können eine Anwendung in eine Reihe von Aufgaben unterteilen, die sich dann weiter unterteilen lassen, bis eine komplizierte Anwendung in eine Reihe von einfachen Teilaufgaben umgewandelt wurde. Danach erstellen Sie für jede Teilaufgabe ein VI und kombinieren diese VIs dann in einem anderen Blockdiagramm, um die umfangreichere Aufgabe auszuführen. Das Top-Level-VI besteht schließlich aus einer Ansammlung von SubVIs, die einzelne Anwendungsfunktionen repräsentieren.

Da jedes SubVI unabhängig vom Rest der Anwendung ausgeführt werden kann, wird das Debugging wesentlich erleichtert. Außerdem führen viele untergeordnete SubVIs oft Aufgaben durch, die mehreren Anwendungen gemeinsam sind. Sie können also ein spezielles Set von einzelnen SubVIs entwickeln, die sich gut für verschiedene, von Ihnen anzufertigende Anwendungen eignen.

Programmieren in G

G ist die leicht einsetzbare Programmiersprache für den grafischen Datenfluß, die LabVIEW zugrundeliegt. Mit G werden wissenschaftliche Berechnungen, die Überwachung und Steuerung einzelner Prozesse und der Einsatz von Test- und Meßanwendungen erleichtert. Darüber hinaus gibt es eine Vielzahl von anderen Anwendungsmöglichkeiten für G.

In Teil I, *Einführung in das Programmieren in G*, werden die Funktionen von G beschrieben, die Sie zu Beginn der meisten Anwendungsmöglichkeiten von LabVIEW benötigen. Eine ausführlichere Erklärung der Funktionen von LabVIEW finden Sie im *Referenzhandbuch zur Programmierung in G*.

Die folgende Liste enthält Beschreibungen der Grundkonzepte von G, die in diesem Handbuch besprochen werden.

- VIs—Virtuelle Instrumente (VI) weisen drei Hauptbestandteile auf: Frontpanel, Blockdiagramm und Icon/Anschluß. Das Frontpanel kennzeichnet die Benutzeroberfläche des VIs. Das Blockdiagramm besteht aus dem ausführbaren Code, den Sie mit Hilfe von Knoten, Terminals und Verbindungen erstellen. Icons bzw. Anschlüsse dienen dazu, ein VI als SubVI in einem Blockdiagramm eines anderen VIs einzusetzen. Weitere Informationen über VIs finden Sie in Kapitel 2, *VIs erstellen* und Kapitel 26, *VIs anpassen*.
- Schleifen und Diagramme—G bietet zwei Strukturen zum wiederholten Ausführen eines Subdiagramms—die *While-Schleife* und die *For-Schleife*. Beide Strukturen sind in der Größe veränderbare Felder. Das Subdiagramm, das wiederholt werden soll, wird innerhalb der Ränder der Schleifenstruktur plaziert. Die While-Schleife wird ausgeführt, solange der Wert des Bedingungsanschlusses TRUE (WAHR) ist. Die For-Schleife wird nur für eine bestimmte Anzahl von Wiederholungen ausgeführt. In Diagrammen kann der Benutzer Trendinformationen in Realzeit einsehen. Weitere Informationen über Schleifen und Diagramme finden Sie in Kapitel 3, *Schleifen und Diagramme*.

- Case- und Sequenzstrukturen—Die *Case-Struktur* ist eine Struktur zur bedingten Verzweigungssteuerung, durch die aufgrund einer bestimmten Eingabe ein Subdiagramm ausgeführt wird. Eine *Sequenzstruktur* ist eine Programmsteuerungsstruktur, durch die Subdiagramme in numerischer Folge ausgeführt werden. Weitere Informationen über Case- und Sequenzstrukturen finden Sie in Kapitel 4, *Case- und Sequenzstrukturen und der Formel-Knoten*.
- Attributknoten—*Attributknoten* sind spezielle Blockdiagrammknoten, die Sie dazu verwenden können, um das Erscheinungsbild und die funktionalen Eigenschaften von Bedien- und Anzeigeelementen zu steuern. Weitere Informationen über Attributknoten finden Sie in Kapitel 27, *Attribute der Frontpanel-Objekte*.
- Arrays, Cluster und Graphen—Ein *Array* ist eine in der Größe veränderbare Ansammlung von Datenelementen desselben Typs. Ein *Cluster* ist eine statistisch relevante Ansammlung von Datenelementen des selben oder unterschiedlichen Typs. Graphen werden üblicherweise zum Anzeigen von Daten verwendet. Weitere Informationen über Arrays, Cluster und Graphen finden Sie in Kapitel 5, *Arrays, Cluster und Graphen*.

Organisation des LabVIEW-Systems (Windows)

Nachdem Sie die Installation entsprechend den Anleitungen in den *LabVIEW Release Notes*, die der Software beiliegen, durchgeführt haben, sollte das LabVIEW-Verzeichnis die folgenden Dateien enthalten.

- `LABVIEW.EXE`—Hierbei handelt es sich um die eigentliche LabVIEW-Programmdatei. Wenn Sie diese Datei aufrufen, wird LabVIEW gestartet.
- `vi.lib`-Verzeichnis—Enthält Bibliotheken der VIs, die mit LabVIEW geliefert werden, z.B. VIs für GPIB, die Analyse und die Datenerfassung (DAQ). Auf die meisten dieser VIs haben Sie Zugriff über die Palette **Funktionen**.
- `examples`-Verzeichnis—Enthält zahlreiche Unterverzeichnisse mit Beispielen. In diesem Verzeichnis befindet sich auch ein VI mit der Bezeichnung `readme.vi`, das als Anleitung für die Beispiele dient.
- `serpdrv` und `daqdrv`—Diese Dateien sind ein Bestandteil der Schnittstelle von LabVIEW für die Kommunikation mit dem seriellen Anschluß bzw. zur DAQ-Kommunikation. Diese Dateien müssen sich im selben Verzeichnis befinden wie `vi.lib`.

- `resource`-Verzeichnis
  - `labview.rsc`, `lvstring.rsc` und `lvicon.rsc`—Datendateien, die von der LabVIEW-Anwendung benutzt werden.
  - **(Windows 3.1)** `lvdevice.dll`—Diese Datei liefert Timing-Hinweise an LabVIEW. Sie muß sich im selben Verzeichnis wie `vi.lib` befinden, damit LabVIEW ausgeführt werden kann.
  - **(Windows 3.1)** `lvimage.dll`—Diese Datei ermöglicht LabVIEW das Laden von Bildern, die mit unterschiedlichen Grafikprogrammen erstellt wurden.
  - `labview50.tlb`—Diese Datei ist eine Typbibliothek, mit deren Hilfe LabVIEW als ActiveX-Server arbeiten kann.
  - `ole_container.dll`—Diese Datei ermöglicht LabVIEW das Anzeigen und Aktualisieren von ActiveX-Containern.
  - `lvwutil32.dll`—Diese Datei wird vom Solution Wizard verwendet, der aufgrund der von Ihnen angegebenen Kriterien DAQ- und Geräte I/O-Beispiele erstellt.
  - `lvjpeg.dll` und `lvpng.dll`—Diese Dateien unterstützen das Anzeigen von JPEG- und PNG-Grafiken in HTML-Dateien, wenn Sie VI-Dokumentationstexte in eine HTML-Datei drucken.
- `Cintools`-Verzeichnis—Enthält Dateien, die zum Erstellen von Code Interface Nodes (CINs) benötigt werden. Mit CINs kann C-Code mit VIs von LabVIEW verbunden werden.
- `visarc`-Datei—Diese Datei ist ein Bestandteil der Schnittstelle von LabVIEW für VISA (Virtual Instrument Software Architecture). VISA bietet eine einzelne Schnittstellenbibliothek zum Steuern von VXI-, GPIB- und seriellen Instrumenten.
- `labview.ini`—Enthält die Konfigurationsoptionen für LabVIEW.
- `Project`-Verzeichnis—Enthält Dateien, die als Objekte im LabVIEW-Menü **Projekt** erscheinen.
- `menus-Verzeichnis`—Enthält Dateien, die zum Konfigurieren der Struktur der Paletten **Elemente** und **Funktionen** verwendet werden.
- `Instr.lib`-Verzeichnis—Enthält Gerätetreiber, die zum Steuern von VXI-, GPIB- und seriellen Instrumenten verwendet werden. Sie sollten Gerätetreiber von National Instruments, die Sie installieren, in diesem Verzeichnis einrichten, da sie in die Palette **Funktionen** aufgenommen werden.

- `Help`-Verzeichnis—Enthält vollständige Online-Dokumentationsmaterialien sowie die Hilfedatei für die Suche nach Beispielen. Diese Datei erleichtert das Suchen nach Beispielen, die Ihrer Anwendung ähneln.
- `Tutorial`-Verzeichnis—Enthält Dateien, die zur Ausführung des Online-Tutorials benötigt werden. Das interaktive Tutorial behandelt die grundlegenden Konzepte der LabVIEW-Umgebung.
- `Activity-Verzeichnis`—In diesem Verzeichnis können Sie die VIs speichern, die Sie erstellen, wenn Sie die Übungen in diesem Handbuch durchführen.
- `User.lib-Verzeichnis`—In diesem Verzeichnis können Sie die von Ihnen erstellten VIs speichern, die Sie häufig verwenden. Die VIs in diesem Verzeichnis werden in der Palette **Funktionen** angezeigt.
- `Wizard`-Verzeichnis—Dieses Verzeichnis erstellt die Option **Solution Wizard** im Menü **Datei**. Sie können dieses Verzeichnis auch dazu verwenden, dem Menü **Datei** Objekte hinzuzufügen.

LabVIEW installiert Treiber-Software für GPIB-, Datenerfassungs- und VXI-Treiberhardware. Konfigurationshinweise finden Sie in Kapitel 2, *Installation und Konfiguration Ihrer Datenerfassungs-Hardware*, in *LabVIEW Grundlagen der Datenerfassung*, im *VXI VI Reference Manual* und in diesem Handbuch in Kapitel 8, *Tutorial für LabVIEW VISA*.

Startfenster in Windows

Beim Starten von LabVIEW erscheint zunächst ein Navigationsdialogfeld, über das Sie leicht auf einführende Materialien, häufig verwendete Befehle und Quick-Tips für das beschleunigte Arbeiten zugreifen können. Sie können verhindern, daß das Navigationsdialogfeld eingeblendet wird, indem Sie ein Kontrollkästchen am unteren Rand des Dialogfelds markieren. Über das Dialogfeld “Voreinstellungen” können Sie das Navigationsdialogfeld wieder aktivieren.

Wenn alle VIs geschlossen sind, erscheint ein ähnliches Dialogfeld. Über die Taste **Kleines Dialogfeld** können Sie auf eine einfachere Version des Dialogfelds umschalten—es werden dann nur die Tasten **Neu**, **Öffnen** und **Beenden** gezeigt.

Organisation des LabVIEW-Systems (Macintosh)

Nachdem Sie die Installation entsprechend der Anleitung in den *LabVIEW Versionshinweise*, die der Software beiliegen, durchgeführt haben, sollte das LabVIEW-Verzeichnis die folgenden Dateien enthalten.

- `LabVIEW`—Hierbei handelt es sich um die eigentliche LabVIEW-Programmdatei. Wenn Sie diese Datei aufrufen, wird LabVIEW gestartet.
- `vi.lib`-Ordner—Enthält Bibliotheken der VIs, die mit LabVIEW geliefert werden, z.B. VIs für GPIB, die Analyse und die Datenerfassung (DAQ). Auf die meisten dieser VIs haben Sie Zugriff über die Palette **Funktionen**.
- `examples`-Ordner—Enthält zahlreiche Unterordner mit Beispielen. In diesem Ordner befindet sich auch ein VI mit der Bezeichnung `readme.vi`, das als Anleitung für die Beispiele dient.
- `resource`-Ordner
  - `lvstring.rsrc` und `lvicon.rsrc`—Datendateien, die von der LabVIEW-Anwendung benutzt werden.
  - `lvjpeg.lib` und `lvpng.lib`—Diese Dateien unterstützen das Anzeigen von JPEG- und PNG-Grafiken in HTML-Dateien, wenn Sie VI-Dokumentationstexte in eine HTML-Datei drucken.
- `cintools`-Ordner—Enthält Dateien, die zum Erstellen von Code Interface Nodes (CINs) benötigt werden. Mit CINs kann C-Code mit VIs von LabVIEW verbunden werden.
- `visarc`-Datei—Diese Datei ist ein Bestandteil der Schnittstelle von LabVIEW für VISA (Virtual Instrument Software Architecture). VISA bietet eine einzelne Schnittstellenbibliothek zum Steuern von VXI-, GPIB- und seriellen Instrumenten.
- `Project`-Ordner—Enthält Dateien, die als Objekte im LabVIEW-Menü **Projekt** erscheinen.
- `menus`-Ordner—Enthält Dateien, die zum Konfigurieren der Struktur der Paletten **Elemente** und **Funktionen** verwendet werden.
- `instr.lib`-Ordner—Enthält Gerätetreiber, die zum Steuern von VXI-, GPIB- und seriellen Instrumenten verwendet werden. Sie sollten Gerätetreiber von National Instruments, die Sie installieren, in diesem Verzeichnis einrichten, da sie in die Palette **Funktionen** aufgenommen werden.

- `help`-Ordner—Enthält vollständige Online-Dokumentationsmaterialien sowie die Hilfedatei für die Suche nach Beispielen. Diese Datei erleichtert das Suchen nach Beispielen, die Ihrer Anwendung ähneln.
- `activity`-Ordner—In diesem Ordner können Sie die VIs speichern, die Sie erstellen, wenn Sie die Übungen in diesem Handbuch durchführen.
- `user.lib`-Ordner—In diesem Ordner können Sie die von Ihnen erstellten VIs speichern, die Sie häufig verwenden. Die VIs in diesem Verzeichnis werden in der Palette **Funktionen** angezeigt.
- `Wizard`-Ordner—Dieser Ordner erstellt die Option **Solution Wizard** im Menü **Datei** (**nur PCI Macintosh**). Sie können dieses Verzeichnis auch dazu verwenden, dem Menü **Datei** Objekte hinzuzufügen.

Das Installationsprogramm von LabVIEW installiert außerdem mehrere Treiberdateien, die Ihnen den Einsatz von GPIB- und/oder DAQ-Einsteckkarten ermöglichen.

- `Systemordner:Kontrollfelder:NI-488 INIT`—Dieses Kontrollfeld enthält die Treiber für die GPIB-Karten. Sie können die Karten über das Kontrollfeld konfigurieren. Es ist jedoch selten notwendig, die Einstellungen zu ändern.
- `Systemordner:Kontrollfelder:NI-DAQ`—Dieses Kontrollfeld lädt die DAQ-Treiber in den Arbeitsspeicher. Sie können über das Kontrollfeld die Plazierung und das Verhalten der DAQ-Karten und SCXI-Module konfigurieren.
- `Systemordner:Systemerweiterungen:NI-DMA/DSP`—Diese Erweiterung wird sowohl von den GPIB- als auch den DAQ-Treibern verwendet. Sie unterstützt den DMA-Transfer (Direct Memory Access/Direkter Speicherzugriff) von Daten, wodurch höhere Datenübertragungsraten möglich sind. Diese Erweiterung unterstützt außerdem die NI-DSP-Karten.

LabVIEW installiert Treiber-Software für GPIB- und Datenerfassungs-Hardware. Konfigurationshinweise finden Sie in Kapitel 2, *Installation und Konfiguration Ihrer Datenerfassungs-Hardware*, in *LabVIEW Grundlagen der Datenerfassung*.

Organisation des LabVIEW-Systems (UNIX)

Nachdem Sie die Installation entsprechend der Anleitung in den *LabVIEW Versionshinweise*, die der Software beiliegen, durchgeführt haben, sollte das LabVIEW-Verzeichnis die folgenden Dateien enthalten.

- `labview` — Hierbei handelt es sich um die eigentliche LabVIEW-Programmdatei. Wenn Sie diese Datei aufrufen, wird LabVIEW gestartet.
- `vi.lib`-Verzeichnis—Enthält Bibliotheken der VIs, die mit LabVIEW geliefert werden, z.B. VIs für GPIB, die Analyse und die Datenerfassung (DAQ). Auf die meisten dieser VIs haben Sie Zugriff über die Palette **Funktionen**.
- `examples`-Verzeichnis—Enthält zahlreiche Unterverzeichnisse mit Beispielen. In diesem Verzeichnis befindet sich auch ein VI mit der Bezeichnung `readme.vi`, das als Anleitung für die Beispiele dient.
- `serpdrv`—Diese Datei ist ein Bestandteil der Schnittstelle von LabVIEW für die Kommunikation mit dem seriellen Anschluß. Diese Datei muß sich im selben Verzeichnis befinden wie `vi.lib`.
- `resource`-Verzeichnis
  - `labview.rsc`, `lvstring.rsc` und `lvicon.rsc` — Datendateien, die von der LabVIEW-Anwendung benutzt werden.
  - `lvjpeg.lib` und `lvpng.lib` — Diese Dateien unterstützen das Anzeigen von JPEG- und PNG-Grafiken in HTML-Dateien, wenn Sie VI-Dokumentationstexte in eine HTML-Datei drucken.
- `cintools`-Verzeichnis—Enthält Dateien, die zum Erstellen von Code Interface Nodes (CINs) benötigt werden. Mit CINs kann C-Code mit VIs von LabVIEW verbunden werden.
- `visarc`-Datei—Diese Datei ist ein Bestandteil der Schnittstelle von LabVIEW für VISA (Virtual Instrument Software Architecture). VISA bietet eine einzelne Schnittstellenbibliothek zum Steuern von VXI-, GPIB- und seriellen Instrumenten.
- `Project`-Verzeichnis—Enthält die Dateien, die als Objekte im LabVIEW-Menü **Projekt** erscheinen.
- `menus`-Verzeichnis—Enthält Dateien, die zum Konfigurieren der Struktur der Paletten **Elemente** und **Funktionen** verwendet werden.

- instr.lib-Verzeichnis—Enthält Gerätetreiber, die zum Steuern von VXI-, GPIB- und seriellen Instrumenten verwendet werden. Sie sollten Gerätetreiber von National Instruments, die Sie installieren, in diesem Verzeichnis einrichten, da sie in die Palette **Funktionen** aufgenommen werden.
- help-Verzeichnis—Enthält vollständige Online- Dokumentationsmaterialien sowie die Hilfedatei für die Suche nach Beispielen. Diese Datei erleichtert das Suchen nach Beispielen, die Ihrer Anwendung ähneln.
- activity-Verzeichnis—In diesem Verzeichnis können Sie die VIs speichern, die Sie erstellen, wenn Sie die Übungen in diesem Handbuch durchführen.
- user.lib-Verzeichnis—In diesem Verzeichnis können Sie die von Ihnen erstellten VIs speichern, die Sie häufig verwenden. Die VIs in diesem Verzeichnis werden in der Palette **Funktionen** angezeigt.
- Wizard-Verzeichnis—Dieses Verzeichnis erstellt die Option **Solution Wizard** im Menü **Datei**. Sie können dieses Verzeichnis auch dazu verwenden, dem Menü **Datei** Objekte hinzuzufügen.
- acrobat-Verzeichnis— Enthält Online-Dokumentationstexte in Acrobat-Format (.pdf).
- acroread-Verzeichnis—Enthält Lesedateien für Adobe Acrobat.

Toolkit-Unterstützung

Dateien, die in vi.lib\addons installiert werden, erscheinen automatisch auf der höchsten Ebene der Paletten **Elemente** und **Funktionen**. Diese Funktion kann auch für neue Toolkits eingesetzt werden, um nach der Installation den Zugriff auf sie zu vereinfachen. Wenn Sie bereits über Toolkits verfügen, durch die Dateien an anderer Stelle installiert wurden, können Sie diese Dateien in das Verzeichnis addons verschieben, um leichter auf sie zugreifen zu können. Wenn Sie Ihre eigenen VIs in die Paletten aufnehmen möchten, ist es sinnvoller, sie in user.lib abzulegen oder sie einem benutzerspezifischen Palettenset hinzuzufügen.

Wo sollte ich beginnen?

Dieses Handbuch enthält grundsätzliche Informationen zum Erstellen einer Anwendung in LabVIEW. Wenn Sie einen ersten Einblick in die LabVIEW-Umgebung erhalten möchten, sollten Sie das *LabVIEW Online-Tutorial* **(nur Windows)**, die *LabVIEW Schnelleinstieg* und Teil I, *Einführung in das Programmieren in G* in diesem Handbuch durcharbeiten.

Die meisten LabVIEW-Anwendungen sind in die folgenden Tasks unterteilt: I/O-Schnittstelle zu Sensoren oder Instrumenten, Datenanzeige auf dem Frontpanel, Datenanalyse, Datenspeicherung und Datentransfer über ein Netzwerk. Informationen zu diesen Tasks finden Sie in Teil II, *I/O-Schnittstellen*, Teil III, *Analyse*, und Teil IV, *Kommunikation im Netzwerk und zwischen den Anwendungen*. Informationen zu fortgeschrittenen Programmiertechniken in G finden Sie in Teil V, *Fortgeschrittenes Programmieren in G*, in diesem Handbuch.

Über den Solution Wizard **(nur Windows und PCI Macintosh)** oder über die im Startdialogfeld von LabVIEW aufrufbaren Online-Hilfedatei für die Suche nach Beispielen **(nur Windows)** können Sie Beispiele, die Ihrer Anwendung ähneln, generieren oder ermitteln.

Information zu einzelnen Funktionen und VIs finden Sie im *LabVIEW Funktions- und VI-Referenzhandbuch* und in der Online-Hilfe.

Teil I

Einführung in das Programmieren in G

Dieser Teil des Handbuchs enthält grundsätzliche Informationen über das Erstellen von virtuellen Instrumenten (VIs), über die Verwendung von VIs in anderen VIs, über Programmierstrukturen, wie z.B. Schleifen, und über Datenstrukturen, wie z.B. Arrays und Strings.

Teil I, *Einführung in das Programmieren in G*, enthält die folgenden Kapitel:

- Kapitel 2, *VIs erstellen*, erläutert, wie ein VI mit einem Frontpanel, d.h. der eigentlichen Benutzeroberfläche, und einem Blockdiagramm, d.h. dem Quellcode, erstellt wird. Nachdem Sie ein VI erstellt haben, können Sie es in anderen VIs einsetzen.
- Kapitel 3, *Schleifen und Diagramme*, macht Sie damit vertraut, wie Sie Teile des Blockdiagramms mit Hilfe einer While-Schleife bzw. einer For-Schleife wiederholen können. In diesem Kapitel wird außerdem erklärt, wie Sie mehrere Punkte nacheinander grafisch auf einem Diagramm anzeigen können.
- Kapitel 4, *Case- und Sequenzstrukturen und der Formel-Knoten*, erläutert die Verwendung der Case-Struktur, die eine Bedingungsstruktur darstellt, der Sequenzstruktur, die zum Festlegen einer Ausführreihenfolge nützlich ist, und des Formelknotens, der das Ausführen mathematischer Formeln unterstützt.
- Kapitel 5, *Arrays, Cluster und Graphen*, beschreibt, wie eine Gruppe oder ein Array von Datenpunkten auf einem Graphen angezeigt wird. Sie können sowohl Skalierungsparameter als auch ein Array von Datenpunkten an einen Graphen übergeben, indem Sie ein Cluster erstellen, das aus einer Gruppe unterschiedlicher Datentypen besteht.

- Kapitel 6, *Strings und Datei I/O*, erläutert, wie Strings manipuliert und in eine ASCII-Datei geschrieben werden.

☞ **Hinweis** **(Windows 3.1)** ***Sie müssen die VIs, die Sie in Teil I erstellen, in VI-Bibliotheken speichern. VI-Bibliotheken ermöglichen Ihnen die Verwendung von Dateinamen, die länger als 8 Buchstaben sind. Die VIs, die für die Übungen in Teil I benötigt werden, befinden sich in der VI-Bibliothek*** `LabVIEW\Activity\Activity.llb`***. Weitere Informationen zu VI-Bibliotheken finden Sie im Abschnitt*** *Speichern von VIs* ***in Kapitel 2****, Bearbeiten von VIs****, im Referenzhandbuch zur Programmierung in G.***

2

VIs erstellen

In diesem Kapitel erhalten Sie eine Einführung in die Konzepte, die der Arbeit mit virtuellen Instrumenten zugrundeliegen. In verschiedenen Übungen werden die folgenden Aufgaben näher erläutert:

- Icon und Anschluß erstellen
- VIs als SubVIs verwenden

Was ist ein virtuelles Instrument?

Ein virtuelles Instrument (VI) ist ein Programm in der grafischen Programmiersprache G. Die Frontpanels von virtuellen Instrumenten weisen oft eine Benutzeroberfläche auf, die realen Instrumenten oder Geräten ähneln. G bietet außerdem integrierte Funktionen, die Ähnlichkeit mit VIs haben, aber nicht mit Frontpanels oder Blockdiagrammen wie VIs ausgestattet sind. Funktionssymbole erscheinen immer vor einem gelben Hintergrund.

Wie wird ein VI erstellt?

Ein wichtiger Aspekt beim Erstellen von LabVIEW-Anwendungen besteht darin, die hierarchische Struktur von VIs zu verstehen und sinnvoll anzuwenden. Nachdem Sie ein VI erstellt haben, können Sie es als SubVI im Blockdiagramm eines übergeordneten VIs verwenden.

Hierarchie der VIs

Die Arbeit an einer Anwendung beginnt mit dem Top-Level-VI. Dort legen Sie zunächst die Eingänge und Ausgänge für die Anwendung fest. Danach stellen Sie die SubVIs zusammen, die die notwendigen Bearbeitungsvorgänge durchführen, während die Daten durch das Blockdiagramm fließen. Wenn ein Blockdiagramm eine große Anzahl Icons aufweist, sollten Sie sie in einem untergeordneten VI zusammenfassen, damit das Blockdiagramm möglichst einfach gestaltet ist. Dieser modulare Ansatz erleichtert das Debugging von Anwendungen und trägt zur Übersichtlichkeit und leichteren Verwaltung der Anwendungen bei.

Wie bei anderen Anwendungen können Sie das VI in eine Datei in einem regulären Verzeichnis speichern. Mit G können Sie außerdem mehrere VIs in einer einzigen Datei speichern, die als VI-Bibliothek bezeichnet wird.

Wenn Sie mit Windows 3.1 arbeiten, sollten Sie Ihre VIs in VI-Bibliotheken speichern, da Sie auf diese Weise lange Dateinamen (mit bis zu 255 Zeichen) mit Groß- und Kleinbuchstaben verwenden können.

Sie sollten nur dann VI-Bibliotheken verwenden, wenn Sie die VIs in Windows 3.1 übertragen müssen. Das Speichern von VIs in einzelnen Dateien ist effektiver, als VI-Bibliotheken zu verwenden, da die Dateien einfacher kopiert, umbenannt und gelöscht werden können als bei Verwendung einer VI-Bibliothek. Sie finden eine Liste mit den Vor- und Nachteilen von VI-Bibliotheken und Einzeldateien im Abschnitt *Speichern von VIs* in Kapitel 2, *Bearbeiten von VIs*, im *Referenzhandbuch zur Programmierung in G*.

Das Laden, Speichern und Öffnen von Dateien in VI-Bibliotheken ist auf dieselbe Weise möglich wie in anderen Verzeichnissen. VI-Bibliotheken sind jedoch nicht hierarchisch strukturiert, d.h., Sie können eine VI-Bibliothek nicht einer anderen VI-Bibliothek unterordnen. In einer VI-Bibliothek ist es ebenfalls nicht möglich, ein neues Verzeichnis zu erstellen. Außerhalb der LabVIEW-Umgebung besteht keine Möglichkeit, die in einer VI-Bibliothek enthaltenen VIs aufzulisten.

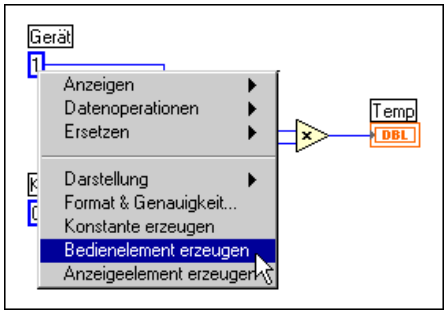
Nachdem Sie eine VI-Bibliothek erstellt haben, erscheint sie im LabVIEW-Dateidialogfeld als Ordner mit der Bezeichnung **VI** auf dem Ordnersymbol. Reguläre Verzeichnisse werden als Ordner ohne VI-Beschriftung angezeigt.

Auch wenn Sie Ihre eigenen VIs nicht in VI-Bibliotheken zu speichern beabsichtigen, sollten Sie sich damit vertraut machen, wie VI-Bibliotheken eingesetzt werden. Im Verlauf der verschiedenen Übungen in diesem Handbuch werden Sie gebeten, Ihre VIs im Verzeichnis `LabVIEW\Activity` zu speichern. Die Lösungen für diese Übungen befinden sich im Verzeichnis `LabVIEW\Activity\Solution`.

Bedienelemente, Konstanten und Anzeigeelemente

Ein Bedienelement ist ein Objekt, das Sie auf dem Frontpanel plazieren, um Daten interaktiv in ein VI oder programmatisch in ein SubVI einzugeben. Ein Anzeigeelement ist ein Objekt, das Sie zur Datenausgabe auf dem Frontpanel plazieren. Die Bedien- und Anzeigeelemente in G sind vergleichbar mit den Eingabe- und Ausgabeparametern in traditionellen Programmiersprachen. Anstatt solche Bedien- und Anzeigeelemente auf

dem Frontpanel einzurichten und sie dann mit Funktionen oder VIs auf einem Blockdiagramm zu verbinden, können Sie die Bedien- oder Anzeigeelemente auch direkt vom Blockdiagramm aus erstellen. Rufen Sie dazu das *Popup*-Menü auf dem Eingangsterminal einer Funktion oder eines VIs auf dem Blockdiagramm auf, und wählen Sie **Bedienelement erzeugen**. Auf diese Weise wird ein Bedienelement für den richtigen Datentyp erstellt und mit dem Terminal verbunden.



Sie können ein Anzeigeelement erstellen und es mit einem Ausgangsterminal verbinden, indem Sie das Popup-Menü auf dem Terminal aufrufen und **Anzeigeelement erzeugen** wählen. Anstatt Konstanten auf dem Blockdiagramm zu plazieren und sie mit Funktionen und VIs zu verbinden, können Sie das Popup-Menü für ein Funktions- oder VI-Terminal aufrufen und **Konstante erzeugen** wählen. Vom Blockdiagramm aus ist es nicht möglich, ein Bedien- oder Anzeigeelement zu löschen. Wie bei allen anderen Objekten des Frontpanels müssen Sie zum Frontpanel wechseln, das Positionierungswerkzeug wählen und dann das Objekt löschen.

Jedesmal, wenn Sie ein neues Bedien- oder Anzeigeelement auf dem Frontpanel erstellen, richtet LabVIEW das entsprechende Terminal im Blockdiagramm ein. Die Terminalsymbole geben Hinweise auf den Datentyp des Bedien- oder Anzeigeelements. Ein DBL-Terminal steht z.B. für eine Fließkommazahl mit Double-Präzision; ein TF-Terminal steht für einen Booleschen Wert; ein I16-Terminal entspricht einem regulären 16-Bit Integer-Wert; und ein ABC-Terminal entspricht einem String. Weitere Informationen über die Datentypen in G und ihre grafische Darstellung finden Sie auf der Karte *G Kurzüberblick*.

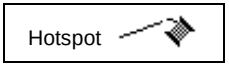
Terminals

Terminals stellen Bereiche in einem VI oder einer Funktion dar, durch die Daten fließen. Terminals entsprechen den Parametern in Programmiersprachen auf Textbasis. Es ist wichtig, daß Sie die richtigen Terminals einer Funktion oder eines VIs miteinander verbinden. Sie können den Icon-Anschluß anzeigen, um das Herstellen einer richtigen Verbindung zu erleichtern. Rufen Sie dazu das Popup-Menü für eine Funktion oder ein VI auf, und wählen Sie **Anzeigen»Anschlüsse**. Sie können wieder zum Icon zurückkehren, indem Sie das Popup-Menü für die Funktion oder das VI erneut aufrufen und noch einmal **Anzeigen» Anschlüsse** wählen.

Verbindungen

Eine *Verbindung* stellt einen Datenpfad zwischen Knoten dar. Die Verbindungen werden je nach Datentyp, der über die Verbindung übertragen wird, in unterschiedlichen Farben angezeigt. Blaue Verbindungen übertragen Integer-Werte, orangfarbige Verbindungen übertragen Fließkommazahlen, grüne Verbindungen übertragen Boolesche Werte und rosafarbige Verbindungen übertragen Strings. Weitere Informationen zu Verbindungstypen und -farben finden Sie auf der Karte *G Kurzüberblick*.

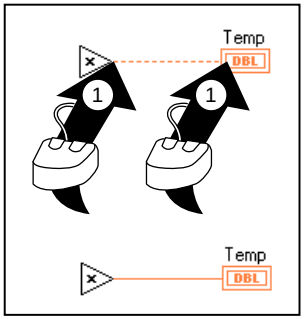
Klicken Sie mit dem Verbindungswerkzeug auf das erste Terminal, das Sie verbinden möchten, ziehen Sie dann das Werkzeug zum zweiten Terminal, und klicken Sie auf das zweite Terminal. Es spielt dabei keine Rolle, mit welchem Terminal Sie beginnen. Der Hotspot des Verbindungswerkzeugs befindet sich an der Spitze des abgewickelten Drahtsegments.



In den Abbildungen in diesem Abschnitt, die Verbindungen darstellen, zeigt der Pfeil am Ende dieses Maussymbols auf die Stelle, an der Sie klicken müssen, und die Zahl, die auf dem Pfeil zu sehen ist, gibt an, wie oft Sie mit der Maustaste klicken müssen.

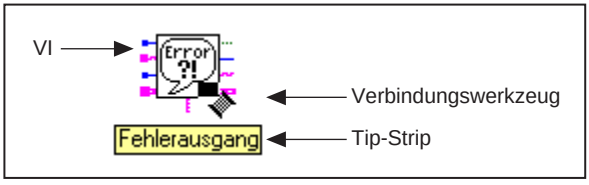
Wenn sich das Verbindungswerkzeug über einem Terminal befindet, blinkt der Terminalbereich. Dadurch wird darauf hingewiesen, daß durch Klicken die Verbindung zu diesem Terminal hergestellt werden kann. Halten Sie die Maustaste nicht gedrückt, während Sie das Verbindungswerkzeug zwischen den Terminals hin- und herziehen. Sie können eine Verbindung biegen, indem Sie die Maus im rechten Winkel zur gegenwärtigen Ausrichtung ziehen. Wenn Sie auf die Maustaste klicken, können Sie mehrere Biegestellen in einer Verbindung festlegen. Drücken Sie auf die

Leertaste, um die Ausrichtung der Verbindung zu ändern. Klicken Sie mit der Maustaste, um die Verbindung zu verankern und die Maus rechtwinklig zu verschieben.



Tip-Strips

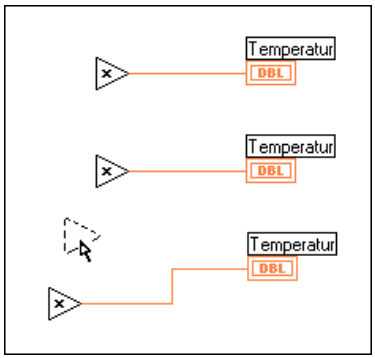
Wenn Sie das Verbindungswerkzeug über das Terminal eines Knotens ziehen, wird ein *Tip-Strip* für dieses Terminal eingeblendet. Tip-Strips bestehen aus kleinen, gelben Textstreifen, in denen der Name des jeweiligen Terminals angezeigt wird. Diese Tip-Strips sind beim Herstellen von Verbindungen zwischen den Terminals sehr nützlich. Die folgende Abbildung zeigt den Tip-Strip, der eingeblendet wird, wenn Sie das Verbindungswerkzeug über einen Ausgang des VIs “Einfacher Error-Handler” ziehen.



Hinweis ***Wenn Sie das Verbindungswerkzeug über einen Knoten ziehen, zeigt G für jeden Ein- und Ausgang eine Verdickung an. Am Ende dieser Verdickung befindet sich ein Punkt, falls es sich um einen Eingang für den Knoten handelt.***

Verbindungen strecken

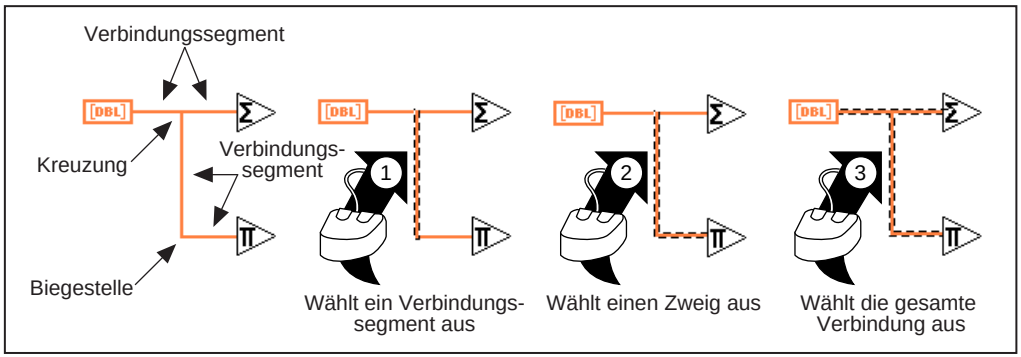
Sie können miteinander verbundene Objekte einzeln oder in Gruppen verschieben. Dazu müssen Sie lediglich die ausgewählten Objekte mit dem Positionierwerkzeug an eine neue Stelle ziehen.



Verbindungen auswählen und löschen

Es kann vorkommen, daß Sie Knoten falsch miteinander verbinden. Wählen Sie in diesem Fall die Verbindung aus, die Sie löschen möchten, und drücken Sie auf <Entf>. Ein Verbindungssegment stellt ein einzelnes horizontales oder vertikales Verbindungsstück dar. Der Punkt, an dem drei oder vier Verbindungssegmente aufeinandertreffen, wird als *Kreuzung* bezeichnet. Ein Verbindungszweig enthält alle Verbindungssegmente, die von einer Kreuzung zu einer anderen Kreuzung, von einem Terminal zur nächsten Kreuzung oder von einem Terminal zu einem anderen Terminal verlaufen, sofern dazwischen keine Kreuzungen vorhanden sind. Sie können ein Verbindungssegment durch Klicken mit dem

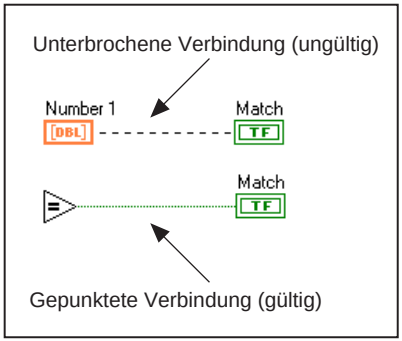
Positionierwerkzeug auswählen. Durch Doppelklicken wird ein Zweig ausgewählt, und durch dreifaches Klicken wird die gesamte Verbindung ausgewählt.

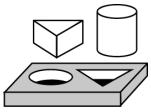


Ungültige Verbindungen

Eine unterbrochene Linie weist darauf hin, daß es sich um eine ungültige Verbindung handelt. Eine ungültige Verbindung kann aus verschiedenen Gründen entstehen, so z.B. wenn Sie zwei Bedienelemente miteinander verbinden oder wenn Sie ein Quellterminal mit einem Zielterminal verbinden, obwohl die Datentypen nicht übereinstimmen (z.B. bei der Verbindung von numerischen und Booleschen Werten). Sie können eine ungültige Verbindung entfernen, indem Sie die Verbindung mit dem Positionierwerkzeug anklicken und dann auf <Entf> drücken. Wenn Sie **Bearbeiten»Ungültige Verbindungen entfernen** oder <Strg-B> wählen, werden alle ungültigen Verbindungen in einem Blockdiagramm gelöscht. Es handelt sich dabei um ein nützliches Mittel, das Sie ausprobieren sollten, wenn ein VI nicht arbeitet oder wenn die Fehlermeldung **`Signal hat offene Verbindungen`** (“Signal has Loose Ends”) ausgegeben wird.

Hinweis ***Achten Sie darauf, daß Sie eine schwarze unterbrochene Verbindung nicht mit einer gepunkteten Verbindung verwechseln. Wie in der folgenden Abbildung zu sehen ist, weist eine gepunktete Verbindung auf einen Booleschen Datentyp hin.***





Übung 2-1. VI erstellen

Übungsziel ist das Erstellen eines VIs.

Stellen Sie sich vor, Sie setzen Sensoren ein, die Temperatur- und Volumenangaben als Spannungen erfassen. Im Rahmen dieser Übung werden Sie ein VI im Verzeichnis `LabVIEW\Activity` verwenden, um Temperatur- und Volumenmessungen in Volt-Werten zu simulieren. Sie stellen zu diesem Zweck ein VI zusammen, durch das diese Messungen als Grad- (Fahrenheit) und Literangaben wiedergegeben werden.

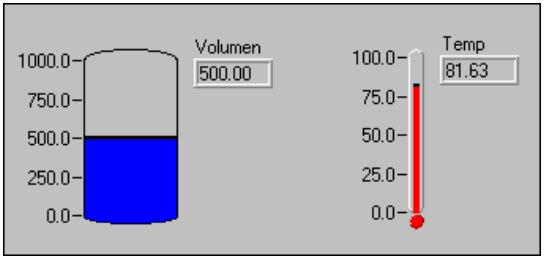
1. Öffnen Sie ein neues Frontpanel, indem Sie **Datei»Neu** wählen. Falls Sie alle VIs geschlossen haben, sollten Sie **Neues VI** im Dialogfeld LabVIEW wählen.

Hinweis ***Wenn die Palette* Elemente *nicht sichtbar ist, sollten Sie* Fenster» Elementepalette anzeigen *wählen, um die Palette anzuzeigen. Sie können auf die Palette* Elemente *auch zugreifen, indem Sie das Popup-Menü in einem offenen Bereich des Frontpanels aufrufen. Klicken Sie mit der rechten Maustaste, um das Popup-Menü aufzurufen (für Macintosh: Taste <Option> gedrückt halten).***

2. Wählen Sie **Tank** unter **Elemente»Numerisch**, und plazieren Sie das Objekt auf dem Frontpanel.
3. Geben Sie `Volumen` in das Beschriftungsfeld ein, und klicken Sie an einer beliebigen Stelle auf das Frontpanel.

Hinweis ***Wenn Sie außerhalb des Textfelds klicken, ohne Text einzugeben, verschwindet das Label wieder. Wenn Sie das Label wieder anzeigen möchten, brauchen Sie nur das Popup-Menü für das Bedienelement aufzurufen und* Anzeigen»Label *zu wählen.***

4. Verändern Sie die Skala für die Tankanzeige, damit ein Tankvolumen zwischen 0,0 und 1000,0 angezeigt wird.
   a. Doppelklicken Sie mit dem Beschriftungswerkzeug auf der Tankskala auf `10,0`, um diese Angabe hervorzuheben.
   b. Geben Sie `1000` in die Skala ein, und klicken Sie mit der Maustaste an einer beliebigen Stelle auf dem Frontpanel. Die dazwischenliegenden Stufen werden automatisch skaliert.
5. Plazieren Sie über **Elemente»Numerisch** ein Thermometer auf dem Frontpanel. Beschriften Sie es mit `Temp`, und legen Sie eine Skala zwischen 0 und 100 fest.
6. Das Frontpanel sollte der folgenden Abbildung ähneln.



7. Öffnen Sie das Blockdiagramm, indem Sie **Fenster»Diagramm** wählen. Wählen Sie aus der Palette **Funktionen** die unten aufgeführten Objekte aus, und plazieren Sie sie auf dem Blockdiagramm.

Hinweis ***Wenn die Palette* Funktionen *nicht sichtbar ist, sollten Sie* Fenster»Funktionspalette *wählen, um die Palette anzuzeigen. Sie können auf die Palette* Funktionen *auch zugreifen, indem Sie das Popup-Menü in einem offenen Bereich des Blockdiagramms aufrufen.***

8. Plazieren Sie die folgenden Objekte auf dem Blockdiagramm.

Prozeßmonitor (**Funktionen»VI auswählen** im Verzeichnis `LabVIEW\Activity`) — Simuliert das Ablesen einer Temperaturspannung und eines Volumenwerts von einem Sensor oder Umwandler.

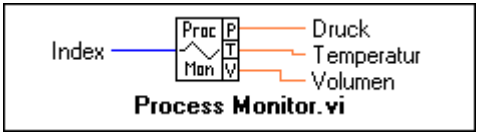
Zufallszahl-Generator (**Funktionen»Numerisch**)—Generiert eine Zahl zwischen 0 und 1.

Multiplikationsfunktion (**Funktionen»Numerisch**)—Multipliziert zwei Zahlen miteinander und gibt das Produkt aus. Im Rahmen dieser Übung benötigen Sie zwei Objekte dieser Art. Entnehmen Sie diese Funktion der Palette, und erstellen Sie dann von ihr eine Kopie.

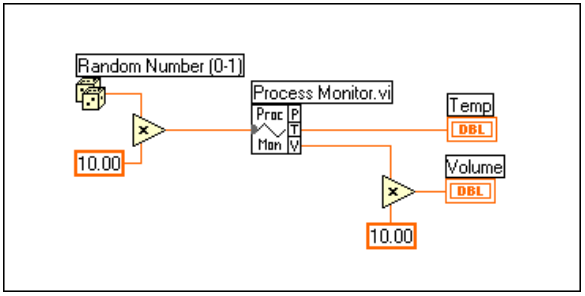
Numerische Konstante (**Funktionen»Numerisch**)—Sie benötigen zwei Objekte dieser Art. Entnehmen Sie diese Funktion der Palette. Verändern Sie den Wert mit Hilfe des Beschriftungswerkzeugs auf `10,00`. Erstellen Sie dann eine Kopie.

Hinweis ***Eine andere Möglichkeit, eine Konstante zu erstellen, besteht darin, mit dem Verbindungswerkzeug das Popup-Menü auf dem Terminal für eine Funktion oder ein VI aufzurufen. Wählen Sie* Konstante erzeugen *in dem kontextsensitiven Menü. Daraufhin erscheint eine Konstante des entsprechenden Datentyps.***

9. Wählen Sie **Hilfe anzeigen** im Menü **Hilfe**, und ziehen Sie den Cursor über die einzelnen Funktionen und VIs, um die Eingänge und Ausgänge einer Funktion oder eines VIs zu sehen. Die folgende Abbildung zeigt das Hilfefenster für das Prozeßmonitor-VI.



10. Verbinden Sie die Objekte mit dem Verbindungswerkzeug in der nachfolgend dargestellten Weise.



Hinweis ***Klicken Sie auf das Positionierwerkzeug in der Palette* Werkzeuge, *um die Objekte auf dem Blockdiagramm zu verschieben.***

11. Wählen Sie **Datei»Speichern**, und speichern Sie das VI unter dem Namen `Temp & Vol.vi` im Verzeichnis `LabVIEW\Activity`.
12. Führen Sie das VI über das Frontpanel aus, indem Sie auf die Taste **Ausführen** klicken. Auf dem Frontpanel werden nun Werte für das Volumen und die Temperatur angezeigt.
13. Schließen Sie das VI, indem Sie **Datei»Schließen wählen.**

Ende der Übung 2-1.

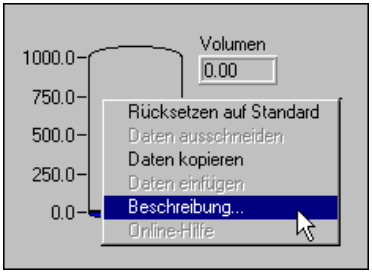
VI-Dokumentation

Über **Fenster»VI-Info…** können Sie ein VI dokumentieren. Geben Sie die Beschreibung für das VI im VI-Informationsdialogfeld ein. Sie können diese Beschreibung anschließend über **Fenster»VI-Info…** aufrufen.

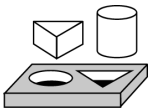
Sie können die Beschreibungen von Objekten auf dem Frontpanel (oder der jeweiligen Terminals auf dem Blockdiagramm) bearbeiten, indem Sie das Popup-Menü für das Objekt aufrufen und **Datenoperationen» Beschreibung…** wählen.

Hinweis ***Sie können die Beschreibung eines VIs oder seiner Objekte auf dem Frontpanel nicht ändern, während das VI ausgeführt wird.***

Die folgende Abbildung zeigt ein Beispiel für das Popup-Menü, das erscheint, während Sie ein VI ausführen. Sie können die Beschreibung nicht ergänzen oder ändern, während das VI ausgeführt wird. Sie können jedoch die bereits eingegebenen Informationen einsehen.



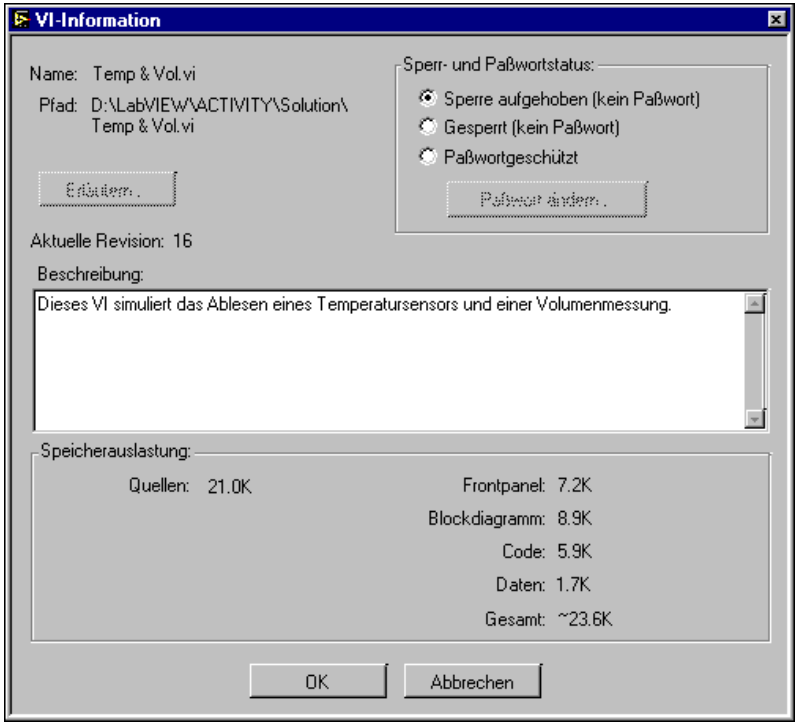
Sie können die Beschreibung eines Objekts auf dem Frontpanel auch einsehen, indem Sie das Hilfefenster (**Hilfe»Hilfe anzeigen**) öffnen und den Cursor über das Objekt ziehen.



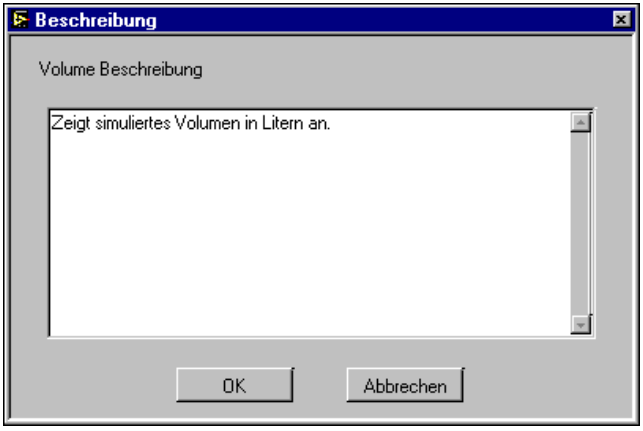
Übung 2-2. VI dokumentieren

Übungsziel ist das Dokumentieren eines VIs, das Sie erstellt haben.

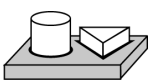
1. Öffnen Sie das VI `Temp & Vol.vi`, das Sie in Übung 2-1 erstellt und im Verzeichnis `LabVIEW\Activity` gespeichert haben.
2. Wählen Sie **Fenster»VI-Info…**. Geben Sie eine Beschreibung für das VI wie in der folgenden Abbildung ein. Klicken Sie dann auf **OK.**



3. Rufen Sie auf dem Tank das Popup-Menü auf, und wählen Sie **Datenoperationen»Beschreibung…**. Geben Sie die Beschreibung für das Anzeigeelement wie in der folgenden Abbildung ein. Klicken Sie dann auf **OK**.



4. Rufen Sie das Popup-Menü auf dem Thermometer auf, und wählen Sie **Datenoperationen»Beschreibung…**. Geben Sie die folgende Beschreibung ein: `Zeigt simulierte Temperaturmessungen (° F) an.` Klicken Sie auf **OK**.
5. Wählen Sie **Hilfe anzeigen** im Menü **Hilfe**. Ziehen Sie den Cursor zuerst auf das Volumen und dann auf die Temperatur. Die Beschreibungen, die Sie eingegeben haben, erscheinen nun im Hilfefenster.
6. Speichern Sie das VI, und schließen Sie es.



Ende der Übung 2-2.

Was ist ein SubVI?

Ein *SubVI* entspricht im großen und ganzen einer Subroutine in einer Programmiersprache auf Textbasis. Es stellt ein VI dar, das im Blockdiagramm eines anderen VIs verwendet wird.

Sie können jedes VI, das ein Icon und einen Anschluß aufweist, als SubVI in einem anderen VI verwenden. Im Blockdiagramm können Sie über **Funktionen»VI auswählen…** die VIs auswählen, die als SubVIs verwendet werden sollen. Bei Wahl dieser Option wird ein Dateidialogfeld eingeblendet, in dem Sie jedes beliebige VI im System auswählen können. Wenn Sie ein VI öffnen, das kein Icon und keinen Anschluß besitzt, erscheint im Blockdiagramm des aufrufenden VIs ein leeres, quadratisches Feld. Sie können keine Verbindung zu diesem Knoten herstellen. Weitere Informationen zu Icons und Anschlüssen finden Sie im *LabVIEW Online-Tutorial*, auf das Sie vom Startdialogfeld aus zugreifen können.

Ein SubVI entspricht einer Subroutine. Ein SubVI-Knoten entspricht einem Subroutinen-Aufruf. Der SubVI-Knoten ist ebensowenig mit dem SubVI identisch, wie ein Subroutinen-Aufruf in einem Programm mit der eigentlichen Subroutine identisch ist. Ein Blockdiagramm, das mehrere identische SubVI-Knoten enthält, ruft dasselbe SubVI mehrmals auf.

Hierarchiefenster

Das Hierarchiefenster zeigt in einer grafischen Darstellung die Aufrufhierarchie für alle im Speicher vorhandenen VIs an, einschließlich der Typdefinitionen und der globalen Variablen. Im Hierarchiefenster (**Projekt»VI-Hierarchie anzeigen**) können die Abhängigkeiten der VIs untereinander durch Angabe von Informationen über VI-Aufrufe und SubVIs angezeigt werden. Dieses Fenster enthält eine Symbolleiste, mit der Sie für die angezeigten Elemente mehrere Einstellungstypen konfigurieren können. Die folgende Abbildung zeigt ein Beispiel für die VI-Hierarchiesymbolleiste.

Sie können auf die folgenden Optionen zugreifen, indem Sie auf die Tasten auf der Symbolleiste im Hierarchiefenster klicken, die Optionen im Ansichtsmenü verwenden oder das Popup-Menü auf einem leeren Bereich des Fensters aufrufen. Sie finden weitere Informationen zum Hierarchiefenster im Abschnitt *Verwendung des Hierarchiefensters* in

Kapitel 3, *Verwendung von SubVIs*, im *Referenzhandbuch zur Programmierung in G*.

Erneut zeichnen—Richtet die Knoten nach wiederholter Bearbeitung der Hierarchieknoten neu aus, falls Überschneidungen reduziert und die symmetrische Ausrichtung optimiert werden soll. Wenn ein Fokus-Knoten vorhanden ist, können Sie durch das Fenster blättern, bis die erste Wurzel, die SubVIs aufweist, sichtbar ist.

Auf vertikales Layout umschalten—Richtet die Knoten von oben nach unten aus, wobei Wurzeln an oberster Stelle plaziert werden.

Auf horizontales Layout umschalten—Richtet die Knoten von links nach rechts aus, wobei Wurzeln auf der linken Seite plaziert werden.

VIs aufnehmen/ausschließen — Bewirkt, daß der Hierarchiegraph VI-Bibliotheken aufnimmt bzw. VIs in VI-Bibliotheken ausschließt.

Globale Variable aufnehmen/ausschließen — Bewirkt, daß der Hierarchiegraph globale Variablen aufnimmt bzw. ausschließt. Globale Variablen speichern Daten, die von mehreren VIs verwendet werden.

Typedefs aufnehmen/ausschließen — Bewirkt, daß der Hierarchiegraph Typedefs aufnimmt bzw. ausschließt. Ein Typedef ist die Hauptkopie eines benutzerspezifischen Bedienelements, das von mehreren VIs verwendet werden kann.

Das Ansichtsmenü und die Popup-Menüs bieten zusätzlich die **Label**-Optionen **Alle VIs anzeigen** und **Gesamter VI-Pfad**, auf die Sie nicht über die Symbolleiste zugreifen können.

Wenn Sie das Bedienwerkzeug über die einzelnen Objekte im Hierarchiefenster ziehen, zeigt LabVIEW den Namen des VIs unter dem VI-Icon an.

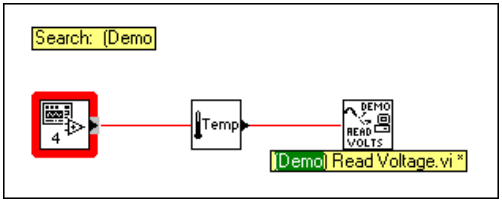
Verwenden Sie die <Tab>-Taste, um zwischen dem Positionier- und dem Blätterwerkzeug hin- und herzuschalten. Diese Funktion ist besonders nützlich, wenn Sie Knoten aus dem Hierarchiefenster in das Blockdiagramm verschieben möchten.

Durch Klicken auf den VI- oder SubVI-Knoten können Sie den Knoten in das Blockdiagramm ziehen oder in die Zwischenablage kopieren. Halten Sie die <Umschalt>-Taste gedrückt, während Sie auf einen VI- oder SubVI-Knoten klicken, wenn Sie mehrere Objekte zum Kopieren in andere Blockdiagramme oder Frontpanels auswählen möchten. Durch Doppelklicken auf einen VI- oder SubVI-Knoten wird das Frontpanel für den jeweiligen Knoten geöffnet.

Alle VIs, die SubVIs enthalten, sind durch eine Pfeiltaste gekennzeichnet, die sich neben dem jeweiligen VI befindet, das Sie zum Ein- bzw. Ausblenden der SubVIs verwenden können. Wenn Sie auf die rote Pfeiltaste klicken oder auf das VI doppelklicken, werden die SubVIs im jeweiligen VI angezeigt. Eine schwarze Pfeiltaste auf einem VI-Knoten weist darauf hin, daß alle SubVIs bereits angezeigt werden. Sie können auf einem VI- oder SubVI-Knoten auch ein Popup-Menü aufrufen, das verschiedene Optionen enthält, so z.B. zum Ein- bzw. Ausblenden von SubVIs, zum Öffnen des Frontpanels für das VI oder SubVI, zum Bearbeiten des VI-Icons usw.

Suchhierarchie

Sie können in den zur Zeit sichtbaren Knoten im Hierarchiefenster auch nach bestimmten Namen suchen. Sie leiten die Suche ein, indem Sie den Namen des Knotens an einer beliebigen Stelle des Fensters eingeben. Bei der Eingabe des Textes erscheint ein Such-String, in dem der eingegebene Text angezeigt wird, während gleichzeitig die Hierarchie durchsucht wird. Die folgende Abbildung zeigt die Suchhierarchie.

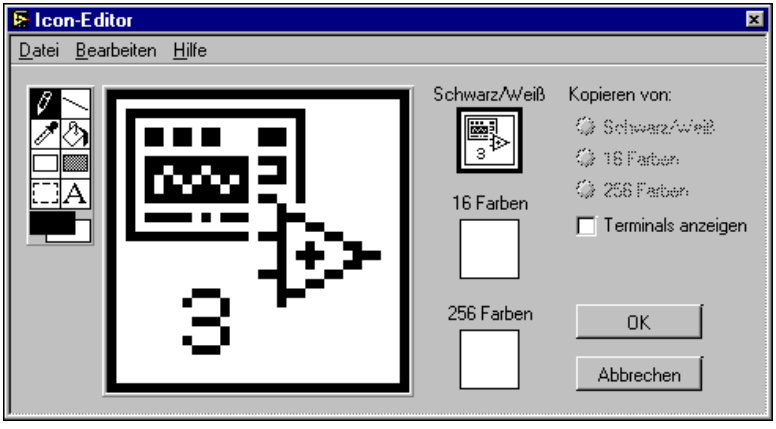


Wenn der gesuchte Knoten gefunden wurde, können Sie auf <Eingabe> drücken, um nach dem nächsten Knoten zu suchen, der mit der Suchzeichenfolge übereinstimmt. Wenn Sie auf <Umschalt-Eingabe> drücken, wird die Suche nach dem vorherigen Knoten durchgeführt, der der Suchzeichenfolge entspricht.

Icon und Anschluß

Für jedes VI wird in der oberen rechten Ecke des Frontpanel- und Diagrammfensters ein Standard-Icon angezeigt. Für VIs wird als Standard das VI-Icon von LabVIEW mit einer Zahl angezeigt, die angibt, wie viele neue VIs Sie seit dem Start von LabVIEW geöffnet haben. Mit dem Icon-Editor können Sie einzelne Pixel ein- und ausschalten, um das Icon Ihren Vorstellungen anzupassen. Um den Icon-Editor zu aktivieren, müssen Sie das Popup-Menü auf dem Standard-Icon in der oberen rechten Ecke des Panelfensters aufrufen und **Icon bearbeiten** wählen.

Die folgende Abbildung zeigt das Fenster für den Icon-Editor. Mit den Werkzeugen auf der linken Seite können Sie das Design des Icons im Pixelbearbeitungsbereich gestalten. In einem der Felder rechts neben dem Bearbeitungsbereich wird das Icon in seiner tatsächlichen Größe dargestellt.



Mit den Werkzeugen links neben dem Bearbeitungsbereich können die folgenden Funktionen ausgeführt werden:

Bleistift — Ermöglicht das Zeichnen und Löschen einzelner Pixel.

Linien — Ermöglicht das Zeichnen gerader Linien. Drücken Sie die <Umschalt>-Taste, und ziehen Sie horizontale, vertikale und diagonale Linien mit diesem Werkzeug.

Farbe kopieren — Dient zum Kopieren der Vordergrundfarbe eines Elements im Icon.

Fläche füllen — Dient zum Ausfüllen eines umgrenzten Bereichs mit der Vordergrundfarbe.

Rechtecke — Ermöglicht das Zeichnen eines rechteckigen Rahmens in der Vordergrundfarbe. Doppelklicken Sie auf dieses Werkzeug, um das Icon mit einem Rahmen in der Vordergrundfarbe zu umgeben.

Ausgefüllte Rechtecke — Ermöglicht das Zeichnen eines Rechtecks mit Rändern in der Vordergrundfarbe und einer ausgefüllten Innenfläche in der Hintergrundfarbe. Doppelklicken Sie auf dieses Werkzeug, um das Icon mit einem Rahmen in der Vordergrundfarbe zu umgeben und mit der Hintergrundfarbe auszufüllen.

Auswahl — Dient zur Auswahl eines Bereichs des Icons, der verschoben, geklont oder auf andere Weise verändert werden soll.

Text — Dient zur Eingabe von Text in das Icon-Design.

Vordergrund/Hintergrund — Zeigt die aktuellen Vordergrund- und Hintergrundfarben an. Klicken Sie auf eines der beiden Symbole, um eine Farbpalette anzuzeigen, in der Sie neue Farben auswählen können.

Mit den Tasten auf der rechten Seite des Bearbeitungsfensters können die folgenden Funktionen ausgeführt werden:

- **Rückgängig** — Macht den zuletzt ausgeführten Vorgang wieder rückgängig.
- **OK** — Speichert die Zeichnung als VI-Icon und wechselt zum Frontpanel zurück.
- **Abbrechen** — Wechselt zum Frontpanel, ohne die Änderungen zu speichern.

Je nach verwendetem Monitortyp können Sie ein separates Icon für den monochromen, 16-Farben- und 256-Farben-Modus entwerfen. Jedes Icon-Design wird separat gespeichert. Die Standardeinstellung für den Editor ist **Schwarz/Weiß**; durch Klicken auf eine der Farboptionen können Sie jedoch in einen anderen Modus umschalten.

☞ **Hinweis** ***Wenn Sie lediglich ein farbiges Icon entwerfen und das VI im Verzeichnis*** `*.lib` ***ablegen, erscheint das Icon nicht in einer Subpalette der Palette*** **Funktionen.** ***Darüber hinaus kann das Icon weder ausgedruckt noch auf einem Schwarzweißmonitor angezeigt werden.***

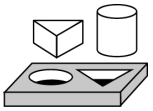
Der *Anschluß* ist die programmatische Schnittstelle zu einem VI. Wenn Sie die Bedien- oder Anzeigeelemente des Panels für die Übertragung von Daten zu und von den SubVIs verwenden, ist es notwendig, für diese Bedien- oder Anzeigeelemente Terminals auf dem Anschlußfeld einzurichten. Sie können die Anschlüsse definieren, indem Sie die Anzahl

der Terminals wählen, die Sie für das VI verwenden möchten, und jedem dieser Terminals ein Bedien- oder Anzeigeelement des Frontpanels zuweisen.

Wählen Sie **Anschluß anzeigen** im Popup-Menü für das Icon-Feld im Panel-Fenster, um einen Anschluß zu definieren.

Das Anschluß-Icon ersetzt das Icon in der oberen rechten Ecke des Panel-Fensters. LabVIEW wählt ein Terminalmuster aus, das Ihrem VI angemessen ist und Terminals für Bedienelemente auf der linken Seite des Anschlußfelds und Terminals für Anzeigeelemente auf der rechten Seite aufweist. Die Anzahl der ausgewählten Terminals hängt von der Anzahl der auf dem Frontpanel vorhandenen Bedien- und Anzeigeelemente ab.

Jedes Rechteck auf dem Anschluß entspricht einem Terminalbereich, und Sie können die Rechtecke entweder als Eingang oder als Ausgang für das VI verwenden. Falls notwendig, können Sie ein anderes Terminalmuster für das VI wählen. Rufen Sie dazu das Popup-Menü für das Icon auf, wählen Sie **Anschluß anzeigen**, rufen Sie das Popup-Menü noch einmal auf, und wählen Sie **Muster**.



Übung 2-3. Icon und Anschluß erstellen

Übungsziel ist das Erstellen eines Icons und Anschlusses für ein VI.

Wenn Sie ein VI als SubVI verwenden möchten, müssen Sie sowohl ein Icon erstellen, durch das das VI auf dem Blockdiagramm eines anderen VIs repräsentiert wird, als auch ein Anschlußfeld, an das Sie Ein- und Ausgänge anschließen können. LabVIEW stellt mehrere Werkzeuge bereit, mit denen Sie ein Icon für Ihre VIs erstellen und bearbeiten können.

Das Icon repräsentiert ein VI als SubVI im Blockdiagramm eines anderen VIs. Es kann sich dabei um eine verbildlichte Darstellung des vom VI erfüllten Zwecks oder um einen beschreibenden Text für das VI handeln.

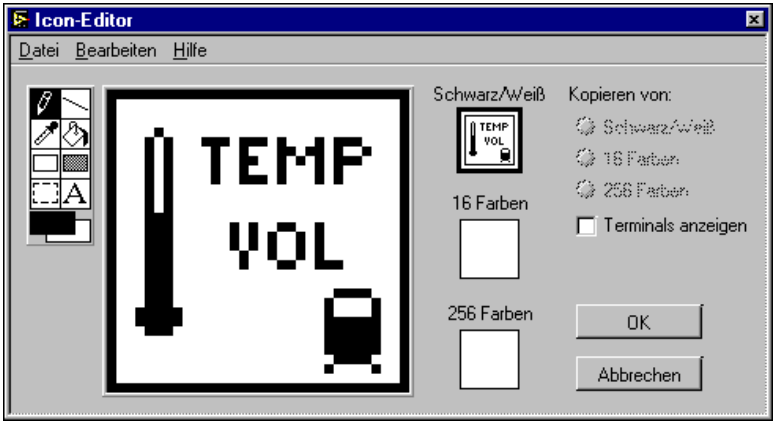
1. Öffnen Sie `Temp & Vol.vi` im Verzeichnis `LabVIEW\Activity`.
2. Rufen Sie auf dem Frontpanel das Popup-Menü für das Icon in der oberen rechten Ecke auf, und wählen Sie **Icon bearbeiten…**. Sie können den Icon-Editor auch durch Doppelklicken auf das Icon aufrufen.

☞ **Hinweis** ***Sie können nur vom Frontpanel aus auf das Icon bzw. den Anschluß für ein VI zugreifen.***

3. Löschen Sie das Standard-Icon. Klicken Sie mit dem Auswahlwerkzeug, das als gepunktetes Rechteck angezeigt wird, und ziehen Sie es über den Bereich, den Sie löschen möchten. Drücken Sie dann die <Entf>-Taste. Sie können auch auf das schattierte Rechteck im Werkzeugfeld doppelklicken, um das Icon zu löschen.

4. Zeichnen Sie mit dem Bleistift-Werkzeug ein Thermometer.

5. Benutzen Sie das Text-Werkzeug, um den Text einzugeben. Doppelklicken Sie auf das Text-Werkzeug, um die verwendete Schriftart zu ändern. Das Icon sollte der folgenden Abbildung ähneln.



6. Klicken Sie auf **OK**, um den Icon-Editor zu schließen. Das neue Icon wird im Icon-Feld angezeigt.

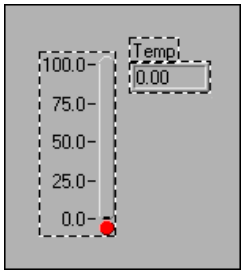
7. Definieren Sie das Anschlußterminalmuster, indem Sie im Icon-Feld auf dem Frontpanel das Popup-Menü aufrufen und **Anschluß anzeigen** wählen. LabVIEW wählt standardmäßig ein Terminalmuster aus, das der Anzahl der auf dem Frontpanel vorhandenen Bedien- und Anzeigeelemente entspricht. Da sich zwei Objekte auf dem Frontpanel befinden, hat der Anschluß wie links abgebildet zwei Terminals.

8. Rufen Sie das Popup-Menü auf dem Anschlußfeld auf, und wählen Sie **Um 90 Grad drehen**. Das Anschlußfeld ändert sich daraufhin wie links abgebildet.

9. Weisen Sie die Terminals der Temperatur und dem Volumen zu.

   a. Klicken Sie auf das obere Terminal im Anschluß. Der Cursor verwandelt sich daraufhin automatisch in das Verbindungswerkzeug, und das Terminal wird schwarz.

b. Klicken Sie auf das Temperatur-Anzeigeelement. Eine gestrichelte Linie bewegt sich nun wie in der folgenden Abbildung als Rahmen um das Anzeigeelement. Das ausgewählte Anzeigeelement erhält eine neue Farbe, die für den Datentyp des gewählten Bedien-/Anzeigeelements charakteristisch ist.



Wenn Sie in einen offenen Bereich auf dem Frontpanel klicken, verschwindet die gestrichelte Linie, und das ausgewählte Terminal erscheint abgeblendet. Dadurch wird angezeigt, daß Sie das Anzeigeelement dem Terminal zugewiesen haben. Falls das Terminal weiß ist, haben Sie den Anschluß nicht korrekt hergestellt.

c. Wiederholen Sie die Schritte a und b, um das untere Terminal mit dem Volumen-Anzeigeelement zu verbinden.

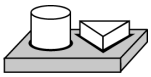
d. Rufen Sie das Popup-Menü auf dem Anschluß auf, und wählen Sie **Icon anzeigen…**.

10. Speichern Sie das VI, indem Sie **Datei»Speichern** wählen.

Dieses VI ist damit fertiggestellt und kann als SubVI in anderen VIs eingesetzt werden. Das Icon repräsentiert das VI im Blockdiagramm des aufrufenden VI. Der Anschluß (mit zwei Terminals) gibt die Temperatur und das Volumen aus.

☞ **Hinweis** ***Der Anschluß kennzeichnet die Eingänge und Ausgänge eines VIs, das als SubVI verwendet wird. Denken Sie daran, daß* Bedienelemente *auf dem Frontpanel nur als Eingänge,* Anzeigeelemente *dagegen nur als Ausgänge verwendet werden können.***

11. Schließen Sie das VI, indem Sie **Datei»Schließen** wählen.

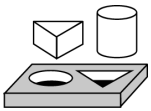


Ende der Übung 2-3.

SubVIs öffnen, bedienen und ändern

Sie können ein VI, das als SubVI eingesetzt wird, vom Blockdiagramm des aufrufenden VIs aus öffnen, indem Sie auf das Icon des SubVIs doppelklicken oder **Projekt»SubVIs des VIs** wählen. In einer Palette können Sie alle SubVIs des aufrufenden VIs sehen. Wählen Sie das SubVI aus, das Sie öffnen möchten.

Solange Sie das SubVI nicht speichern, wirken sich alle Änderungen, die Sie an einem SubVI durchführen, nur auf die Version im Arbeitsspeicher aus. Die Änderungen gelten für alle vorhandenen SubVIs dieses Typs und nicht nur für den Knoten, den Sie zur Bearbeitung des VIs verwendet haben.



Übung 2-4. SubVI aufrufen

Übungsziel ist das Erstellen eines VIs, das das VI `Temp & Vol.vi` ***als SubVI verwendet.***

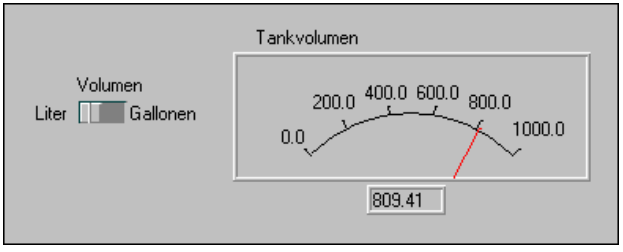
Das VI “Temp & Vol”, das Sie in Übung 2-1 angefertigt haben, gibt die Temperatur und das Volumen aus. In der folgenden Übung werden Sie ein Volumen ermitteln und den Wert durch Drücken eines Schalters in Gallonen umrechnen.

Frontpanel

1. Öffnen Sie ein neues Frontpanel, indem Sie **Datei»Neu** wählen.

2. Wählen Sie in der Palette **Elemente»Boolesch** einen horizontalen Schalter aus, und beschriften Sie ihn mit `Volumen`. Benutzen Sie das Beschriftungswerkzeug, um freie Label auf dem Frontpanel zu plazieren, die für `Liter` und `Gallonen` stehen.

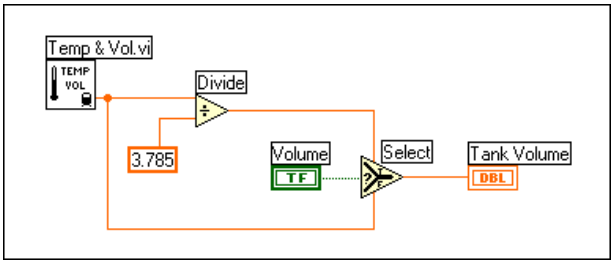
3. Wählen Sie über **Elemente»Numerisch** ein Meßgerät aus, und plazieren Sie es auf dem Frontpanel. Beschriften Sie es mit `Tankvolumen`.



4. Stellen Sie den Meßbereich des Meßgeräts so ein, daß Werte zwischen 0,0 und 1000,0 erfaßt werden. Doppelklicken Sie mit dem Bedienwerkzeug auf den Höchstwert, und ändern Sie ihn von 10,0 auf 1000,0. Wechseln Sie zum Positionierwerkzeug, und verändern Sie die Größe des Meßgeräts, indem Sie auf eine seiner Ecken klicken und daran ziehen.

Blockdiagramm

5. Wählen Sie **Fenster»Diagramm anzeigen**, um zum Blockdiagramm zu wechseln.
6. Rufen Sie das Popup-Menü in einem freien Bereich des Blockdiagramms auf, und wählen Sie **Funktionen»VI auswählen…**. Ein Dialogfeld wird angezeigt. Wählen Sie `Temp & Vol.vi` im Verzeichnis `LabVIEW\Activity`. Klicken Sie im Dialogfeld auf **Öffnen**. LabVIEW plaziert das VI “Temp & Vol” auf dem Blockdiagramm.
7. Fügen Sie die anderen Objekte wie in der folgenden Abbildung in das Blockdiagramm ein.



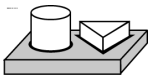
Numerische Konstante (**Funktionen»Numerisch**)—Fügen Sie eine numerische Konstante in das Blockdiagramm ein. Weisen Sie der Konstante mit dem Beschriftungswerkzeug den Wert `3,785` zu. Bei diesem Wert handelt es sich um den Umrechnungsfaktor für die Umrechnung von Litern in Gallonen.

Auswählen (**Funktionen»Vergleichsoperationen**) — Gibt je nach Booleschem Eingangswert den Wert zurück, der am Eingang WAHR oder FALSCH anliegt.

Dividieren (**Funktionen»Numerisch**)—Dividiert den in Litern gemessenen Wert durch 3,785, um ihn in Gallonen umzuwandeln.

8. Verbinden Sie die Diagrammobjekte wie in der Abbildung.

9. Kehren Sie zum Frontpanel zurück, und klicken Sie in der Symbolleiste auf die Taste **Ausführen**. Das Meßgerät zeigt den Wert in Litern an.
10. Klicken Sie auf den Schalter, um `Gallonen` auszuwählen, und klicken Sie auf die Taste **Ausführen**. Das Meßgerät zeigt den Wert in Gallonen an.
11. Speichern Sie das VI unter dem Namen `Using Temp & Vol.vi` im Verzeichnis `LabVIEW\Activity`.



Ende der Übung 2-4.

Wie wird das Debugging für ein VI durchgeführt?

Ein unterbrochenes VI kann weder kompilieren, noch kann es ausgeführt werden. In der Regel ist ein VI unterbrochen, während es erstellt oder bearbeitet wird und solange nicht alle Icons im Diagramm miteinander verbunden sind. Wenn es auch nach Fertigstellung noch unterbrochen ist, sollten Sie versuchen, das Problem durch Wahl von **Ungültige Verbindungen entfernen** im Menü **Bearbeiten** zu beheben. Oft läßt sich auf diese Weise die Funktionsfähigkeit eines unterbrochenen VIs herstellen.

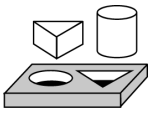
Wenn das VI nicht ausgeführt werden kann, erscheint anstelle der Taste **Ausführen** ein *unterbrochener* Pfeil. Klicken Sie auf die durch den unterbrochenen Pfeil gekennzeichnete Taste **Ausführen**, um eine Liste mit den Fehlern anzuzeigen. Klicken Sie auf einen der aufgeführten Fehler, und klicken Sie dann auf **Suchen**, um das Objekt oder Terminal, von dem der Fehler gemeldet wird, zu markieren.

Sie können die Ausführung des VI-Blockdiagramms animieren, indem Sie auf die Taste **Highlight-Funktion** klicken. Die Highlight-Funktion wird häufig im Zusammenhang mit dem Einzelschrittmodus verwendet, um den Datenfluß in einem Blockdiagramm zu verfolgen.

Beim Debugging ist es sinnvoll, ein Blockdiagramm Knoten für Knoten auszuführen. Dieses Verfahren wird als Einzelschrittausführung bezeichnet. Klicken Sie auf die Taste **Hineinspringen** oder **Überspringen**, um den Einzelschrittmodus zu aktivieren. Der erste Knoten beginnt daraufhin zu blinken. Dadurch wird angezeigt, daß die Ausführung möglich ist. Sie können dann noch einmal auf die Taste **Hineinspringen** oder **Überspringen** klicken, um den Knoten auszuführen und zum nächsten Knoten zu wechseln. Falls es sich bei dem Knoten um eine

Struktur oder um ein VI handelt, können Sie die Taste **Überspringen** wählen, um den Knoten *auszuführen*, ohne seine Ausführung in *Einzelschritten* zu verfolgen. Wenn der Knoten z.B. ein SubVI darstellt und Sie auf die Taste **Überspringen** klicken, wird das SubVI ausgeführt und der nächste Knoten angesprungen, ohne daß Sie die Ausführung der SubVI-Knoten verfolgen können. Wenn Sie eine Struktur oder ein SubVI in Einzelschritten verfolgen möchten, müssen Sie die Taste **Hineinspringen** wählen.

Klicken Sie auf die Taste **Herausspringen**, um die Ausführung der Blockdiagrammknoten zu beenden und/oder die Einzelschritte zu vervollständigen. Weitere Informationen zum Debugging finden Sie in Kapitel 4, *Ausführung und Debugging von VIs und SubVIs*, im *Referenzhandbuch zur Programmierung in G*.

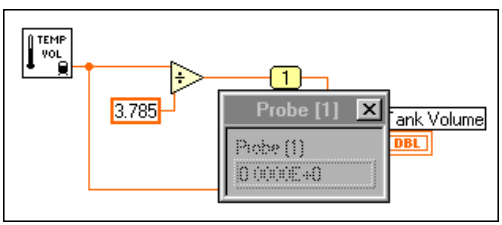


Übung 2-5. Debugging eines VIs in LabVIEW

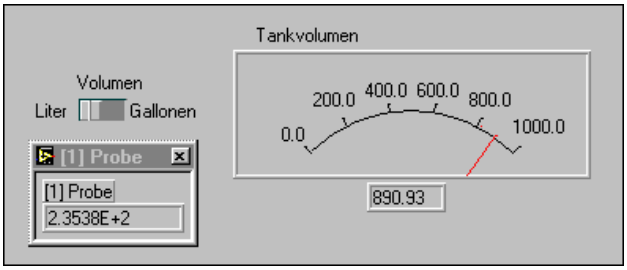
Übungsziel ist das Verwenden des Probenwerkzeugs und des Probenfensters und das Überprüfen des Datenflusses im Blockdiagramm mit Hilfe der Highlight-Funktion.

1. Öffnen Sie `Using Temp & Vol.vi` im Verzeichnis `LabVIEW\Activity`.
2. Wählen Sie **Fenster»Diagramm**.
3. Wählen Sie **Fenster»Werkzeugpalette**, falls die Werkzeugpalette nicht bereits geöffnet ist.

4. Wählen Sie das Probenwerkzeug in der Palette **Werkzeuge**. Klicken Sie mit dem Probenwerkzeug auf die Verbindung zur Divisionsfunktion. Daraufhin wird wie in der folgenden Abbildung ein Probenfenster mit dem Titel `Probe 1` und einer gelben Ziffer, die die Probennummer angibt, eingeblendet. Das Probenfenster bleibt auch dann geöffnet, wenn Sie zum Frontpanel wechseln.



5. Kehren Sie zum Frontpanel zurück. Verschieben Sie das Probenfenster, so daß Sie wie in der folgenden Abbildung sowohl die Proben- als auch die Volumenwerte sehen können. Führen Sie das VI aus. Das Volumen wird im Probenfenster in Gallonen angezeigt, während das `Tankvolumen` in Litern wiedergegeben wird.



☞ **Hinweis** ***Die Volumenwerte, die auf Ihrem Bildschirm angezeigt werden, können von den Werten in dieser Abbildung abweichen. Weitere Informationen finden Sie im Abschnitt*** *Numerische Konvertierung* ***in*** *Kapitel 3, Schleifen und Diagramme****.***

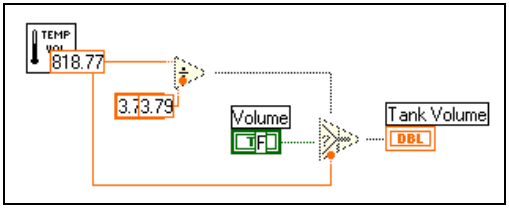
6. Schließen Sie das Probenfenster, indem Sie in das Schließfeld in der Titelleiste des Probenfensters klicken.

Eine weitere nützliche Debugging-Methode besteht darin, den Datenfluß im Blockdiagramm mit Hilfe der Highlight-Funktion zu überprüfen.

7. Kehren Sie zum Blockdiagramm des VIs zurück.

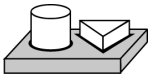
8. Beginnen Sie mit dem Markieren der Ausführung, indem Sie auf die Taste **Highlight-Funktion** in der Symbolleiste klicken. Die Taste **Highlight-Funktion** verändert sich daraufhin in eine leuchtende Glühlampe.
9. Klicken Sie auf die Taste **Ausführen**, um das VI auszuführen. Sie können nun sehen, wie die Ausführung des VI-Blockdiagramms durch die Highlight-Funktion animiert wird. Der Datenfluß durch das VI wird durch sich bewegende Blasen symbolisiert. Beachten Sie auch, daß auf den Verbindungen Datenwerte erscheinen, durch die die zur Zeit in der Verbindung vorliegenden Werte auf dieselbe Weise

angezeigt werden wie bei Durchführung einer Probe für die Verbindung. Das folgende Blockdiagramm verdeutlicht diesen Vorgang.



Sie können auch die Tasten für die Einzelschrittausführung verwenden, wenn Sie den grafischen Code Schritt für Schritt einsehen möchten.

10. Beginnen Sie mit der Einzelschrittausführung, indem Sie in der Symbolleiste auf **Überspringen** klicken.
11. Springen Sie in das SubVI “Temp & Vol”, indem Sie in der Symbolleiste auf die Taste **Hineinspringen** klicken. Durch Klicken auf diese Taste wird das Frontpanel und das Blockdiagramm des SubVIs “Temp & Vol” geöffnet. Klicken Sie wiederholt auf die Taste **Überspringen**, bis die Ausführung des VIs abgeschlossen ist.
12. Beenden Sie die Ausführung des Blockdiagramms, indem Sie in der Symbolleiste auf die Taste **Herausspringen** klicken. Durch Klicken auf diese Taste werden alle verbleibenden Sequenzen im Blockdiagramm zu Ende ausgeführt.



Ende der Übung 2-5.

3

Schleifen und Diagramme

In diesem Kapitel werden Strukturen eingeführt und die Grundkonzepte von Diagrammen, der While-Schleife und der For-Schleife erklärt. Außerdem enthält dieses Kapitel Übungen, durch die die folgenden Aufgaben illustriert werden:

- Unterschiedliche Diagramm-Modi kennenlernen
- While-Schleife und Diagramm verwenden
- Schaltverhalten eines Booleschen Schalters ändern
- Schleifen-Timing steuern
- Schieberegister verwenden
- Mehrteiliges Diagramm erstellen
- For-Schleife verwenden

Was ist eine Struktur?

Eine *Struktur* ist ein Programmsteuerelement. Strukturen steuern den Datenfluß in einem VI. G bietet fünf verschiedene Strukturen: die While-Schleife, die For-Schleife, die Case-Struktur, die Sequenzstruktur und den Formel-Knoten. Dieses Kapitel enthält eine Einführung in die While- und die For-Schleifenstrukturen einschließlich des Diagramms und des Schieberegisters. Die Case-Struktur, die Sequenzstruktur und der Formel-Knoten werden in Kapitel 4, *Case- und Sequenzstrukturen und der Formel-Knoten* näher erläutert.

While- und For-Schleifen stellen Grundstrukturen für das Programmieren mit G dar, und sie kommen in den meisten Beispielen für G und in den Übungen in diesem Handbuch vor. Weitere Informationen zu Schleifen finden Sie außerdem in Kapitel 19, *Strukturen*, im *Referenzhandbuch zur Programmierung in G*.

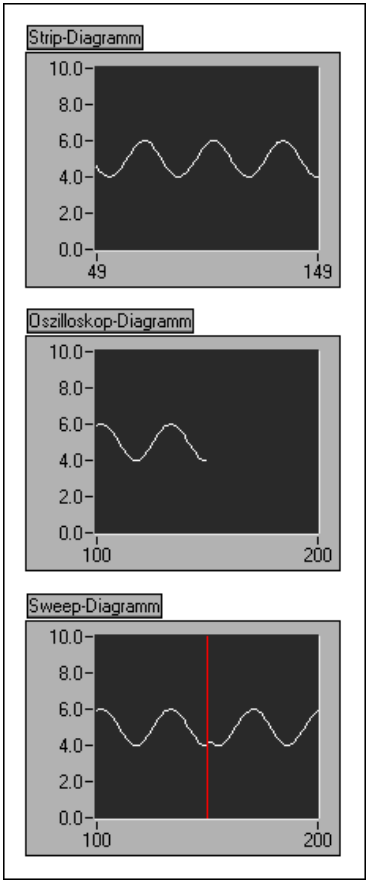
Beispiele für Strukturen finden Sie in `Examples\General\structs.llb`. Beispiele für Diagramme finden Sie in `Examples\General\Graphs\charts.llb`.

Diagramme

Ein *Diagramm* ist ein numerisches Anzeigeelement, das periodisch mit neuen Daten aktualisiert wird. In der Palette **Elemente»Graph** können Sie zwei verschiedene Diagrammarten finden: Kurvendiagramme und Intensitätsdiagramme. Sie können die Diagramme so anpassen, daß die von Ihnen gewünschten Daten oder zusätzliche Informationen angezeigt werden. Für die Diagramme stehen folgende Funktionselemente zur Verfügung: ein Rollbalken, eine Legende, eine Palette, eine digitale Anzeige und die Zuordnung einer Zeitachse zu den Skalen. Weitere Informationen zu Diagrammen finden Sie in Kapitel 15, *Bedien- und Anzeigeelemente vom Typ Graph und Diagramm*, im *Referenzhandbuch zur Programmierung in G*.

Diagramm-Modi

Die folgende Abbildung zeigt die drei Anzeigeoptionen für Diagramme—**Strip-Diagramm, Oszilloskop-Diagramm und Sweep-Diagramm**—die über das Untermenü **Datenoperationen»Aktualisierungsmodus** gewählt werden können. Der Standardmodus ist das Strip-Diagramm.

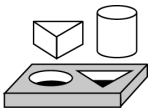


Beschleunigte Diagrammaktualisierung

Sie können ein Array mit mehreren Werten an das Diagramm weiterleiten. Das Diagramm behandelt diese Eingaben wie neue Daten für einen einzelnen Plot. Beachten Sie das Beispiel `charts.vi`, das sich in `Examples\General\Graphs\charts.llb` befindet.

Überlagerte Plots gegenüber Stapelplots

Sie können mehrere Plots in einem Diagramm mit einer einzigen vertikalen Skala anzeigen. Diese Plots werden als überlagerte Plots bezeichnet. Bei den sogenannten Stapelplots werden mehrere vertikale Skalen verwendet. Beachten Sie das Beispiel `charts.vi`, das sich in `Examples\General\Graphs\charts.llb` befindet.



Übung 3-1. Mit Diagramm-Modi experimentieren

Übungsziel ist das Anzeigen eines Diagramms, während das VI im Strip-Diagramm-, Oszilloskop-Diagramm- und Sweep-Diagramm-Modus ausgeführt wird.

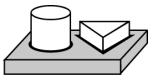
1. Öffnen Sie das VI `Charts.vi`, das sich im Verzeichnis `LabVIEW\Examples\General\Graphs\charts.11b` befindet.
2. Führen Sie das VI aus.

Die Skalenanzeige im Strip-Diagramm-Modus ähnelt der Diagrammaufzeichnung auf einem Papierstreifen. Neue Werte werden am rechten Rand gezeichnet, während die älteren Werte nach links verschoben werden.

Die Nachführanzeige im Oszilloskop-Diagramm-Modus ähnelt der Anzeige eines Oszilloskops. Neue Werte, die vom VI empfangen werden, werden rechts neben dem letzten Wert gezeichnet. Wenn das gezeichnete Diagramm den rechten Rand erreicht, wird die Zeichnung gelöscht, und das VI beginnt mit der Zeichnung wieder am linken Rand. Das Oszilloskop-Diagramm kann wesentlich schneller erstellt werden, da keine Bearbeitungsressourcen für den Bildlauf verwendet werden müssen.

Der Sweep-Diagramm-Modus bewirkt eine ähnliche Anzeige wie bei einem Oszilloskop-Diagramm. Allerdings werden die Daten dabei nicht gelöscht, wenn sie den rechten Rand erreichen. Stattdessen zeigt das VI beim Eingang neuer Daten durch eine über den Bildschirm wandernde vertikale Linie an, wo die neuen Daten beginnen.

3. Rufen Sie das Popup-Menü für eines der Diagramme auf, während das VI ausgeführt wird, und wählen Sie **Aktualisierungsmodus**. Sie können nun den aktuellen Modus in einen anderen Diagramm-Modus ändern. Beachten Sie die Unterschiede zwischen den verschiedenen Diagrammen und Modi.
4. Halten Sie das VI an, und schließen Sie es.

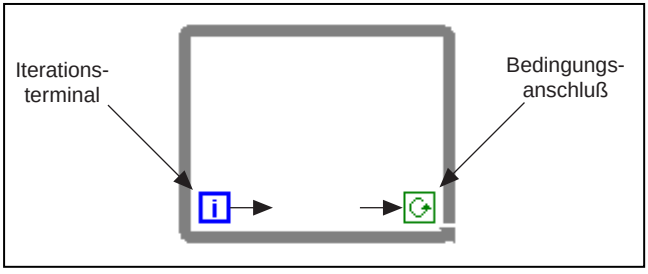


Ende der Übung 3-1.

While-Schleifen

Eine While-Schleife ist eine Struktur, durch die ein Abschnitt eines Codes so lange wiederholt wird, bis eine bestimmte Bedingung erfüllt ist. Eine solche Schleife entspricht einer Do- oder Repeat Until-Schleife in traditionellen Programmiersprachen.

Bei der While-Schleife, die in der folgenden Abbildung dargestellt ist, handelt es sich um ein in der Größe veränderbares Feld, das zur Ausführung des darin enthaltenen Diagramms benutzt wird, bis der Boolesche Wert, der an den Bedingungsanschluß (d.h. an ein Eingangsterminal) geleitet wird, FALSCH ist. Das VI überprüft den Bedingungsanschluß am Ende jeder Iteration; die While-Schleife wird also immer mindestens einmal ausgeführt. Das Iterationsterminal ist ein numerisches Ausgangsterminal, das die Anzahl der Schleifenausführungen ausgibt. Das Iterationsterminal beginnt beim Zählen der Iterationen immer mit dem Wert Null. Wenn die Schleife also einmal ausgeführt wird, gibt das Iterationsterminal “0” aus.



Die While-Schleife entspricht dem folgenden Pseudocode:

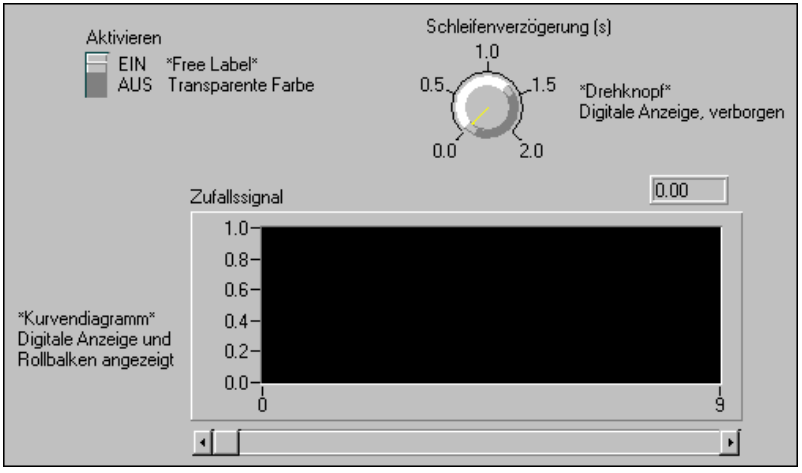
```
Do
Execute Diagram Inside the Loop (which sets the
condition)
While Condition is TRUE
```

Übung 3-2. While-Schleife und Diagramm verwenden

Übungsziel ist das Verwenden einer While-Schleife und eines Diagramms zum Erfassen und Anzeigen von Daten in Echtzeit.

In dieser Übung erstellen Sie ein VI, das Zufallsdaten erzeugt und in einem Diagramm anzeigt. Mit einem Drehknopf auf dem Frontpanel wird eine Schleifenrate zwischen 0 und 2 Sekunden eingestellt, und durch einen Schalter wird das VI angehalten. Damit Sie den Schalter nicht jedesmal einschalten müssen, wenn Sie das VI ausführen, werden Sie außerdem das Schaltverhalten des Schalters ändern. Beginnen Sie mit dem in der folgenden Abbildung dargestellten Frontpanel.

Frontpanel



1. Wählen Sie **Datei»Neu**, um ein neues Frontpanel zu öffnen.
2. Plazieren Sie einen vertikalen Schalter (**Elemente»Boolesch**) auf dem Frontpanel. Beschriften Sie den Schalter mit `Aktivieren`.
3. Verwenden Sie das Beschriftungswerkzeug, um freie Textfelder für `EIN` und `AUS` zu erstellen. Wählen Sie das Beschriftungswerkzeug, und geben Sie den Text für das Label ein. Benutzen Sie das Farbenwerkzeug, das links abgebildet ist, um den Rand des freien Labels transparent zu gestalten. Wählen Sie dazu T in der unteren linken Ecke der Palette **Farbe**.

4. Plazieren Sie ein Kurvendiagramm (**Elemente»Graph**) auf dem Frontpanel. Beschriften Sie das Diagramm mit `Zufallssignal`. Das Diagramm zeigt Zufallsdaten in Echtzeit an.

Hinweis ***Achten Sie darauf, daß Sie ein Kurvendiagramm und nicht einen Kurvengraphen wählen. Das Kurvendiagramm wird in der Palette* Graph *auf der linken Seite angezeigt.***

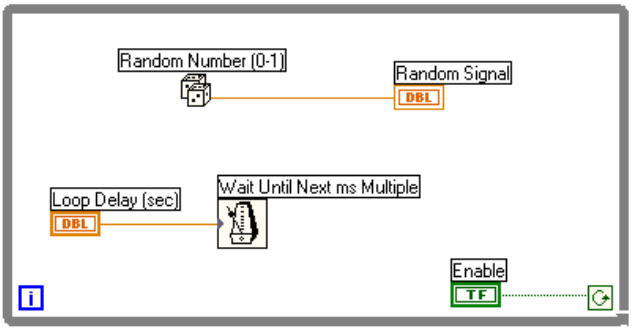
5. Rufen Sie das Popup-Menü für ein Diagramm auf, und wählen Sie **Anzeigen»Palette** und **Anzeigen»Legende**, um die Palette und die Legende auszublenden. In der digitalen Anzeige wird der letzte Wert angezeigt. Rufen Sie dann noch einmal das Popup-Menü für das Diagramm auf, und wählen Sie **Anzeigen»Digitale Anzeige** und **Anzeigen»Rollbalken**.

6. Richten Sie die Skala für das Diagramm auf Werte zwischen `0,0` und `1,0` ein. Verwenden Sie das Beschriftungswerkzeug, um den Höchstwert von `10,0` auf `1,0` zu ändern.

7. Plazieren Sie einen Drehknopf (**Elemente»Numerisch** auf dem Frontpanel. Beschriften Sie den Drehknopf mit `Schleifenverzögerung (Sek.)`. Mit diesem Drehknopf wird der Zeittakt der While-Schleife gesteuert. Rufen Sie das Popup-Menü für den Drehknopf auf, und entfernen Sie die Markierung von **Anzeigen» Digitale Anzeige**, um die digitale Anzeige auszublenden.
8. Versehen Sie den Drehknopf mit einer neuen Skala. Doppelklicken Sie mit dem Beschriftungswerkzeug in der Skala um den Drehknopf auf `10,0`, und ersetzen Sie den Wert durch `2,0`.

Blockdiagramm

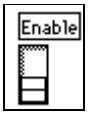
9. Öffnen Sie das Blockdiagramm, und erstellen Sie das in der folgenden Abbildung dargestellte Diagramm.



a. Wählen Sie die While-Schleife in **Funktionen»Strukturen** aus, um sie im Blockdiagramm zu plazieren. Die While-Schleife stellt ein in der Größe veränderbares Feld dar, das nicht sofort in das Diagramm eingesetzt wird. Sie haben stattdessen die Möglichkeit, sie Ihren Anforderungen entsprechend zu positionieren und in der Größe zu verändern. Klicken Sie dazu in einen Bereich links über allen Terminals. Halten Sie die Maustaste gedrückt, und ziehen Sie ein Rechteck um die Terminals.

b. Wählen Sie die Funktion für Zufallszahlen (0-1) über **Funktionen»Numerisch** aus.

c. Stellen Sie die im Blockdiagramm gezeigten Verbindungen her. Verbinden Sie die Funktion für Zufallszahlen (0–1) mit dem Zufallssignal-Diagrammterminal und den Aktivierungsschalter mit dem Bedingungsanschluß der While-Schleife. Stellen Sie noch keine Verbindung zum Schleifenverzögerungsterminal her.

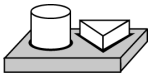


10. Kehren Sie zum Frontpanel zurück, und schalten Sie den vertikalen Schalter ein, indem Sie mit dem Bedienwerkzeug darauf klicken.
11. Speichern Sie das VI unter der Bezeichnung `Random Signal.vi` im Verzeichnis `LabVIEW\Activity`.
12. Führen Sie das VI aus.

Die While-Schleife stellt eine Endlosschleifenstruktur dar. Das darin enthaltene Diagramm wird ständig ausgeführt, solange die angegebene Bedingung WAHR ist. In diesem Beispiel generiert das Diagramm Zufallszahlen und zeigt sie im Diagramm an, solange sich der Schalter in eingeschalteter Position (WAHR) befindet.

13. Halten Sie das VI an, indem Sie auf den vertikalen Schalter klicken. Durch das Ausschalten wird der Wert FALSCH an den Bedingungsanschluß der Schleife gesendet, und die Schleife wird angehalten.
14. Blättern Sie durch das Diagramm. Klicken Sie mit der Maustaste auf einen der beiden Pfeile im Rollbalken, und halten Sie die Taste gedrückt.
15. Leeren Sie den Anzeigepuffer, und setzen Sie das Diagramm zurück, indem Sie das Popup-Menü für das Diagramm aufrufen und **Datenoperationen»Diagramm löschen** wählen.

Hinweis ***Die Größe des Anzeigepuffers beträgt 1024 Punkte. Sie können die Puffergröße erhöhen oder verringern, indem Sie das Popup-Menü für das Diagramm aufrufen und* Länge der Historie eines Diagramms… *wählen. Sie können diese Funktion nur verwenden, wenn das VI nicht ausgeführt wird.***



Ende der Übung 3-2.

Schaltverhalten eines Booleschen Schalters

Es ist möglich, daß Sie bei jedem Ausführen des VIs den vertikalen Schalter erneut einschalten und dann auf die Taste **Ausführen** in der Symbolleiste klicken müssen. G bietet darum die Möglichkeit, das Schaltverhalten Boolescher Bedienelemente zu verändern.

Es gibt sechs verschiedene Einstellmöglichkeiten für das Schaltverhalten eines Booleschen Steuerelements:

- Schaltet, wenn gedrückt
- Schaltet, wenn losgelassen
- Schaltet, bis losgelassen
- Latch, während gedrückt
- Latch, wenn losgelassen
- Latch bis zum Loslassen

Die untenstehenden Abbildungen zeigen die verschiedenen Booleschen Schalter. Daneben befindet sich jeweils eine Beschreibung des Schaltverhaltens.

Schaltet, wenn gedrückt — Ändert jedesmal den Steuerwert, wenn Sie mit dem Bedienwerkzeug auf das Steuerelement klicken. Dieser Vorgang ähnelt der Benutzung eines Lichtschalters und wird nicht dadurch beeinflußt, wie oft das VI das Steuerelement abliest.

Schaltet, wenn losgelassen — Ändert den Steuerwert erst dann, wenn Sie die Maustaste nach dem Klicken im grafischen Bereich des Steuerelements wieder loslassen. Der Vorgang wird nicht dadurch beeinflußt, wie oft das VI das Steuerelement abliest. Er ähnelt dem Klicken auf ein Kontrollkästchen in einem Dialogfeld. Das Kontrollkästchen wird dabei zwar hervorgehoben, aber eine Änderung tritt erst dann ein, wenn Sie die Maustaste loslassen.

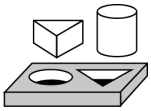
Schaltet, bis losgelassen — Ändert den Steuerwert, wenn Sie auf das Steuerelement klicken. Es behält den neuen Wert bei, bis Sie die Maustaste

wieder loslassen. Danach kehrt das Steuerelement zu dem ursprünglichen Wert zurück. Der Vorgang ähnelt dem Drücken einer Türklingel und wird nicht dadurch beeinflußt, wie oft das VI das Steuerelement abliest.

Latch, während gedrückt — Ändert den Steuerwert, wenn Sie auf das Steuerelement klicken. Das Steuerelement behält den neuen Wert bei, bis das VI den Wert einmal gelesen hat. Danach kehrt das Steuerelement zu seinem Standardwert zurück. Dabei spielt es keine Rolle, ob Sie weiterhin auf die Maustaste drücken. Dieser Vorgang ähnelt dem Vorgang beim Auslösen eines Spannungsschutzschalters und ist besonders nützlich, wenn Sie While-Schleifen anhalten möchten oder wenn das VI jedesmal nach dem Einstellen des Steuerelements nur eine einzelne Aufgabe ausführen soll.

Latch, wenn losgelassen — Ändert den Steuerwert erst dann, wenn Sie die Maustaste loslassen. Wenn das VI den Wert einmal gelesen hat, kehrt das Steuerelement zum alten Wert zurück. Dieser Vorgang garantiert, daß mindestens ein neuer Wert gelesen wird. Wie das Schalten beim Loslassen ähnelt auch dieser Vorgang dem Verhalten von Tasten in einem Dialogfeld. Durch Klicken werden die Tasten hervorgehoben, und beim Loslassen der Maustaste wird ein gelesener Wert festgehalten.

Latch bis zum Loslassen — Ändert den Steuerwert, wenn Sie auf das Steuerelement klicken. Das Steuerelement behält den Wert bei, bis das VI den Wert einmal gelesen hat oder bis Sie die Maustaste wieder loslassen. Dabei ist entscheidend, was zuletzt geschieht.



Übung 3-3. Schaltverhalten eines Booleschen Schalters ändern

Übungsziel ist das Experimentieren mit dem unterschiedlichen Schaltverhalten von Booleschen Schaltern.

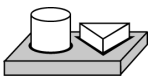
1. Öffnen Sie das VI `Random Signal.vi`, das Sie in Übung 3-2 im Verzeichnis `LabVIEW\Activity` gespeichert haben. Der Standardwert für den Schalter `Aktivieren` ist FALSCH.
2. Ändern Sie den vertikalen Schalter, so daß er nur zum Anhalten des VIs benutzt wird. Führen Sie die notwendigen Änderungen durch, damit Sie den Schalter nicht jedesmal betätigen müssen, wenn Sie das VI ausführen.
   a. Schalten Sie den vertikalen Schalter mit dem Bedienwerkzeug ein.

b. Rufen Sie das Popup-Menü für den Schalter auf, und wählen Sie **Datenoperationen»Aktuellen Wert als Standard übernehmen**. Auf diese Weise wird die Stellung EIN zum Standardwert.

c. Rufen Sie das Popup-Menü für den Schalter auf, und wählen Sie **Schaltverhalten»Latch, während gedrückt**.

3. Führen Sie das VI aus. Klicken Sie auf den Schalter `Aktivieren`, um die Datenerfassung anzuhalten. Der Schalter springt kurzzeitig in die Stellung AUS und wird dann wieder in die Stellung EIN zurückgesetzt.
4. Speichern Sie das VI.

☞ **Hinweis** ***Zu Ihrer Information enthält LabVIEW unter der Bezeichnung*** `Mechanical Action of Booleans.vi` ***ein Beispiel, durch das dieses Verhalten demonstriert wird. Es befindet sich in*** `Examples\General\Controls\booleans.llb`**.**



Ende der Übung 3-3.

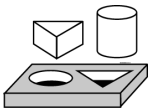
Timing

Beim Aktivieren des VIs in der vorherigen Übung wurde die While-Schleife so schnell wie möglich ausgeführt. Sie können die Ausführung der Schleife jedoch auch mit Hilfe der Funktionen in der Palette **Funktionen»Zeit & Dialog** verlangsamen, damit die Schleife in bestimmten Intervallen wiederholt wird.

Die Timing-Funktionen messen die Zeit in Millisekunden (ms); unter Umständen ist Ihr Betriebssystem jedoch nicht in der Lage, ein so hohes Maß an Genauigkeit aufrechtzuerhalten.

- **(Windows 95/NT)** Der Zeitgeber arbeitet mit einer Auflösung von 1 ms. Dieser Wert wird von der Hardware beeinflußt. Langsamere Systeme, z.B. ein 80386-Prozessor, arbeiten eventuell mit längeren Zeitschritten.
- **(Windows 3.1)** Der Zeitgeber arbeitet mit einer Standardauflösung von 55 ms. Sie können LabVIEW so konfigurieren, daß eine Auflösung von 1 ms verwendet wird. Wählen Sie dazu **Bearbeiten» Voreinstellungen…**, wählen Sie “Leistung und Speichermedium” im Pfadring, und entfernen Sie die Markierung von dem Kontrollkästchen “Standard-Timer verwenden”. LabVIEW verwendet die Auflösung von 1 ms nicht als Standard, da dadurch das Betriebssystem stärker belastet wird.

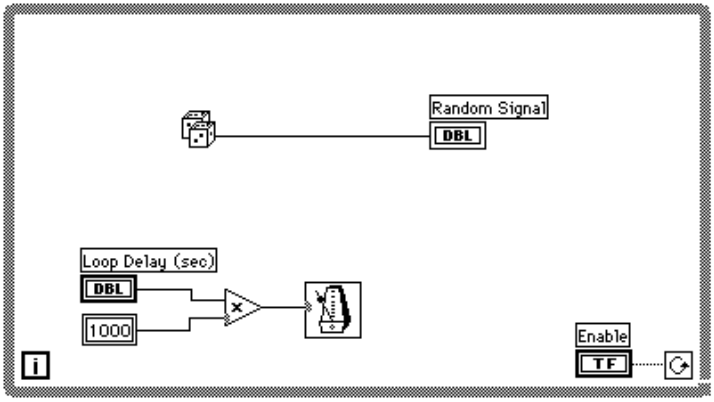
- **(Macintosh)** Bei 68K-Systemen ohne die QuickTime-Erweiterung verwendet der Zeitgeber eine Auflösung von 16 2/3 ms (1/60 einer Sekunde). Wenn Sie einen Power Macintosh verwenden oder QuickTime installiert ist, beträgt die Auflösung für den Zeitgeber 1 ms.
- **(UNIX)** Der Zeitgeber hat eine Auflösung von 1 ms.



Übung 3-4. Schleifen-Timing steuern

Übungsziel ist das Steuern des Schleifen-Timings, damit sichergestellt ist, daß bei jeder Iteration die angegebene Millisekundenzahl nicht unterschritten wird.

1. Öffnen Sie `Random Signal.vi`, das Sie in Übung 3-3 geändert und im Verzeichnis `LabVIEW\Activity` gespeichert haben.
2. Verändern Sie das VI entsprechend der folgenden Abbildung, damit eine neue Zufallszahl in dem durch den Drehknopf eingestellten Zeitintervall generiert wird.

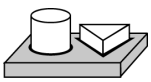


Wartet bis zum nächsten Vielfachen von ms (**Funktionen» Zeit & Dialog**) — Multiplizieren Sie den Wert des Drehknopfterminals mit 1000, um den Drehknopfwert von Sekunden in Millisekunden umzuwandeln. Benutzen Sie diesen Wert als Eingangswert für die Funktion “Wartet bis zum nächsten Vielfachen von ms”.

Multiplizieren (**Funktionen»Numerisch**) — Mit der Multiplikationsfunktion wird der Drehknopfwert mit 1000 multipliziert, um Sekunden in Millisekunden umzuwandeln.

Numerische Konstante (**Funktionen»Numerisch**)—Die numerische Konstante gibt die Konstante an, mit der Sie den Drehknopfwert multiplizieren müssen, um eine Angabe in Millisekunden zu erhalten. Wenn der Drehknopf auf einen Wert von 1,0 eingestellt ist, wird die Schleife einmal pro 1000 Millisekunden (d.h. einmal pro Sekunde) ausgeführt.

3. Führen Sie das VI aus. Drehen Sie am Drehknopf, damit Sie unterschiedliche Werte für die Schleifenverzögerung erhalten. Beobachten Sie den Effekt der Schleifenverzögerung auf die Aktualisierung der Anzeige für das `Zufallssignal`.
4. Speichern Sie das VI unter dem Namen `Random Signal with Delay.vi` im Verzeichnis `LabVIEW\Activity`. Schließen Sie das VI.

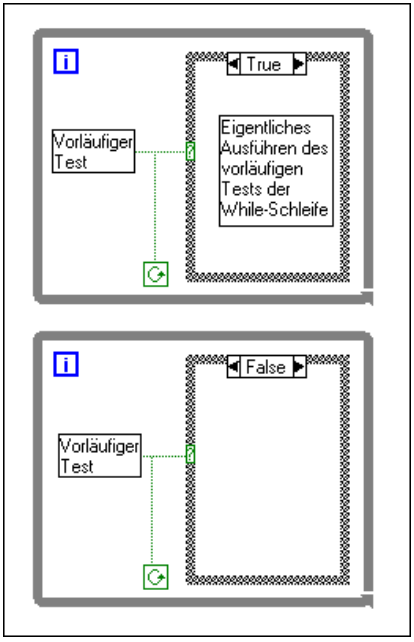


Ende der Übung 3-4.

Codeausführung in der ersten Iteration verhindern

Die While-Schleife wird immer mindestens einmal ausgeführt, weil G den Kontinuitätsschleifentest nach der Ausführung des Diagramms durchführt. Sie können eine While-Schleife einrichten, die einen vorläufigen Test ihres Bedingungsanschlusses durch Einbeziehung einer Case-Struktur in die Schleife durchführt. Verbinden Sie einen Booleschen Eingang mit dem Case-Strukturauswahlterminal, damit das Unterdiagramm für die Bedingung FALSCH ausgeführt wird, falls der Code in der While-Schleife nicht ausgeführt wird. Weitere Informationen über die Verwendung von Case-Strukturen finden Sie in Kapitel 4, *Case- und Sequenzstrukturen und der Formel-Knoten*.

Das Unterdiagramm für die Bedingung WAHR enthält das Arbeitsergebnis der While-Schleife. Der Kontinuitätstest läuft außerhalb der Case-Struktur ab, und die Resultate werden mit dem Bedingungsanschluß der While-Schleife und dem Auswahlterminal der Case-Struktur verbunden. Die Beschriftungsfelder in der folgenden Abbildung zeigen die Bedingung während des vorläufigen Tests.

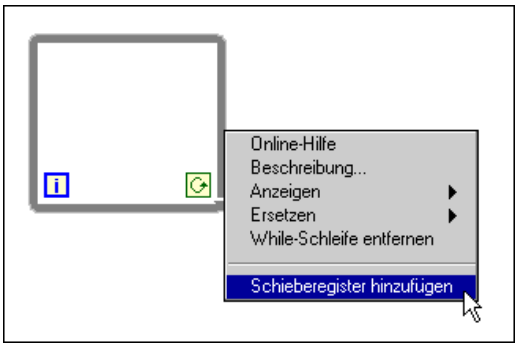


Dieses Beispiel führt zum selben Ergebnis wie der folgende Pseudocode:

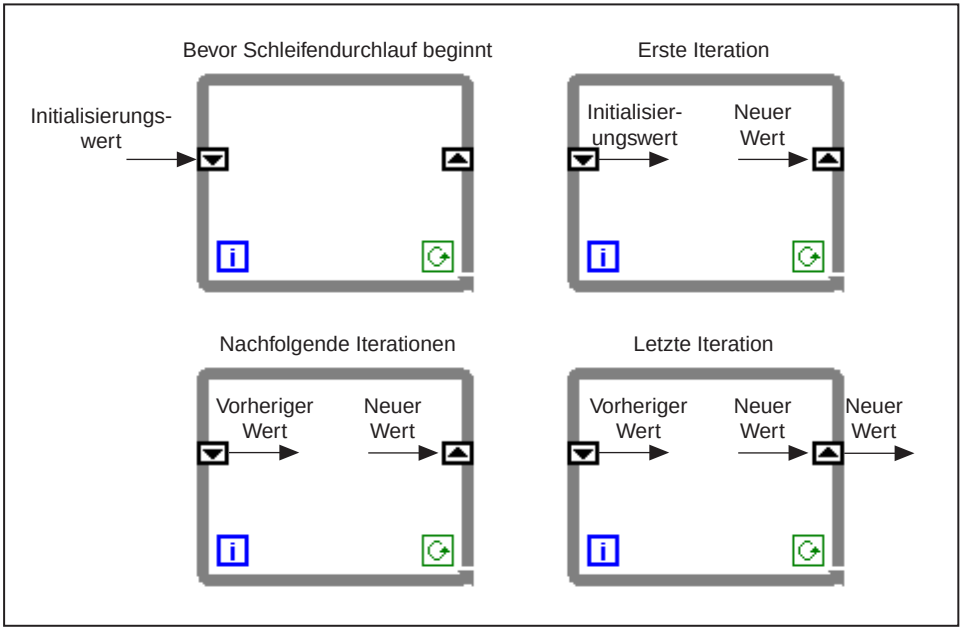
```
While (pretest condition)
Do actual work of While Loop
Loop
```

Schieberegister

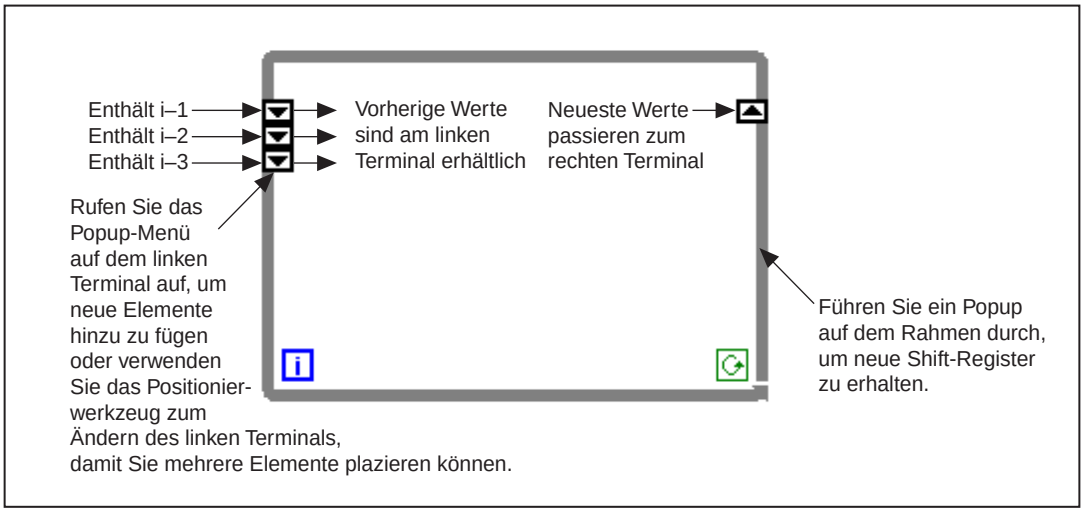
Schieberegister (verfügbar für While-Schleifen und für For-Schleifen) übertragen Werte von einer Schleifeniteration zur nächsten. Sie können ein Schieberegister erstellen, indem Sie das Popup-Menü für den linken oder rechten Rand einer Schleife aufrufen und **Schieberegister hinzufügen** wählen.

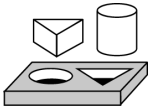


Das Schieberegister enthält zwei sich direkt gegenüberliegende Terminals auf der vertikalen Seite des Schleifenrahmens. Das rechte Terminal speichert die Daten nach Abschluß einer Iteration. Wie die folgende Abbildung zeigt, werden diese Daten am Ende der Iteration verschoben und erscheinen zu Beginn der nächsten Iteration im linken Terminal. Ein Schieberegister kann jeden Datentyp aufnehmen — numerisch, Boolesch, String, Array usw. Das Schieberegister paßt sich automatisch an den Datentyp des ersten Objekts an, das mit dem Schieberegister verbunden wird.



Sie können das Schieberegister so konfigurieren, daß es sich die Werte von mehreren vorherigen Iterationen merkt. Diese Funktion ist besonders nützlich, wenn Sie den Durchschnitt von Datenpunkten ermitteln möchten. Sie können zusätzliche Terminals erstellen, um auf Werte von vorherigen Iterationen zuzugreifen. Rufen Sie dazu das Popup-Menü für das linke oder rechte Terminal auf, und wählen Sie **Element hinzufügen**. Wenn ein Schieberegister zum Beispiel drei Elemente im linken Terminal enthält, können Sie, wie die folgende Abbildung illustriert, auf Werte der letzten drei Iterationen zugreifen.



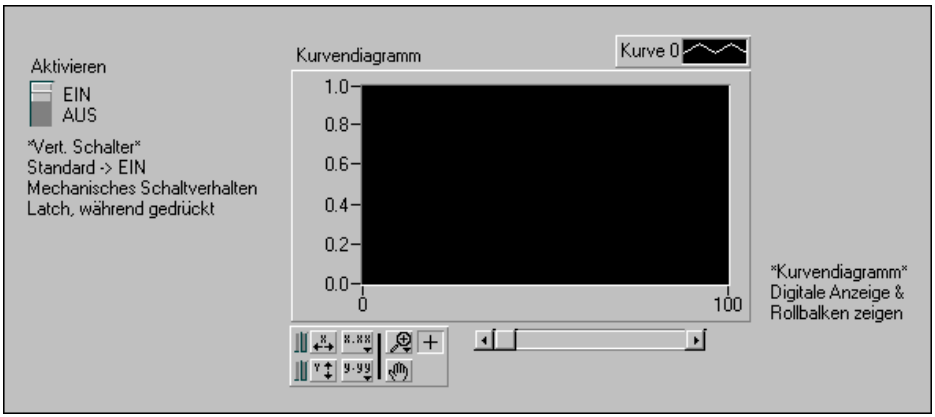


Übung 3-5. Schieberegister verwenden

Übungsziel ist das Erstellen eines VIs, das laufend einen Durchschnittswert in einem Diagramm anzeigt.

Frontpanel

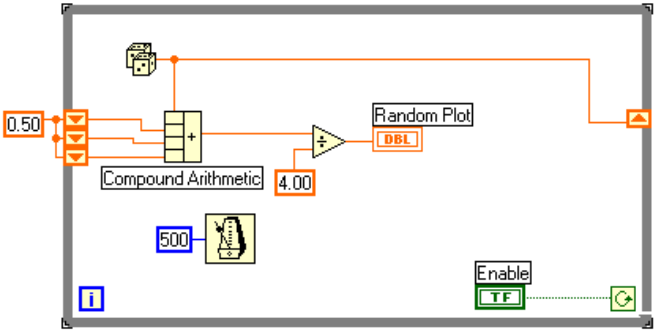
1. Öffnen Sie ein neues Frontpanel, und erstellen Sie die Objekte, die in der folgenden Abbildung dargestellt sind.



2. Ändern Sie die Skala für das Kurvendiagramm, damit Werte zwischen `0,0` und `2,0` erfaßt werden.
3. Fügen Sie den vertikalen Schalter ein. Rufen Sie dann das Popup-Menü für den Schalter auf, wählen Sie **Schaltverhalten»Latch, während gedrückt**. Legen Sie EIN als Standardzustand fest, indem Sie **Ausführen»Aktuelle Werte als Standard übernehmen** wählen.

Blockdiagramm

4. Erstellen Sie das Blockdiagramm, das in der folgenden Abbildung dargestellt ist.



5. Fügen Sie die While-Schleife (**Funktionen»Strukturen**) in das Blockdiagramm ein, und erstellen Sie das Schieberegister.

   a. Rufen Sie das Popup-Menü für den linken oder rechten Rand der While-Schleife auf, und wählen Sie **Schieberegister hinzufügen**.

   b. Fügen Sie ein zusätzliches Element hinzu, indem Sie das Popup-Menü für das linke Terminal des Schieberegisters aufrufen und **Element hinzufügen** wählen. Fügen Sie auf dieselbe Weise ein drittes Element hinzu.

Zufallszahl (0–1) (**Funktionen»Numerisch**) — Durch diese Funktion werden Zufallszahlen zwischen 0 und 1 generiert.

Mehrfacharithmetik (**Funktionen»Numerisch**) — In dieser Übung wird durch die Funktion “Mehrfacharithmetik” die Summe der Zufallszahlen aus zwei Iterationen ausgegeben. Rufen Sie das Popup-Menü für einen Eingang auf, wenn Sie weitere Eingänge hinzufügen möchten, und wählen Sie dann “Eingang hinzufügen” im Popup-Menü.

Dividieren (**Funktionen»Numerisch**) — In dieser Übung wird durch die Funktion “Dividieren” der Durchschnittswert für die letzten vier Zufallszahlen ausgegeben.

Numerische Konstante (**Funktionen»Numerisch**) — Bei jeder Iteration der While-Schleife wird durch die die Funktion “Zufallszahl (0–1)” ein Zufallswert generiert. Das VI fügt diesen Wert den letzten drei Werten hinzu, die in den linken Terminals des Schieberegisters gespeichert sind. Durch die Funktion “Zufallszahl (0–1)” wird das Ergebnis durch vier geteilt und so der Durchschnitt der Werte (d.h. des aktuellen Werts plus der drei letzten Werte) ermittelt. Dieser Durchschnittswert wird im Kurvendiagramm angezeigt.

Wartet bis zum nächsten Vielfachen von ms (**Funktionen» Zeit & Dialog**) — Durch diese Funktion wird sichergestellt, daß jede Iteration der Schleife nicht vor dem in Millisekunden festgelegten Eingangswert durchgeführt wird. Der Eingangswert für diese Übung beträgt 500 Millisekunden. Wenn Sie das Popup-Menü für das Icon aufrufen und **Anzeigen»Label** wählen, erscheint das Label “Wartet bis zum nächsten Vielfachen von ms”.

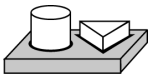
6. Rufen Sie das Popup-Menü für den Eingang der Funktion “Wartet bis zum nächsten Vielfachen von ms” auf, und wählen Sie **Konstante erzeugen**. Daraufhin wird eine numerische Konstante eingeblendet, die automatisch mit der Funktion verbunden wird.

7. Geben Sie `500` als Beschriftung ein. Die numerische Konstante, die mit der Funktion “Wartet bis zum nächsten Vielfachen von ms” verbunden ist, legt eine Wartezeit von 500 Millisekunden (eine halbe Sekunde) fest. Die Schleife wird also nach jeder halben Sekunde einmal ausgeführt.

Beachten Sie, daß das VI die Schieberegister mit einer Zufallszahl initialisiert. Wenn Sie das Schieberegisterterminal nicht initialisieren, enthält es den Standardwert oder den letzten Wert des vorherigen Durchlaufs, und die ersten Durchschnittswerte sind in diesem Fall bedeutungslos.

8. Führen Sie das VI aus, und beobachten Sie den Vorgang.
9. Speichern Sie dieses VI als `Random Average.vi` im Verzeichnis `LabVIEW\Activity`.

Hinweis ***Denken Sie daran, die Schieberegister zu initialisieren, um zu vermeiden, daß alte Daten oder Standarddaten die aktuellen Datenmessungen beeinträchtigen.***

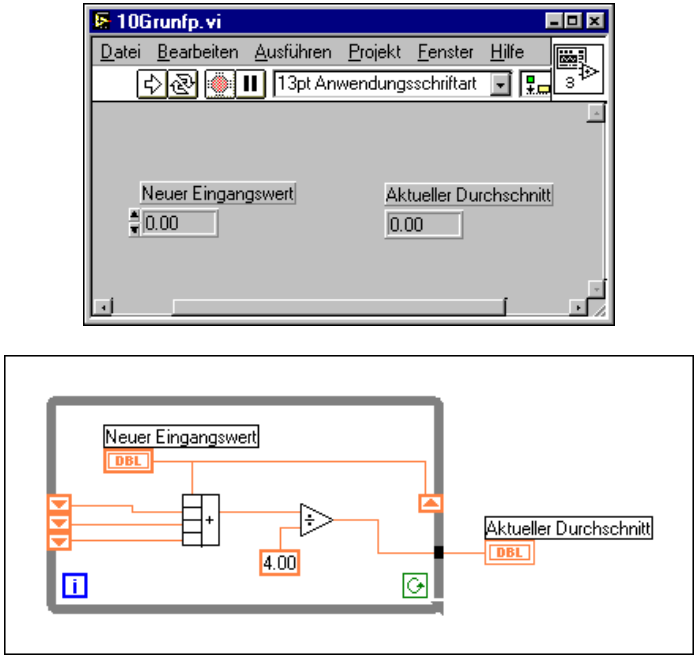


Ende der Übung 3-5.

Nicht-initialisierte Schieberegister verwenden

Sie können ein Schieberegister initialisieren, indem Sie einen Wert von außerhalb der While- oder For-Schleife mit dem linken Terminal des Schieberegisters verbinden. Es können sich jedoch Situationen ergeben, in denen Sie ein VI wiederholt mit einer Schleife und einem Schieberegister ausführen möchten, so daß der letzte Wert der vorherigen Ausführung bei jeder erneuten Ausführung des VIs als Anfangsausgangswert des Schieberegisters verwendet wird. Dazu ist es notwendig, daß keine Verbindung von außerhalb der Schleife zum linken Schieberegisterterminal vorhanden ist. Wenn keine Verbindung zum Eingang des linken Schieberegisterterminals vorhanden ist, werden die jeweiligen Zustandsinformationen zwischen aufeinanderfolgenden Ausführungen eines VIs beibehalten.

Die folgende Abbildung zeigt ein Beispiel für ein SubVI, das den aktuellen Durchschnitt für vier Datenpunkte laufend neu berechnet. Das VI benutzt ein nicht-initialisiertes Schieberegister (mit drei zusätzlichen Elementen), um vorherige Datenpunkte zu speichern.



Bei jedem Aufrufen des VIs wird der `aktuelle Durchschnitt` aufgrund des neuen Eingangswerts und der drei vorherigen Werte neu berechnet. Der neue Wert wird dann im Schieberegister gespeichert, und die vorherigen beiden Werte werden im Schieberegister nach oben verschoben. Es besteht keine Verbindung von einem Eingangswert zur Eingangsseite des linken Schieberegisters, d.h., alle drei Werte werden für die nächste Ausführung des VIs beibehalten.

Da bei diesem SubVI keine Verbindung zum Bedingungsanschluß vorhanden ist, wird es genau einmal ausgeführt, wenn es aufgerufen wird. Die While-Schleife in diesem SubVI wird nicht dazu verwendet, mehrere Schleifendurchgänge auszuführen, sondern um zwischen den einzelnen Durchgängen Werte in den Schleifen-Schieberegistern zu speichern.

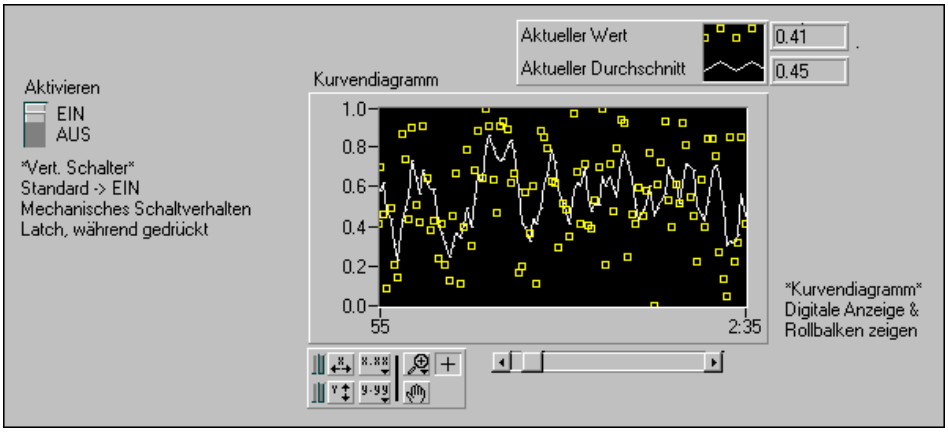
Wenn das VI zum Erstellen des aktuellen Durchschnitts in den Speicher geladen wird, werden die nicht-initialisierten Schieberegister automatisch auf Null gesetzt. Bei Schieberegistern, die mit Booleschen Werten verbunden sind, ist der Ausgangswert FALSCH.

Übung 3-6. Mehrteiliges Diagramm erstellen

Übungsziel ist das Erstellen eines Diagramms, das mehr als einen Plot darstellen kann.

Frontpanel

1. Öffnen Sie das VI `Random Average.vi`, das Sie in Übung 3-5 erstellt haben.
2. Ändern Sie das Frontpanel, so daß es der folgenden Abbildung entspricht.



a. Verwenden Sie das Positionierwerkzeug, um die Legende so weit zu strecken, daß sie zwei Plots aufnehmen kann.

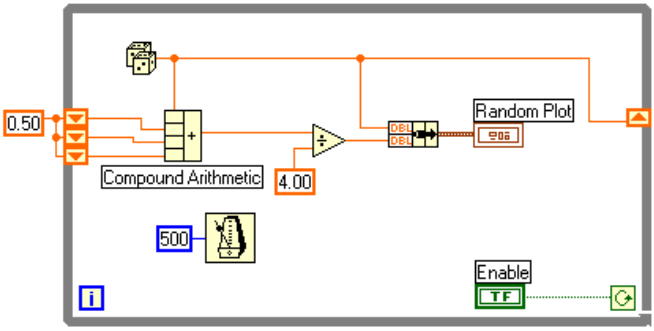
b. Blenden Sie die digitale Anzeige ein, indem Sie das Popup-Menü für das Diagramm aufrufen und **Anzeigen»Digitale Anzeige** aufrufen. Verschieben Sie die Legende, falls notwenig.

c. Benennen Sie den Plot 0 in `Aktueller Wert` um, indem Sie mit dem Beschriftungswerkzeug auf die Beschriftung doppelklicken und den neuen Text eingeben. Sie können die Größe des Beschriftungsbereichs verändern, indem Sie mit dem Positionierwerkzeug an einer der linken Ecken ziehen. Benennen Sie den Plot 1 auf dieselbe Weise in `Aktueller Durchschnitt` um.

d. Ändern Sie den Plot `Aktueller Wert`, so daß eine unverbundene Interpolation, ein quadratischer Punktstil und die Farbe Grün verwendet wird. Sie können den Stil und die Farbe des Plots ändern, indem Sie das Popup-Menü für die Legende aufrufen.

Blockdiagramm

3. Ändern Sie das Blockdiagramm entsprechend der folgenden Abbildung ab, damit sowohl der Durchschnitt als auch die aktuelle Zufallszahl im selben Diagramm angezeigt werden.

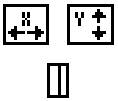


Elemente bündeln (**Funktionen»Cluster**)—In dieser Übung wird die Funktion “Elemente bündeln” dazu verwendet, um den Durchschnitt und den aktuellen Wert für das Zeichnen des Diagramms zu bündeln. Wenn Sie einen Bündelknoten im Blockdiagramm plazieren, wird er durch das links neben diesem Text abgebildete Symbol repräsentiert. Sie können zusätzliche Elemente hinzufügen, indem Sie den Knoten mit Hilfe des Skaliercursors vergrößern. Sie erhalten diesen Cursor, wenn Sie das Positionierwerkzeug auf der Ecke der Funktion plazieren.

Hinweis ***Die Reihenfolge der Eingänge in der Bündelfunktion bestimmt die Reihenfolge der Plots in einem Diagramm. Wenn Sie z.B. die Rohdaten mit dem oberen Eingang und den Durchschnitt mit dem unteren Eingang der Bündelfunktion verbinden, entspricht der erste Plot den Rohdaten und der zweite Plot dem Durchschnitt.***

4. Führen Sie das VI im Frontpanel aus. Das VI zeigt zwei Plots im Diagramm an. Die Plots liegen übereinander. Dies bedeutet, daß Sie dieselbe vertikale Skala gemeinsam benutzen.
5. Führen Sie das VI vom Blockdiagramm aus mit eingeschalteter Highlight-Funktion aus, damit Sie die Daten in den Schieberegistern sehen können.
6. Schalten Sie die Highlight-Funktion aus. Führen Sie das VI im Frontpanel aus. Verändern Sie das Diagramm mit den Tasten in der Palette, während das VI läuft. Sie können jederzeit das Diagramm zurücksetzen, die X- oder Y-Achse skalieren und das Anzeigeformat ändern. Sie können auch mit Hilfe der Rollbalken andere Bereiche

einsehen oder einzelne Bereiche eines Graphen oder eines Diagramms vergrößern.



Mit den Tasten **X** und **Y** können Sie die X- und Y-Achse neu skalieren. Wenn Sie den Graphen so einrichten möchten, daß eine der beiden Skalen ständig automatisch skaliert wird, sollten Sie auf den Fixierschalter links neben der jeweiligen Taste klicken. Dadurch wird die automatische Skalierung fixiert.

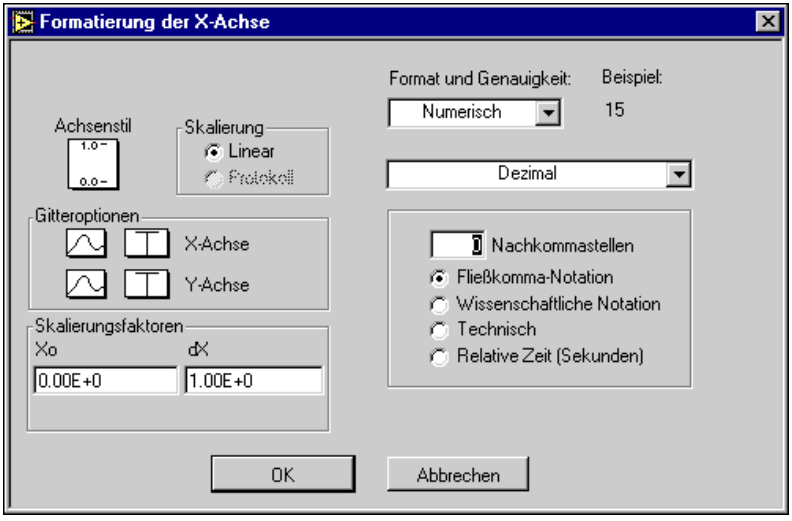
Mit den anderen Tasten können Sie die Textgenauigkeit der Achsen ändern oder den Bedienungsmodus für das Diagramm steuern. Experimentieren Sie mit diesen Tasten, um ihre Funktionen kennenzulernen und um durch den angezeigten Bereich zu blättern oder einzelne Bereiche des Diagramms zu vergrößern.

7. Formatieren Sie die Skalen des Kurvendiagramms, damit sie entweder die absolute oder die relative Zeit angeben. Rufen Sie zur Auswahl des Zeitformats für die X-Skala das Popup-Menü für die X-Skala auf, und wählen Sie **Formatieren…**.
   a. Wählen Sie die Option **Zeit & Datum** im Menüring **Format und Präzision**, um das absolute Zeitformat einzustellen. Dadurch wird das Dialogfeld wie in der folgenden Abbildung verändert. Wenn Sie einen bestimmten Anfangszeitpunkt und bestimmte Intervallabstände für das Kurvendiagramm festlegen möchten, können Sie die Werte *Xo* bzw. *dX* bearbeiten.

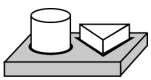
b. Formatieren Sie das Diagramm entsprechend der obigen Abbildung, damit die Daten ab dem 24. Oktober 1996 um 12 Uhr mittags und in Intervallen von jeweils 10 Minuten angezeigt werden.

☞ **Hinweis** ***Bei Veränderung des Achsentextformats wird oft mehr Platz benötigt, als zunächst für die Achse vorgesehen wurde. Wenn Sie die Achse verändern, kann es vorkommen, daß die Länge des Textes die maximale Größe überschreitet, die durch den Signalverlauf korrekt angegeben werden kann. Benutzen Sie in diesem Fall den Skaliercursor, um den Anzeigebereich des Diagramms zu verkleinern.***

8. Wählen Sie **Numerisch** im Menüring **Format und Präzision**, um das relative Zeitformat einzustellen. Danach können Sie die Option **Relative Zeit (Sekunden)** im Dialogfeld wählen, um die Zeit in Sekunden anzuzeigen. Verändern Sie das Dialogfeld entsprechend der folgenden Abbildung, und wählen Sie dann **OK**.



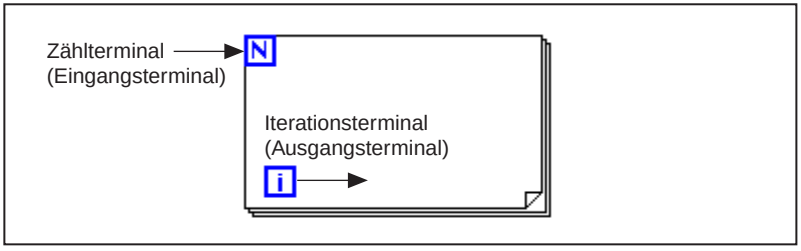
9. Führen Sie das VI aus.
10. Speichern Sie das VI unter dem Namen `Multiple Random Plot.vi` im Verzeichnis `LabVIEW\Activity`.



Ende der Übung 3-6.

For-Schleifen

Eine For-Schleife bewirkt, daß ein Codeabschnitt für eine bestimmte Anzahl von Durchläufen ausgeführt wird. Die Schleife kann in der Größe verändert werden und wird wie auch die While-Schleife nicht sofort auf dem Blockdiagramm eingesetzt. Stattdessen erscheint im Blockdiagramm ein kleines Symbol für die For-Schleife, dessen Größe und Position Sie bestimmen können. Klicken Sie dazu zuerst in einen freien Bereich links oben über allen vorhandenen Terminals. Halten Sie die Maustaste gedrückt, und ziehen Sie ein Rechteck um alle Terminals, die Sie in die For-Schleife aufnehmen möchten. Wenn Sie die Maustaste loslassen, erstellt G an der gewünschten Stelle eine For-Schleife in der gewählten Größe. Sie können die For-Schleife im Blockdiagramm plazieren, indem Sie die Schleife über **Funktionen»Strukturen** auswählen.



Die For-Schleife führt eine im voraus festgelegte Anzahl von Durchläufen für das in der Schleife enthaltene Diagramm durch. Die beiden Terminals, die in der For-Schleife enthalten sind, werden im Folgenden erklärt.

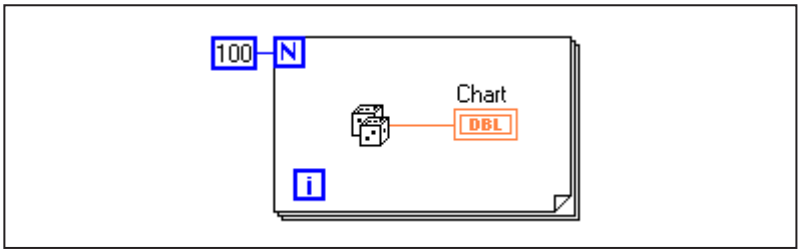
Zählterminal (ein Eingangsterminal)—Das Zählterminal bestimmt, wie oft die Schleife ausgeführt wird.

Iterationsterminal (ein Ausgangsterminal)—Das Iterationsterminal registriert, wie oft die Schleife ausgeführt wurde.

Die For-Schleife entspricht dem folgenden Pseudocode:

```
For i = 0 to N-1
Execute Diagram Inside The Loop
```

Die folgende Abbildung zeigt eine For-Schleife, die 100 Zufallszahlen generiert und die Punkte auf einem Diagramm anzeigt.

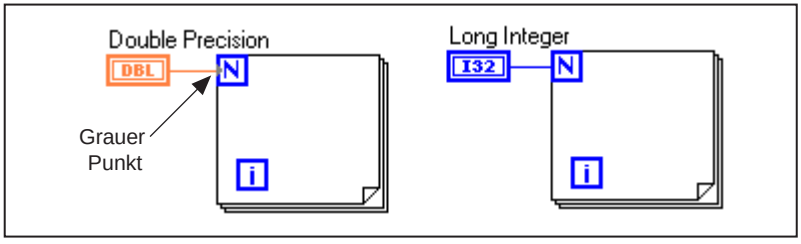

100
N
Chart
DBL
i

Numerische Konvertierung

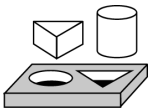
Bisher haben Sie nur numerische Steuer- und Anzeigeelemente benutzt, die Double-Präzision aufweisen, d.h. Fließkommazahlen, die durch 32 Bits repräsentiert werden. G kann numerische Werte allerdings ebenso als Integer-Werte (Byte, Word oder Long) wie auch als Fließkommazahlen (Single-, Double- oder Extended-Präzision) darstellen. Die Standarddarstellung für einen numerischen Wert ist ein Fließkommawert mit Double-Präzision.

Wenn Sie zwei Terminals miteinander verbinden, die unterschiedliche Datentypen aufweisen, konvertiert G eines der Terminals, um die Repräsentierung dem anderen Terminal anzugleichen. Beachten Sie, daß G das Terminal, für das die Konvertierung durchgeführt wird, durch einen grauen Punkt kennzeichnet, der als *Formatumwandlungspunkt* bezeichnet wird.

Die Terminalrepräsentierung für das Zählterminal in der For-Schleife ist z.B. ein Long-Integer-Wert. Wenn Sie eine Fließkommazahl mit Double-Präzision mit dem Zählterminal verbinden, konvertiert G die Zahl in einen Long-Integer-Wert. Beachten Sie den grauen Punkt im Zählterminal der ersten For-Schleife.



Hinweis ***Wenn das VI Fließkommazahlen in Integer-Werte konvertiert, rundet es den Wert auf den nächsten Integer-Wert auf bzw. ab. Wenn eine Zahl genau in der Mitte zwischen zwei Integer-Werten liegt, wird sie auf den am nächsten liegenden geraden Integer-Wert auf- bzw. abgerundet. Das VI rundet z.B. 6,5 auf 6 ab, während es 7,5 auf 8 aufrundet. Es handelt sich dabei um eine IEEE-Standardmethode für das Auf- und Abrunden von Zahlen. Weitere Details finden Sie in der IEEE-Norm 754.***

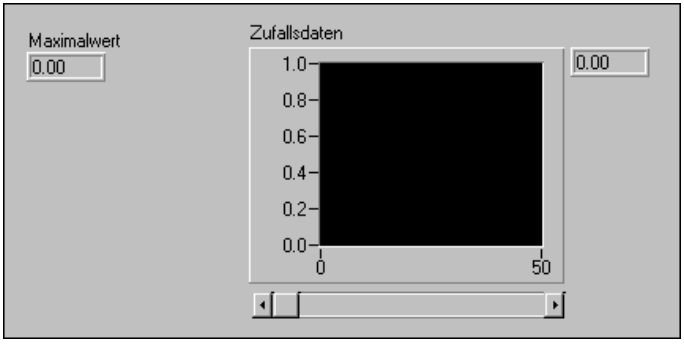


Übung 3-7. For-Schleife verwenden

Übungsziel ist das Verwenden einer For-Schleife und von Schieberegistern zur Berechnung des maximalen Werts in einer Serie von Zufallszahlen.

Frontpanel

1. Öffnen Sie ein neues Frontpanel, und fügen Sie die in der folgenden Abbildung gezeigten Objekte hinzu.



a. Plazieren Sie eine numerische Anzeige auf dem Frontpanel, und kennzeichnen Sie sie als `Maximalwert`.

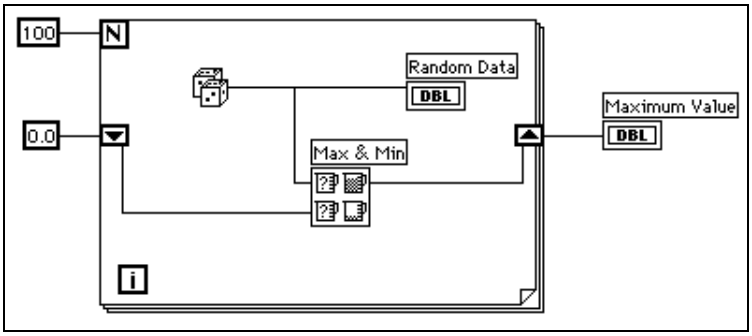
b. Plazieren Sie ein Kurvendiagramm auf dem Frontpanel, und kennzeichnen Sie es als `Zufallsdaten`. Ändern Sie die Skala für das Diagramm, so daß sie von `0,0` bis `1,0` reicht.

c. Rufen Sie das Popup-Menü für das Diagramm auf, und wählen Sie **Anzeigen»Rollbalken** und **Anzeigen»Digitale Anzeige**. Rufen Sie das Popup-Menü für die Palette und die Legende auf, und blenden Sie beide aus.

d. Verändern Sie die Größe des Rollbalkens mit dem Positionierwerkzeug.

Blockdiagramm

2. Öffnen Sie das Blockdiagramm, und verändern Sie es entsprechend der folgenden Abbildung.



3. Plazieren Sie eine For-Schleife (**Funktionen»Strukturen**) auf dem Blockdiagramm.

4. Fügen Sie das Schieberegister hinzu, indem Sie das Popup-Menü für den rechten oder linken Rand der For-Schleife aufrufen bzw. mit der rechten Maustaste auf den Rahmen klicken und dann **Schieberegister hinzufügen** wählen.

5. Fügen Sie dem Blockdiagramm die folgenden Objekte hinzu.

Zufallszahl (0–1) (**Funktionen»Numerisch**)—Durch diese Funktion werden Zufallsdaten generiert.

Numerische Konstante (**Funktionen»Numerisch**)—Für die For-Schleife muß angegeben werden, wie viele Iterationen durchgeführt werden sollen. Im nebenstehenden Beispiel wird die For-Schleife 100mal ausgeführt.

Numerische Konstante (**Funktionen»Numerisch**)—Für diese Übung können Sie den Anfangswert für das Schieberegister auf Null einstellen, da Sie bereits wissen, daß der vom Zufallszahlengenerator ausgegebene Wert zwischen 0,0 und 1,0 liegt.

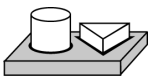
Um ein Schieberegister zu initialisieren, müssen Ihnen ein paar Details der Daten, die Sie sammeln, bereits bekannt sein. Wenn Sie das Schieberegister z.B. auf 1,0 initialisieren, ist dieser Wert bereits größer als alle zu erwartenden Datenwerte und gilt immer als Maximalwert. Wenn Sie das Schieberegister nicht initialisieren, enthält es den Maximalwert eines vorherigen Durchlaufs des VIs. In diesem Fall besteht die Möglichkeit, daß Sie einen maximalen Ausgangswert erhalten, der nicht repräsentativ für die aktuell gesammelten Daten ist.

Max & Min (**Funktionen»Vergleichsoperationen**)—Diese Funktion verwendet zwei numerische Eingangswerte und gibt für beide den maximalen Wert in der oberen rechten Ecke und den minimalen Wert in der unteren rechten Ecke aus. Da nur der maximale Wert im Rahmen dieser Übung von Interesse ist, brauchen Sie nur eine Verbindung zum Ausgang für den maximalen Wert herzustellen und können den minimalen Ausgangswert ignorieren.

6. Verbinden Sie die Terminals in der dargestellten Weise miteinander. Wenn sich das Terminal für den maximalen Wert innerhalb der For-Schleife befände, könnten Sie sehen, wie es ständig aktualisiert wird. Da es sich jedoch außerhalb der Schleife befindet, enthält es nur den zuletzt berechneten maximalen Wert.

☞ **Hinweis** ***Das Aktualisieren der Anzeigeelemente bei jeder Iteration der Schleife ist zeitaufwendig. Sie sollten es möglichst vermeiden, um die Ausführungsgeschwindigkeit zu erhöhen.***

7. Führen Sie das VI aus.
8. Speichern Sie das VI unter dem Namen `Calculate Max.vi` im Verzeichnis `LabVIEW\Activity`.



Ende der Übung 3-7.

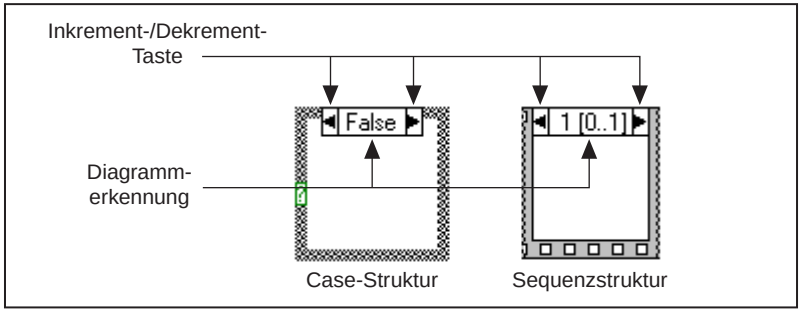
4

Case- und Sequenzstrukturen und der Formel-Knoten

In diesem Kapitel werden die Grundkonzepte für die Arbeit mit Case- und Sequenzstrukturen und Formel-Knoten erklärt. Außerdem enthält dieses Kapitel Übungen, durch die die folgenden Aufgaben illustriert werden:

- Case-Struktur verwenden
- Sequenzstruktur verwenden
- Lokale Sequenz-Variablen verstehen und verwenden
- Formel-Knoten verstehen und verwenden

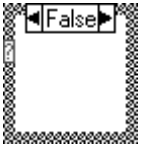
Sowohl Case- als auch Sequenzstrukturen können mehrere Unterdiagramme enthalten, die wie ein Stapel Karten übereinanderliegen und von denen jeweils nur ein Diagramm sichtbar ist. Am oberen Rahmen jeder Struktur befindet sich das *Unterdiagramm-Anzeigefenster*, das in der Mitte eine *Diagrammkennung* und auf jeder Seite Dekrement- und Inkrement-Tasten aufweist. Die Diagrammkennung gibt an, welches Unterdiagramm zur Zeit angezeigt wird. Bei Case-Strukturen wird als Diagrammkennung eine Liste der Werte aufgeführt, mit denen das Unterdiagramm ausgewählt wird. Bei Sequenzstrukturen gibt die Diagrammkennung die Nummer des Rahmens in der Sequenz an (0 bis $n - 1$). Die folgende Abbildung zeigt eine Case-Struktur und eine Sequenzstruktur.



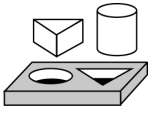
Durch Klicken auf die Dekrement- (links) oder Inkrement-Taste (rechts) wird das vorherige bzw. das nächste Unterdiagramm angezeigt. Wenn Sie im letzten Unterdiagramm auf die Inkrement-Taste klicken, wird das erste Unterdiagramm angezeigt; bei Wahl der Dekrement-Taste im ersten Unterdiagramm wird das letzte Unterdiagramm angezeigt. Weitere Informationen zu Case- und Sequenzstrukturen finden Sie in Kapitel 19, *Strukturen*, im *Referenzhandbuch zur Programmierung in G*.

Case-Struktur

Die Case-Struktur enthält zwei oder mehr Unterdiagramme, die auch als *Cases* bezeichnet werden. Bei jedem Ausführen der Struktur wird jeweils eines dieser Unterdiagramme ausgeführt. Entscheidend dafür ist der Integer-, Boolesche, String- oder Enum-Wert, den Sie mit der externen Seite des Auswahlterminals oder *Wählers* verbinden. Die folgende Abbildung zeigt eine Case-Struktur.



☞ **Hinweis** ***Case-Aussagen in anderen Programmiersprachen führen in der Regel einen Case-Befehl nicht aus, wenn der Case außerhalb des gültigen Bereichs liegt. In G müssen Sie einen Standard-Case angegeben, der Werte außerhalb des gültigen Case-Bereichs bearbeitet, oder Sie müssen jeden möglichen Eingangswert aufführen.***

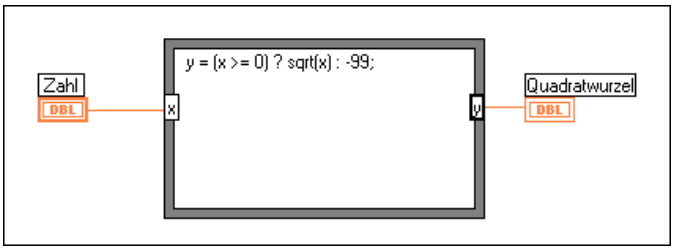


Übung 4-1. Case-Struktur verwenden

Übungsziel ist das Erstellen eines VIs, das eine Zahl überprüft und feststellt, ob sie positiv ist. Wenn die Zahl positiv ist, berechnet das VI die Quadratwurzel für die Zahl; andernfalls zeigt das VI einen Fehler an.

Frontpanel

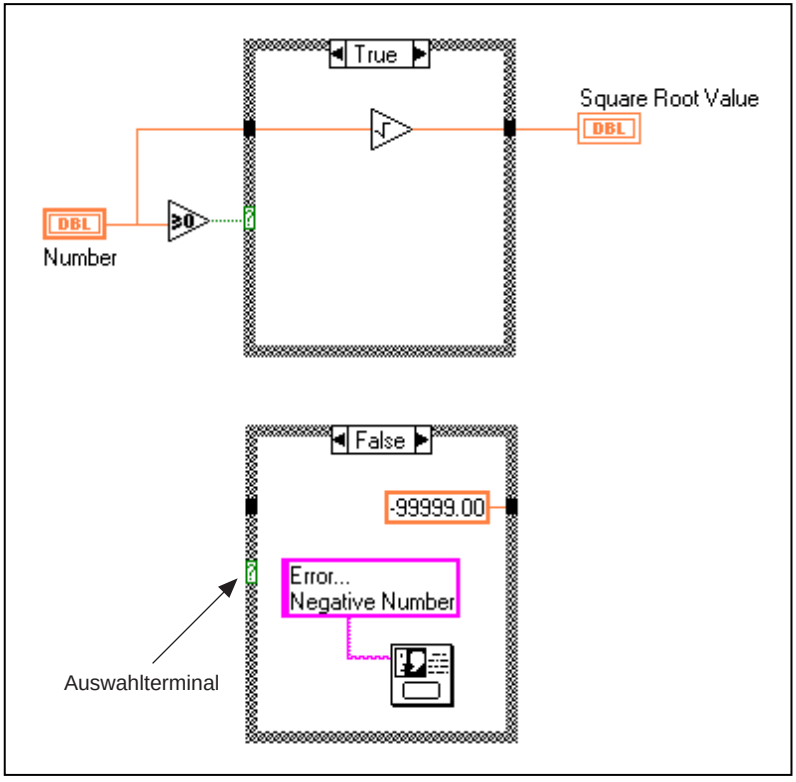
1. Öffnen Sie ein neues Frontpanel, und erstellen Sie die Objekte, die in der folgenden Abbildung dargestellt sind.



Das Bedienelement `Zahl` gibt die Zahl aus. Das Anzeigeelement `Quadratwurzelwert` zeigt die Quadratwurzel für die Zahl an. Das freie Beschriftungsfeld kann für Hinweise für den Benutzer verwendet werden.

Blockdiagramm

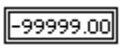
2. Erstellen Sie das in der folgenden Abbildung gezeigte Diagramm.



3. Plazieren Sie eine Case-Struktur im Blockdiagramm, indem Sie sie über **Funktionen»Strukturen** auswählen. Die Case-Struktur ist ein in der Größe veränderbares Feld, das nicht sofort auf dem Diagramm eingesetzt wird. Sie haben stattdessen die Möglichkeit, ihre Position und Größe zu verändern. Klicken Sie dazu in einen Bereich links über allen Terminals, die Sie in die Case-Struktur aufnehmen möchten. Halten Sie die Maustaste gedrückt, und ziehen Sie ein Rechteck um die Terminals.

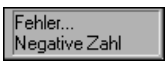
Größer oder gleich 0? (**Funktionen»Vergleichsoperationen**) — Gibt WAHR aus, falls die eingegebene Zahl größer oder gleich 0 ist.

Quadratwurzel (**Funktionen»Numerisch**)—Gibt die Quadratwurzel für die eingegebene Zahl aus.



Numerische Konstante (**Funktionen»Numerisch**)—In dieser Übung gibt die Konstante den numerischen Wert des Fehlers an.

Ein Dialogfeld mit Schaltfläche (**Funktionen»Zeit & Dialog**)—In dieser Übung wird durch diese Funktion ein Dialogfeld aufgerufen, das die Meldung `Fehler...Negative Zahl` enthält.



String-Konstante (**Funktionen»String**) — Geben Sie den gewünschten Text mit dem Beschriftungswerkzeug in das Feld ein.

Das VI führt entweder den Case WAHR oder FALSCH aus. Falls die Zahl größer oder gleich Null ist, führt das VI den Case WAHR aus und gibt die Quadratwurzel für die Zahl aus. Für den Case FALSCH wird der Wert –99999.00 ausgegeben und ein Dialogfeld mit der Meldung `Fehler...Negative Zahl` angezeigt.

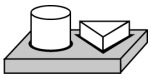
Hinweis ***Sie müssen für jeden Case den Ausgangstunnel definieren. Wenn Sie einen Ausgangstunnel in einem Case erstellen, erscheint auch in allen anderen Cases an derselben Stelle ein Tunnel. Ein Tunnel, zu dem keine Verbindung hergestellt wurde, wird als weißes Quadrat angezeigt.***

4. Kehren Sie zum Frontpanel zurück, und führen Sie das VI aus. Führen Sie jeweils einen Test mit einer Zahl durch, die größer bzw. kleiner als Null ist. Ändern Sie dazu den Wert in dem numerischen Eingabefeld, das Sie mit `Zahl` beschriftet haben. Beachten Sie, daß LabVIEW bei Eingabe einer negativen Zahl im numerischen Eingabefeld die Fehlermeldung anzeigt, die Sie für den Case FALSCH in der Case-Struktur eingegeben haben.
5. Speichern Sie das VI unter dem Namen `Square Root.vi` im Verzeichnis `LabVIEW\Activity`.

VI-Logik

Das Blockdiagramm in dieser Übung hat denselben Effekt wie der folgende Pseudocode in einer auf Text basierenden Programmiersprache.

```
if (Number >= 0) then
Square Root Value = SQRT(Number)
else
Square Root Value = -99999.00
Display Message "Fehler...Negative Zahl"
end if
```

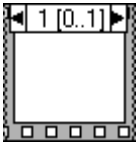


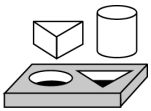
Ende der Übung 4-1.

Sequenzstrukturen

Eine Sequenzstruktur, die wie die Einzelbilder eines Films aussieht, führt Blockdiagramme nacheinander aus. In konventionellen Programmiersprachen werden die Programmaussagen in der Reihenfolge ausgeführt, in der sie vorkommen. Bei der Datenflußprogrammierung wird ein Knoten dann ausgeführt, wenn Daten an allen Knoteneingängen verfügbar sind. Manchmal ist es jedoch notwendig, einen Knoten vor einem anderen auszuführen. G benutzt die Sequenzstruktur, um die Reihenfolge, in der die Knoten ausgeführt werden, zu steuern. G führt zuerst das Diagramm aus, das sich im Rahmen 0 befindet, danach das Diagramm in Rahmen 1 und so weiter. Wie auch bei der Case-Struktur ist immer nur ein Rahmen sichtbar.

Die folgende Abbildung zeigt eine Sequenzstruktur.



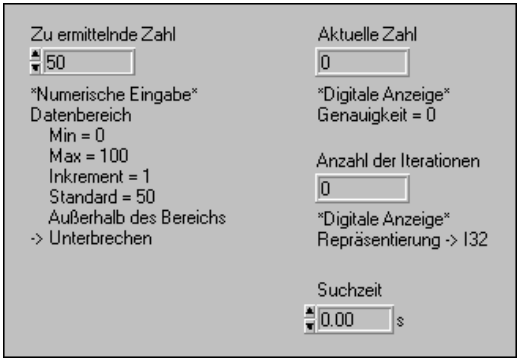


Übung 4-2. Sequenzstruktur verwenden

Übungsziel ist das Erstellen eines VIs, das die Zeit berechnet, die zum Generieren einer mit einer vorgegebenen Zahl identischen Zufallszahl benötigt wird.

Frontpanel

1. Öffnen Sie ein neues Frontpanel, und stellen Sie ein Frontpanel zusammen, das der folgenden Abbildung entspricht. Achten Sie darauf, daß Sie die Bedien- und Anzeigeelemente so verändern, wie im Anschluß an die Illustration beschrieben wird.



Das Bedienelement `Zu ermittelnde Zahl` enthält die Zahl, mit der die Zufallszahl identisch sein soll. Das Anzeigeelement `Aktuelle Zahl` zeigt die aktuelle Zufallszahl an. Das Anzeigeelement `Anzahl der Iterationen` zeigt die Anzahl der Iterationen an, die notwendig sind, um eine identische Zahl zu finden. `Suchzeit` gibt an, wie viele Sekunden notwendig waren, um eine identische Zahl zu finden.

Numerisches Format verändern

LabVIEW zeigt standardmäßig die Werte in numerischen Bedienelementen in Dezimalform mit zwei Dezimalstellen (z.B. 3,14) an. Mit Hilfe der Option **Format & Präzision…** im Popup-Menü für ein Bedien- oder Anzeigeelement können Sie die Präzision ändern oder die numerischen Bedien- und Anzeigeelemente in wissenschaftlicher oder technischer Notation anzeigen. Sie können die Option **Format & Präzision…** auch zum Festlegen eines Zeit- und Datumformats für numerische Werte verwenden.

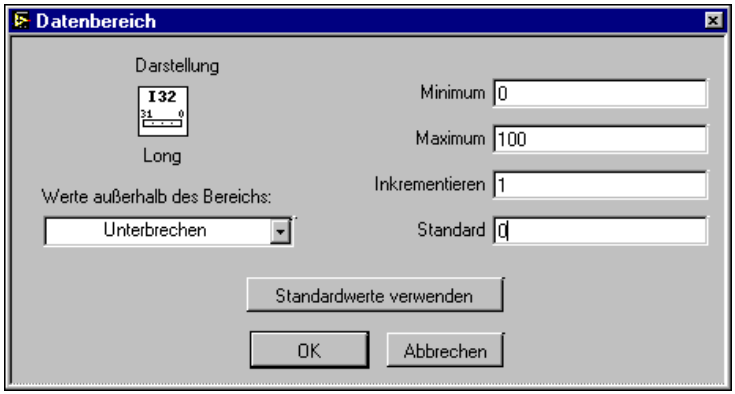
2. Rufen Sie das Popup-Menü für das Anzeigeelement “Suchzeit” auf, und wählen Sie **Format & Präzision…**. Das Frontpanel muß als aktives Fenster gewählt sein, damit Sie auf das Menü zugreifen können.
3. Geben Sie 3 im Feld “Nachkommastellen” ein, und klicken Sie auf **OK**.

4. Rufen Sie das Popup-Menü für das numerische Eingabefeld “Zu ermittelnde Zahl” auf, und wählen Sie **Repräsentierung»I32**.
5. Wiederholen Sie Schritt 4 für die digitalen Anzeigeelemente “Aktuelle Zahl” und “Anzahl der Iterationen”.

Datenbereich einstellen

Mit der Option “Datenbereich…” können Sie verhindern, daß ein Benutzer einen Wert für ein Bedien- oder Anzeigeelement einstellt, der außerhalb des festgelegten Bereichs oder Inkrementwerts liegt. Es wird die Option geboten, den Wert zu ignorieren, ihn in einen gültigen Wert umzuwandeln oder die Ausführung zu unterbrechen. Das Bereichsfehlersymbol wird anstelle der Taste “Ausführen” in der Symbolleiste gezeigt, wenn ein Bereichsfehler zu einer Ausführungsunterbrechung führt. Außerdem wird das Bedienelement, das außerhalb des Bereichs liegt, von einem durchgezogenen schwarzen Rahmen umgeben.

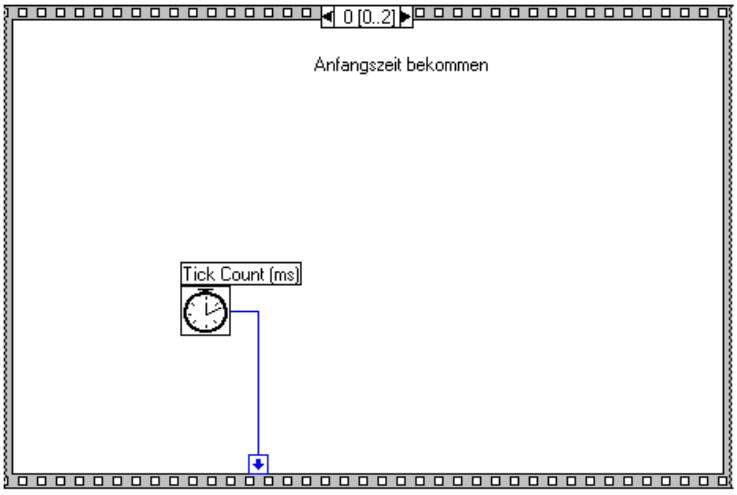
6. Rufen Sie das Popup-Menü für das Anzeigeelement “Zu ermittelnde Zahl” auf, und wählen Sie **Datenbereich…**.
7. Tragen Sie die in der folgenden Abbildung gezeigten Angaben in das Dialogfeld ein, und klicken Sie auf **OK**.

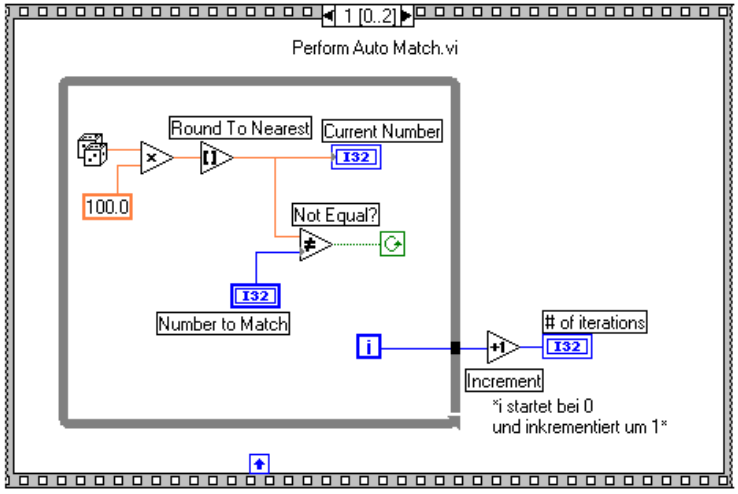


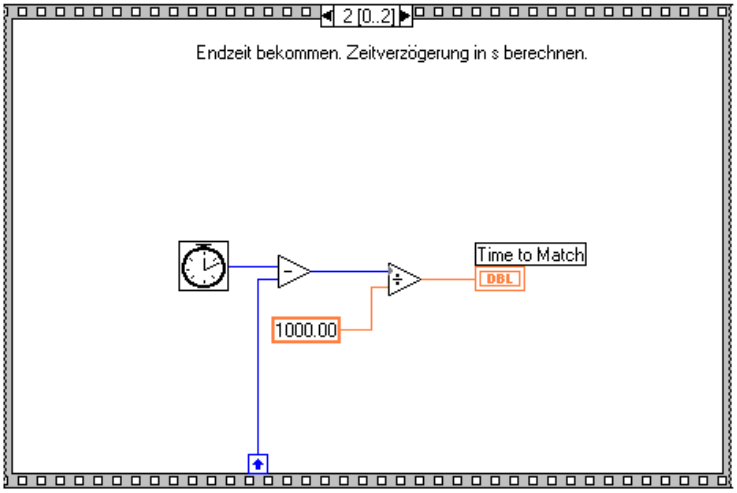
Blockdiagramm

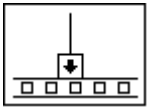
8. Öffnen Sie das Blockdiagramm.
9. Plazieren Sie die Sequenzstruktur (**Funktionen»Strukturen**) im Blockdiagramm.
10. Vergrößern Sie die Struktur, indem Sie mit dem Skalierwerkzeug an einer Ecke ziehen.
11. Erstellen Sie einen neuen Rahmen, indem Sie das Popup-Menü für die Umrandung aufrufen und **Rahmen danach einfügen** wählen. Wiederholen Sie diesen Schritt, um den zweiten Rahmen zu erstellen.

12. Stellen Sie das in den folgenden Abbildungen gezeigte Blockdiagramm zusammen.









Der Rahmen 0 in der vorherigen Abbildung enthält ein kleines Feld mit einem Pfeil. Dieses Feld ist eine *Sequenz-Variable*, die Daten zwischen den Rahmen einer Sequenzstruktur befördert. Sie können Sequenz-Variablen auf dem Rand eines Rahmens erstellen. Die Daten, die mit einer Rahmen-Sequenz-Variablen verbunden werden, stehen daraufhin auch in den nachfolgenden Rahmen zur Verfügung. Sie können jedoch nicht auf die Daten in den Rahmen zugreifen, die vor dem Rahmen liegen, in dem Sie die Sequenz-Variable erstellt haben.

13. Erstellen Sie die Sequenz-Variable, indem Sie das Popup-Menü für den unteren Rand des Rahmens 0 aufrufen und **Lokale Sequenz-Variable hinzufügen** wählen.

Die Sequenz-Variable wird als leeres Quadrat angezeigt. In diesem Quadrat erscheint automatisch ein Pfeil, wenn Sie eine Funktion mit der Sequenz-Variablen verbinden.

14. Stellen Sie das Blockdiagramm entsprechend der ersten Abbildung des Abschnitts *Blockdiagramm* in dieser Übung fertig.

Tick Count (ms) (**Funktionen»Zeit & Dialog**)—Gibt die Anzahl der Millisekunden aus, die seit dem Einschalten verstrichen sind. In dieser Übung werden zwei Tick Count-Funktionen benötigt.

Zufallszahl (0–1) (**Funktionen»Numerisch**) — Gibt eine Zufallszahl zwischen 0 und 1 aus.

Multiplizieren (**Funktionen»Numerisch**) — In dieser Übung wird die Zufallszahl durch diese Funktion mit 100 multipliziert.

Numerische Konstante (**Funktionen»Numerisch**)—In dieser Übung steht die numerische Konstante für die größtmögliche Zahl, die multipliziert werden kann.

Auf nächste ganze Zahl runden (**Funktionen»Numerisch**)—In dieser Übung werden Zufallszahlen zwischen 0 und 100 auf die nächste ganze Zahl gerundet.

Nicht gleich? (**Funktionen»Vergleichsoperationen**)—In dieser Übung wird durch diese Funktion die Zufallszahl mit der Zahl verglichen, die im Frontpanel angegeben wurde. Wenn die Zahlen nicht gleich sind, wird WAHR ausgegeben, andernfalls FALSCH.

Inkrement (**Funktionen»Numerisch**)—In dieser Übung wird die Zählung der While-Schleife durch diese Funktion um das Inkrement 1 erhöht.

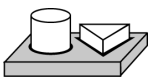
Subtrahieren (**Funktionen»Numerisch**)—In dieser Übung wird durch diese Funktion die Zeit (in Millisekunden) ausgegeben, die zwischen Rahmen 2 und Rahmen 0 verstrichen ist.

Dividieren (**Funktionen»Numerisch**)—In dieser Übung wird durch diese Funktion die Zahl für die verstrichenen Millisekunden durch 1000 geteilt, um die Zahl in Sekunden umzuwandeln.

Numerische Konstante (**Funktionen»Numerisch**)—In dieser Übung werden durch diese Funktion Millisekunden in Sekunden umgewandelt.

Im Rahmen 0 gibt die Funktion “Tick Count (ms)” die aktuelle Zeit in Millisekunden aus. Dieser Wert wird mit der Sequenz-Variablen verbunden, über die der Wert auch für die nachfolgenden Rahmen zur Verfügung steht. In Rahmen 1 führt das VI die While-Schleife so lange aus, bis die angegebene Zahl mit der Zahl übereinstimmt, die durch die Funktion “Zufallszahl (0–1)” ausgegeben wird. In Rahmen 2 gibt die Funktion “Tick Count (ms)” eine neue Zeitangabe in Millisekunden aus. Das VI subtrahiert die alte (von Rahmen 0 durch die Sequenz-Variable geleitete) Zeit von der neuen Zeit, um die verstrichene Zeit zu ermitteln.

15. Kehren Sie zum Frontpanel zurück. Geben Sie eine Zahl im Bedienelement `Zu ermittelnde Zahl` ein, und führen Sie das VI aus.
16. Speichern Sie das VI unter dem Namen `Time to Match.vi` im Verzeichnis `LabVIEW\Activity`.



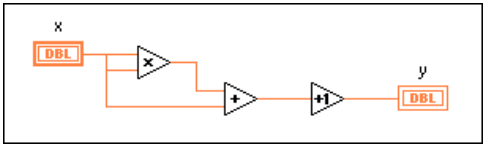
Ende der Übung 4-2.

Formel-Knoten

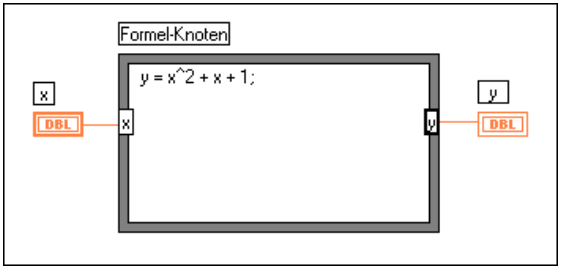
Der Formel-Knoten ist ein in der Größe veränderbares Feld, das zur direkten Eingabe von Formeln in ein Blockdiagramm verwendet werden kann. Sie können den Formel-Knoten auf dem Blockdiagramm plazieren, indem Sie ihn über **Funktionen»Strukturen** auswählen. Diese Funktion ist besonders dann nützlich, wenn eine Gleichung viele Variablen hat oder aus anderen Gründen sehr komplex ist. Sehen Sie sich z.B. die folgende Gleichung an:

$y = x^2 + x + 1$

Wenn Sie diese Gleichung mit den regulären arithmetischen Funktionen von G implementieren, sieht das Blockdiagramm wie in der folgenden Abbildung aus.

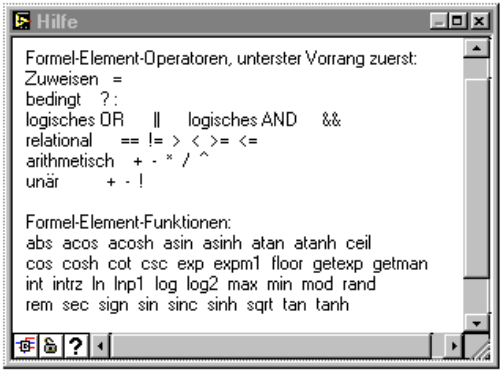


Wie die folgende Abbildung zeigt, können Sie dieselbe Gleichung auch mit einem Formel-Knoten implementieren.



Mit dem Formel-Knoten können Sie eine komplexe Formel bzw. mehrere komplexe Formeln direkt eingeben, ohne Blockdiagramm-Unterabschnitte zu erstellen. Die Eingabe erfolgt mit dem Beschriftungswerkzeug. Sie erstellen die Eingangs- und Ausgangsterminals des Formel-Knotens, indem Sie das Popup-Menü für den Rand des Knotens aufrufen und “Eingang hinzufügen” (bzw. “Ausgang hinzufügen”) wählen. Geben Sie den Variablennamen in das Feld ein. Dabei müssen Sie die Groß-/Kleinschreibung beachten. Geben Sie die Formel/n direkt in das Feld ein. Jede Formelaussage muß durch ein Semikolon abgeschlossen werden (;).

Die im Formel-Knoten verfügbaren Operatoren und Funktionen sind im Hilfefenster für den Formel-Knoten aufgeführt, das in der folgenden Abbildung dargestellt ist. Das Ende jeder Formelaussage wird durch ein Semikolon gekennzeichnet.

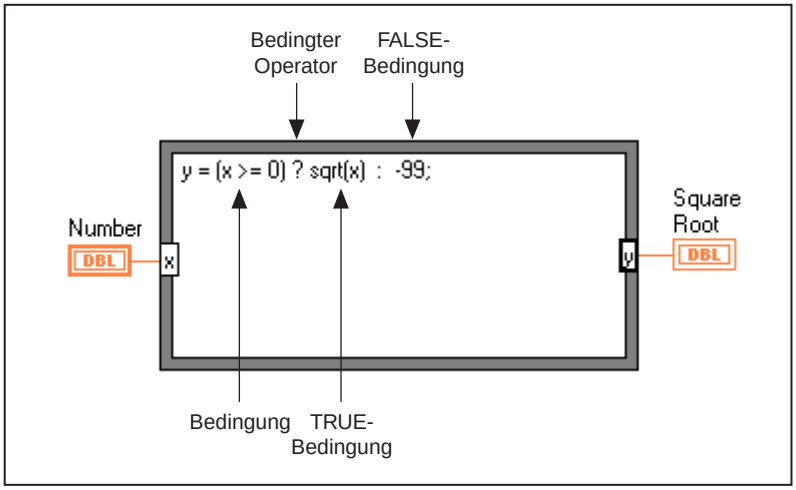


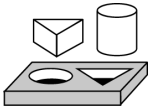
Das folgende Beispiel zeigt, wie Sie eine Bedingungszuordnung in einem Formel-Knoten durchführen können.

Sehen Sie sich das folgende Codefragment an, das die Quadratwurzel von x berechnet, wenn x einen positiven Wert hat, und das Ergebnis y zuordnet. Wenn x negativ ist, ordnet der Code y den Wert -99 zu.

```
if (x >= 0) then
y = sqrt(x)
else
y = -99
end if
```

Sie können das Codefragment, wie in der folgenden Abbildung gezeigt, mit einem Formel-Knoten implementieren.





Übung 4-3. Formel-Knoten verwenden

Übungsziel ist das Erstellen eines VIs, das den Formel-Knoten verwendet, um die folgenden Gleichungen zu berechnen.

$y1 = x^3 - x^2 + 5$

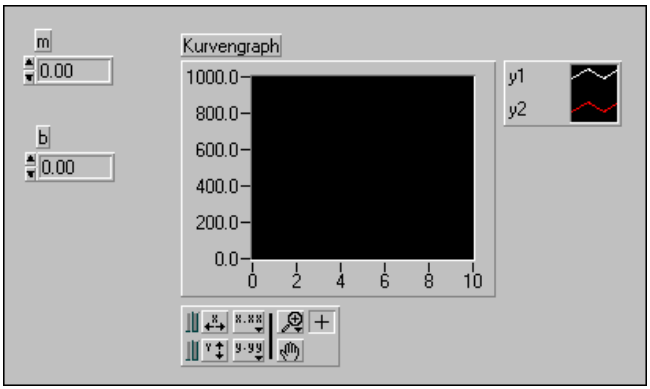
$y2 = m * x + b$

Dabei liegt `x` zwischen 0 und 10.

Sie verwenden dabei nur einen Formel-Knoten für beide Gleichungen und übertragen die Ergebnisse auf denselben Graphen. Weitere Informationen zu Graphen finden Sie in Kapitel 5, *Arrays, Cluster und Graphen*.

Frontpanel

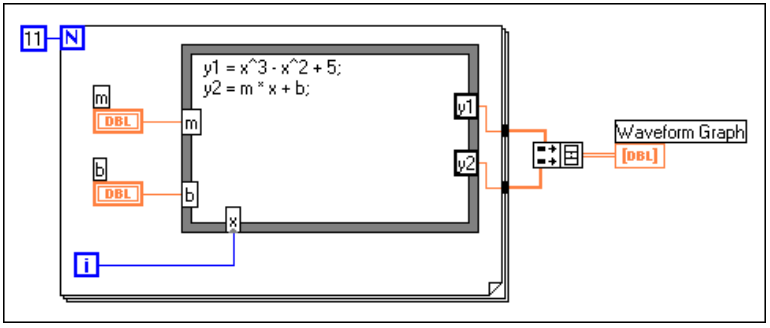
1. Öffnen Sie ein neues Frontpanel, und stellen Sie das in der folgenden Abbildung gezeigte Frontpanel zusammen. Das Kurvengraph-Anzeigeelement zeigt die Plots für die Gleichungen an. Das VI verwendet zwei numerische Eingabefelder als Eingänge für die Werte für `m` und `b`.

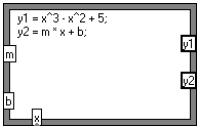


2. Erstellen Sie die in der folgenden Abbildung gezeigte Historie des Graphen, indem Sie **Anzeigen»Legende** wählen. Verwenden Sie den Skaliercursor, um die Legende nach unten zu vergrößern, so daß zwei Plots angezeigt werden können. Benennen Sie die Plots mit dem Beschriftungswerkzeug um. Über das Popup-Menü für die Legende können Sie für jeden Plot den Linienstil definieren. Sie können die einzelnen Plots auch farbig gestalten, indem Sie die Legende für die Plots mit dem Farbenwerkzeug bearbeiten.

Blockdiagramm

3. Erstellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm.





Formel-Knoten (**Funktionen»Strukturen**). Mit diesem Knoten können Sie Formeln direkt eingeben. Erstellen Sie die drei Eingabeterminals, indem Sie das Popup-Menü für den Rand aufrufen und **Eingang hinzufügen** wählen. Zum Erstellen des Ausgangsterminals können Sie **Ausgang hinzufügen** im Popup-Menü wählen.

Wenn Sie ein Eingangs- oder Ausgangsterminal erstellen, müssen Sie ihm einen Variablennamen zuweisen. Der Variablenname muß mit dem in der Formel verwendeten Namen genau übereinstimmen. Dabei ist die Groß-/Kleinschreibung zu beachten. Wenn Sie also bei der Benennung des Terminals ein kleines `a` verwenden, müssen Sie auch in der Formel ein kleines `a` verwenden. Sie können die Variablennamen und die Formel mit dem Beschriftungswerkzeug eingeben.

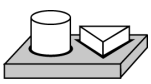
Hinweis ***Variablennamen sind zwar nicht in der Länge beschränkt, aber Sie sollten bedenken, daß lange Namen beträchtlichen Platz im Diagramm beanspruchen. Das Ende einer Formelaussage wird durch ein Semikolon (;) gekennzeichnet.***

Numerische Konstante (**Funktionen»Numerisch**). Sie können das Popup-Menü auch für das Zählterminal aufrufen und **Konstante erzeugen** wählen, um die numerische Konstante automatisch zu erstellen und zu verbinden. Die numerische Konstante bestimmt die Anzahl der Iterationen der For-Schleife. Wenn `x` den Bereich von 0 bis einschließlich 10 abdeckt, müssen Sie 11 mit dem Zählterminal verbinden.

Da das Iterationsterminal von 0 bis 10 zählt, können Sie es zur Steuerung des `x`-Werts im Formel-Knoten verwenden.

“Array erstellen” (**Funktionen»Array**) werden zwei Eingaben in einem Mehrfachplot zusammengefaßt. Erstellen Sie die beiden Eingangsterminals, indem Sie mit dem Skalierwerkzeug an einer der Ecken ziehen. Weitere Informationen zu Arrays finden Sie in Kapitel 5, *Arrays, Cluster und Graphen*.

4. Kehren Sie zum Frontpanel zurück, und führen Sie das VI mit anderen Werten für `m` und `b` aus.
5. Speichern Sie das VI unter dem Namen `Equations.vi` im Verzeichnis `LabVIEW/Activity`.



Ende der Übung 4-3.

Künstliche Datenabhängigkeit

Knoten, zu denen keine Verbindung hergestellt wurde, können in beliebiger Reihenfolge ausgeführt werden. Das Ausführen von Knoten erfolgt nicht immer von links nach rechts und von oben nach unten. Wenn keine natürliche Datenabhängigkeit gegeben ist, bieten Sequenzstrukturen eine Möglichkeit, die Reihenfolge bei der Ausführung zu steuern.

Eine andere Möglichkeit, die Ausführreihenfolge zu steuern, besteht darin, eine künstliche Datenabhängigkeit zu erzeugen, d.h., eine Bedingung zu schaffen, unter der die Ausführung eines Objekts nicht durch einen Wert, sondern durch das Eintreffen von Daten ausgelöst wird. Unter Umständen verarbeitet der Empfänger dabei die Daten überhaupt nicht intern. Der Vorteil dieser künstlichen Abhängigkeit besteht darin, daß alle Knoten auf einer Ebene sichtbar sind. Nachteilig ist, daß die künstlichen Verbindungen zwischen den Knoten in manchen Fällen verwirrend wirken können.

Sie können das `Timing Template (data dep).vi` in `Examples\General\structs.llb` öffnen, um zu sehen, wie die `Timing-Vorlage` verändert wurde, um künstliche Datenabhängigkeit anstelle einer Sequenzstruktur zu verwenden.

5

Arrays, Cluster und Graphen

In diesem Kapitel werden die Grundkonzepte des Polymorphismus, von Arrays, Clustern und Graphen erklärt. Außerdem enthält dieses Kapitel Übungen, durch die die Auto-Indizierung und die Graphen- und Analyse-VIs illustriert werden.

Arrays

Ein Array ist eine Ansammlung von Datenelementen, die alle demselben Typ angehören. Ein Array verfügt über eine oder mehrere Dimensionen und enthält bis zu $2^{31} - 1$ Elemente pro Dimension, sofern der Speicher dafür ausreicht. Sie können auf jedes Array-Element über seinen Index zugreifen. Der Index deckt den Bereich von 0 bis $n - 1$ ab, wobei n der Anzahl der Elemente im Array entspricht. Das folgende 1D-Array numerischer Werte illustriert diese Struktur. Beachten Sie, daß das erste Element den Index 0 aufweist, das zweite Element den Index 1 und so weiter.

Index	0	1	2	3	4	5	6	7	8	9
Array mit 10 Elementen	1.2	3.2	8.2	8.0	4.8	5.1	6.0	1.0	2.5	1.7

Wie werden Arrays erstellt und initialisiert?

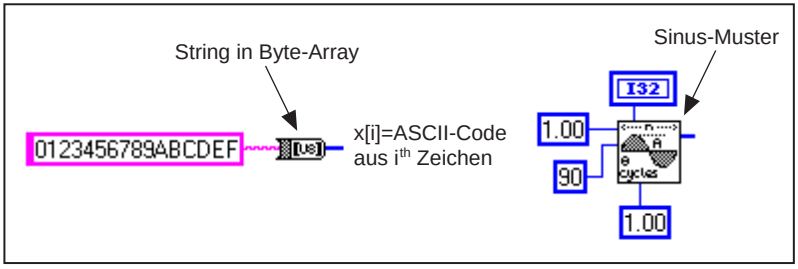
Wenn Sie ein Array als Datenquelle in einem Blockdiagramm benötigen, können Sie **Funktionen»Array** und dann die Array-Shell wählen, um sie auf dem Blockdiagramm zu plazieren. Mit dem Bedienwerkzeug können Sie eine numerische Konstante, eine Boolesche Konstante oder eine String-Konstante wählen, die in das leere Array eingesetzt werden soll. Die folgende Abbildung zeigt ein Beispiel für eine Array-Shell mit einer numerischen Konstanten, die in die Array-Shell eingefügt wurde.

Wählen Sie zum Erstellen eines Arrays auf dem Frontpanel die Option **Array & Cluster** in der Palette **Elemente**, und plazieren Sie die Array-Shell auf dem Frontpanel. Wählen Sie dann ein Objekt aus (z.B. einen numerischen Wert), und plazieren Sie es in der Array-Shell. Dadurch wird ein Array mit numerischem Wert erstellt.

☞ **Hinweis** ***Sie können ein Array und das dazugehörige Bedienelement auch auf dem Frontpanel erstellen und das Array-Bedienelement anschließend in das Blockdiagramm kopieren oder ziehen, um eine dazugehörige Konstante zu erstellen.***

Weitere Informationen zum Erstellen von Bedien- und Anzeigeelementen für Arrays auf dem Frontpanel finden Sie in Kapitel 14, *Bedien- und Anzeigeelemente vom Typ Array und Cluster*, im *Referenzhandbuch zur Programmierung in G*.

Es gibt verschiedene Möglichkeiten, Arrays auf dem Blockdiagramm zu erstellen und zu initialisieren. Wie die folgende Abbildung zeigt, dienen einige Blockdiagrammfunktionen ebenfalls zum Anfertigen von Arrays.



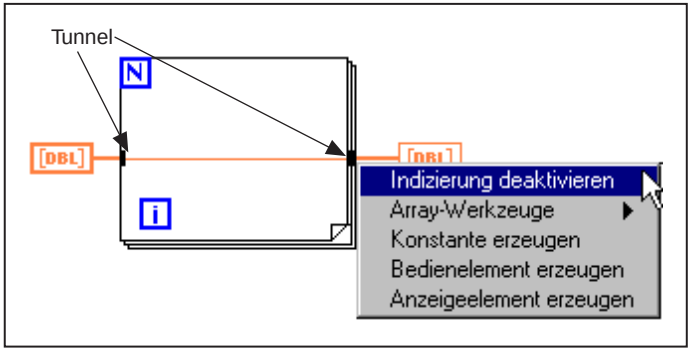
Bedienelemente, Konstanten und Anzeigen für Arrays

Sie können Bedienelemente, Konstanten und Anzeigen für Arrays auf dem Frontpanel erstellen, indem Sie eine Array-Shell mit einem numerischen Wert, einem Booleschen Wert, einem String oder einem Cluster kombinieren. Ein anderes Array, ein Diagramm oder ein Graph können nicht als Array-Element verwendet werden. Sie finden Beispiele für Arrays in `Examples\General\arrays.llb`.

Auto-Indizierung

For-Schleifen- und While-Schleifenstrukturen können Arrays automatisch an Ihren Rändern indizieren und akkumulieren. Diese Fähigkeit wird mit dem Sammelbegriff *Auto-Indizierung* bezeichnet. Wenn Sie die Auto-Indizierung aktivieren und ein Array einer beliebigen Dimension von einem externen Knoten mit einem Eingangstunnel auf dem Schleifenrand verbinden, geht eine Komponente nach der anderen, beginnend mit der ersten Komponente, in die Schleife über. Die Schleife indiziert skalare Elemente von 1D Arrays, 1D Arrays von 2D Arrays und so weiter. Der umgekehrte Vorgang läuft an Ausgangstunneln ab. Dort akkumulieren die Elemente in sequentieller Folge in 1D Arrays, 1D Arrays akkumulieren in 2D Arrays und so weiter.

☞ **Hinweis** ***Auto-Indizierung wird standardmäßig für alle Arrays verwendet, die mit einer For-Schleife verbunden sind. Sie können die Auto-Indizierung deaktivieren, indem Sie das Popup-Menü für den Tunnel (Übergangspunkt des Eingabearrays) aufrufen und* Indizierung deaktivieren *wählen.***



In der Standardeinstellung ist die Auto-Indizierung für alle Arrays, die mit einer While-Schleife verbunden sind, deaktiviert. Rufen Sie das Popup-Menü für den Array-Tunnel einer While-Schleife auf, um die Auto-Indizierung zu aktivieren.

Übung 5-1. Array mit Auto-Indizierung erstellen

Übungsziel ist das Erstellen eines Arrays mit Hilfe der Auto-Indizierungsfunktion einer For-Schleife und das Zeichnen des Arrays in einem Kurvengraphen.

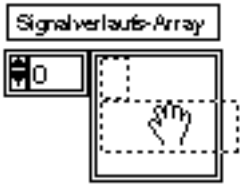
Sie stellen ein VI zusammen, das ein Array mit Hilfe des VIs zum Erzeugen eines Signalverlaufs generiert und das Array in einem Kurvengraphen zeichnet. Außerdem führen Sie Veränderungen am VI durch, damit mehrere Plots gezeichnet werden können.

Frontpanel

1. Öffnen Sie ein neues Frontpanel.

2. Plazieren Sie eine Array-Shell über **Elemente»Array & Cluster** im Frontpanel. Beschriften Sie die Array-Shell mit `Signalverlaufs-Array`.

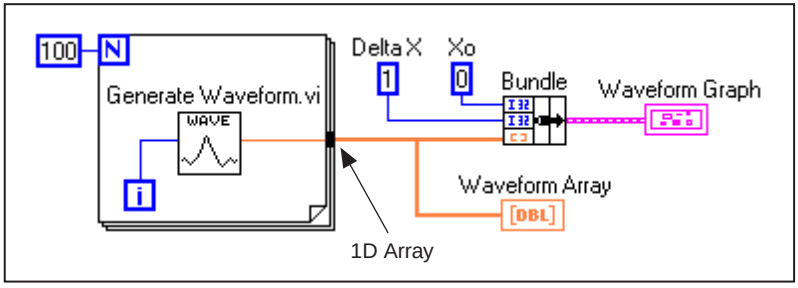
3. Plazieren Sie ein digitales Anzeigeelement über **Elemente»Numerisch** in der Elementanzeige der Array-Shell, wie in der folgenden Abbildung gezeigt. Dieses Anzeigeelement zeigt den Inhalt des Arrays an.



4. Plazieren Sie einen Kurvengraphen über **Elemente»Graph** im Frontpanel. Beschriften Sie den Graphen mit `Kurvengraph`.
5. Vergrößern Sie den Graphen, indem Sie mit dem Skaliercursor an einer Ecke ziehen.
6. Blenden Sie die Legende und Palette aus.
7. Deaktivieren Sie die automatische Skalierung, indem Sie das Popup-Menü für den Graphen aufrufen und die Markierung von der Option **Y-Maßstab»Automatische Skalierung Y** entfernen.
8. Verwenden Sie das Textwerkzeug, um die Y-Achse für den Bereich `-0,5` bis `1,5` neu zu skalieren.

Blockdiagramm

9. Stellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm zusammen.



VI zum Erzeugen eines Signalverlaufs (**Funktionen»VI auswählen…** im Verzeichnis `LabVIEW\Activity`) — Gibt einen Punkt eines Signalverlaufs aus. Das VI setzt eine skalare Index-Eingabe voraus. Sie sollten also das Terminal für die Schleifen-Iteration mit diesem Eingang verbinden.

Beachten Sie, daß die Verbindung vom VI zum Erzeugen eines Signalverlaufs dicker wird, während das Array am Schleifenrand entsteht.

Die For-Schleife akkumuliert die Arrays automatisch am Schleifenrand. Dieser Vorgang wird als Auto-Indizierung bezeichnet. In diesem Fall bewirkt die numerische Konstante, die mit dem numerischen Eingang für die Schleifenzählung verbunden ist, daß die For-Schleife ein Array mit 100 Elementen (indiziert von 0 bis 99) erstellt.

Elemente bündeln (**Funktionen»Cluster**)—Faßt die Plot-Komponenten in einem Cluster zusammen. Sie müssen das Symbol für die Bündelfunktion in der Größe verändern, bevor Sie es ordnungsgemäß verbinden können. Plazieren Sie das Positionierwerkzeug in der unteren linken Ecke des Symbols. Das Werkzeug verwandelt sich daraufhin in den

Skaliercursor, der in der Abbildung links neben diesem Text zu sehen ist. Sobald sich das Werkzeug geändert hat, können Sie mit der Maustaste klicken und den Cursor nach unten ziehen, bis ein drittes Eingangsterminal erscheint. Sie können nun damit fortfahren, das Blockdiagramm entsprechend der obigen Abbildung zu verbinden.

Numerische Konstante (**Funktionen»Numerisch**)—Drei numerische Konstanten legen die Anzahl der Iterationen der For-Schleife, den Anfangswert für X und den Wert für Delta X fest. Beachten Sie, daß Sie das

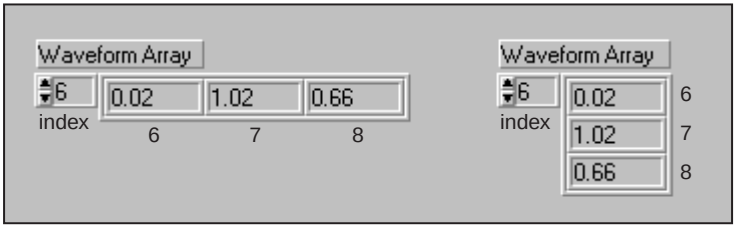
Popup-Menü für das Zählterminal der For-Schleife, das links abgebildet ist, aufrufen können. Wählen Sie “Konstante erzeugen”, um automatisch in diesem Terminal eine numerische Konstante einzufügen und zu verbinden.

10. Führen Sie das VI auf dem Frontpanel aus. Das VI überträgt das automatisch indizierte Signalverlaufs-Array auf den Kurvengraphen. Der Anfangswert für *X* ist 0 und der Wert für Delta *X* ist 1.
11. Ändern Sie den Wert für Delta *X* auf 0,5 und den Anfangswert für *X* auf 20. Führen Sie das VI noch einmal aus.

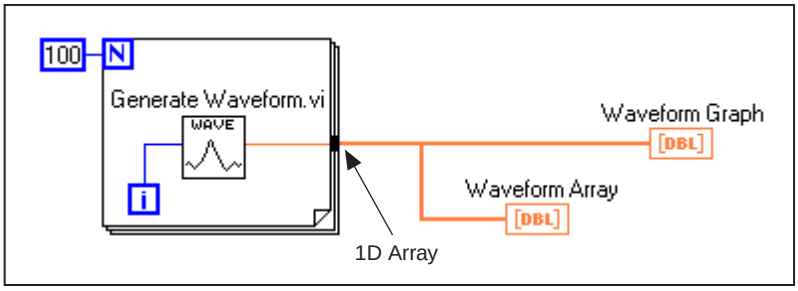
Beachten Sie, daß der Graph nun dieselben 100 Datenpunkte mit einem Anfangswert von 20 und einem Delta *X*-Wert von 0,5 für jeden Punkt anzeigt (siehe die X-Achse). In einem zeitlich begrenzten Test könnte dieser Graph den Daten entsprechen, die im Verlauf von 50 Sekunden bei Beginn nach 20 Sekunden erfaßt werden.

12. Sie können jedes Element im Signalverlaufs-Array anzeigen, indem Sie den Index für das Element in die Index-Anzeige eingeben. Wenn Sie eine Zahl eingeben, die die Array-Größe überschreitet, wird die Anzeige abgeblendet. Dadurch werden Sie darauf hingewiesen, daß kein definiertes Element für diesen Index vorhanden ist.

Wenn Sie mehrere Elemente gleichzeitig einsehen möchten, können Sie das Array-Anzeigeelement in der Größe verändern. Plazieren Sie das Positionierwerkzeug in der unteren linken Ecke des Arrays. Das Werkzeug verwandelt sich daraufhin in das Array-Skalierwerkzeug, das in der Abbildung links neben diesem Text zu sehen ist. Wenn sich das Werkzeug verwandelt hat, können Sie es nach rechts oder gerade nach unten ziehen. Wie in der folgenden Abbildung zu sehen ist, zeigt das Array nun mehrere Elemente in aufsteigender Index-Reihenfolge an. Dabei wird zuerst das Element angezeigt, das dem angegebenen Index entspricht.



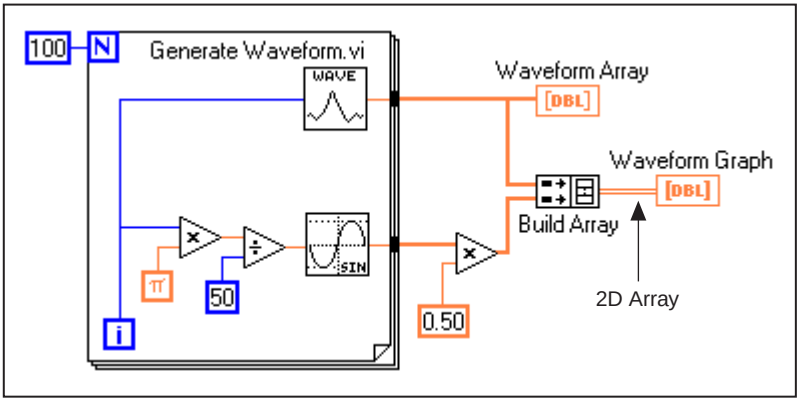
Im vorherigen Blockdiagramm haben Sie für den Signalverlauf einen Anfangswert für X und einen Wert für Delta *X* festgelegt. Der Standardwert für *X* ist Null, und der Standardwert für Delta *X* ist 1. Wie die folgende Abbildung zeigt, können Sie also das Signalverlaufs-Array ohne festgelegten Anfangswert für *X* und festgelegten Wert für Delta *X* direkt mit dem Kurvengraphterminal verbinden.



13. Kehren Sie zum Blockdiagramm zurück. Löschen Sie die Bündelfunktion und die numerischen Konstanten, die damit verbunden sind. Wählen Sie dazu die Funktion und die Konstanten mit dem Positionierwerkzeug aus, und drücken Sie dann auf <Entf>. Wählen Sie **Bearbeiten»Ungültige Verbindungen entfernen**. Stellen Sie die Verbindungen für das Blockdiagramm entsprechend der vorherigen Abbildung fertig.
14. Führen Sie das VI aus. Beachten Sie, daß das VI den Signalverlauf mit dem Anfangswert 0 für *X* und dem Wert 1 für Delta *X* zeichnet.

Multiplot-Graphen

Sie können Multiplot-Kurvengraphen erstellen, indem Sie ein Array des Datentyps zusammenstellen, der normalerweise an einen Graphen mit einem einfachen Plot geleitet wird.



15. Fahren Sie mit dem Erstellen des Blockdiagramms fort, das in der obigen Abbildung dargestellt ist.

Sinus (**Funktionen»Numerisch»Trigonometrisch**)—In dieser Übung verwenden Sie diese Funktion in einer For-Schleife, um ein Array mit Punkten zusammenzustellen, das einem Zyklus einer Sinuswelle entspricht.

Array erstellen (**Funktionen»Array**)—In dieser Übung verwenden Sie diese Funktion, um die korrekte Datenstruktur zu erstellen, mit der zwei Arrays in einem Kurvengraphen (in diesem Fall ein 2D Array) gezeichnet werden. Vergrößern Sie die Funktion “Array erstellen”, um zwei Eingänge einzurichten. Ziehen Sie dazu mit dem Positionierwerkzeug an einer Ecke.

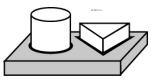
Pi-Konstante (**Funktionen»Numerisch»Zusätzliche numerische Konstanten**)—Denken Sie daran, daß Sie auf die Multiplizier- und Dividierfunktionen über **Funktionen»Numerisch** zugreifen können.

16. Wechseln Sie zum Frontpanel. Führen Sie das VI aus.

Beachten Sie, daß die beiden Signalverläufe auf demselben Kurvengraphen abgebildet werden. In der Standardeinstellung ist in beiden Datensätzen 0 der Anfangswert für *X* und 1 für Delta *X*.

Hinweis ***Sie können das Erscheinungsbild eines Plots auf einem Graphen ändern, indem Sie das Popup-Menü für die Legende des jeweiligen Plots aufrufen. Sie können z.B. von einem Liniengraphen zu einem Balkendiagramm umschalten, indem Sie* Allgemeine Plots»Balkendiagramm *wählen.***

17. Speichern Sie das VI unter dem Namen `Graph Waveform Arrays.vi` im Verzeichnis `LabVIEW\Activity`.



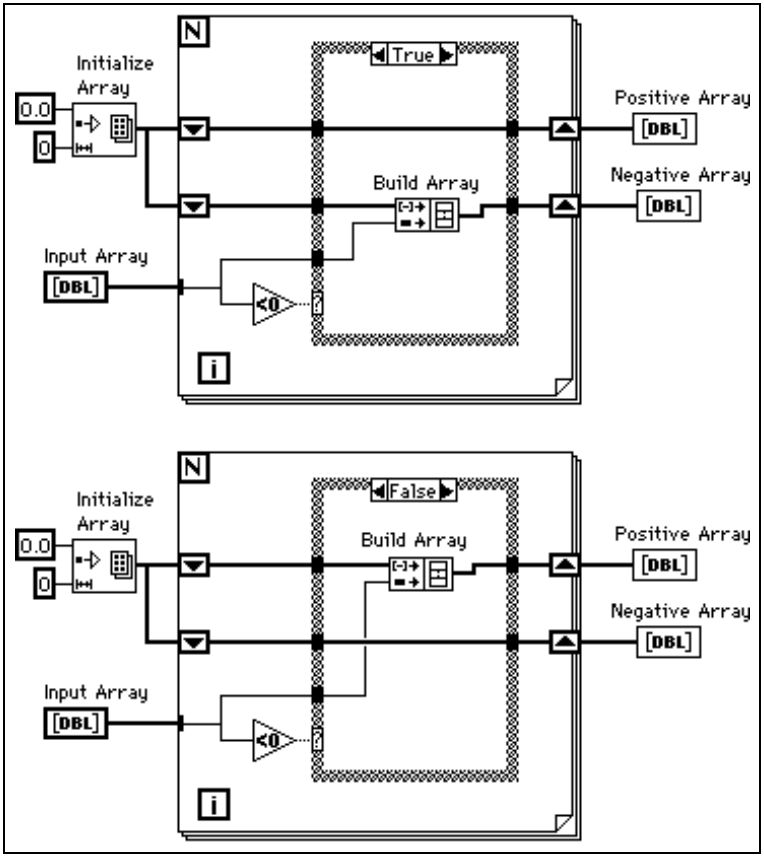
Ende der Übung 5-1.

Im vorherigen Beispiel wurde die For-Schleife 100mal ausgeführt, weil die Konstante `100` mit dem Zählterminal verbunden wurde. Die folgende Übung illustriert ein weiteres Mittel, mit dem Sie festlegen können, wie oft eine Schleife ausgeführt wird.

Übung 5-2. Auto-Indizierung für Eingabearrays

Übungsziel ist das Öffnen und Bedienen eines VIs, das die Auto-Indizierung in einer For-Schleife verwendet, um ein Array zu verarbeiten.

1. Öffnen Sie das VI für separate Array-Werte, indem Sie **Datei» Öffnen…** wählen. Das VI befindet sich in `Examples\General\arrays.llb`.
2. Öffnen Sie das Blockdiagramm. Die folgende Abbildung zeigt das Blockdiagramm mit sichtbaren WAHR- und FALSCH-Cases.

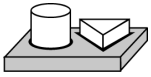


Beachten Sie, daß sich die Verbindung vom `Input Array` in der Dicke verändert. Außerhalb der For-Schleife weist eine dicke Verbindung darauf hin, daß es sich um ein Array handelt. In der Schleife weist eine dünne Verbindung darauf hin, daß es sich um ein einzelnes Element handelt. Das *i-te* Element des Arrays wird automatisch bei jeder Iteration vom Array indiziert.

Auto-Indizierung zum Einstellen der For-Schleifenzählung verwenden

Beachten Sie, daß keine Verbindung zum Zählterminal vorhanden ist. Wenn Sie die Auto-Indizierung auf ein Array anwenden, das in eine Schleife eingeführt wird, erfolgt die Ausführung der Schleife entsprechend der Größe des Arrays. Dadurch wird es überflüssig, einen Wert mit dem Zählterminal zu verbinden. Wenn Sie die Auto-Indizierung für mehrere Arrays verwenden oder zusätzlich zur automatischen Indizierung eines Arrays einen Zählwert festlegen, entspricht die tatsächliche Anzahl der Iterationen der kleinstmöglichen Zahl.

3. Führen Sie das VI aus. Von den acht Eingangswerten befinden sich vier im positiven Array und vier im negativen Array.
4. Verbinden Sie vom Blockdiagramm aus eine Konstante, die den Wert 5 hat, mit dem Zählterminal der For-Schleife. Führen Sie das VI aus. Drei Werte befinden sich danach im positiven Array und zwei im negativen Array, obwohl das Eingabearray nach wie vor acht Elemente enthält. Daran ist zu erkennen, daß immer die kleinere Zahl als tatsächliche Anzahl der Schleifeniterationen verwendet wird, wenn N eingestellt ist und die Indizierung automatisch durchgeführt wird.
5. Schließen Sie das VI, und speichern Sie die Änderungen nicht.

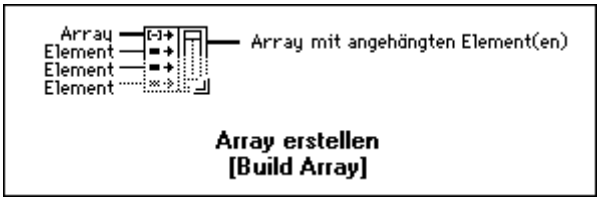


Ende der Übung 5-2.

Array-Funktionen verwenden

G verfügt über viele Funktionen, mit denen Arrays manipuliert werden können. Sie können auf diese Funktionen über **Funktionen»Array** zugreifen. Die folgenden Funktionen stehen dort zur Verfügung: “Array-Element ersetzen”, “1D Array durchsuchen”, “1D Array sortieren”, “1D Array umkehren” und “Multipliziert Array-Elemente”. Weitere Informationen zu Arrays und den verfügbaren Array-Funktionen finden Sie in Kapitel 14, *Bedien- und Anzeigeelemente vom Typ Array und Cluster*, im *Referenzhandbuch zur Programmierung in G* oder über **Online-Referenz»Funktion und VI Referenz**.

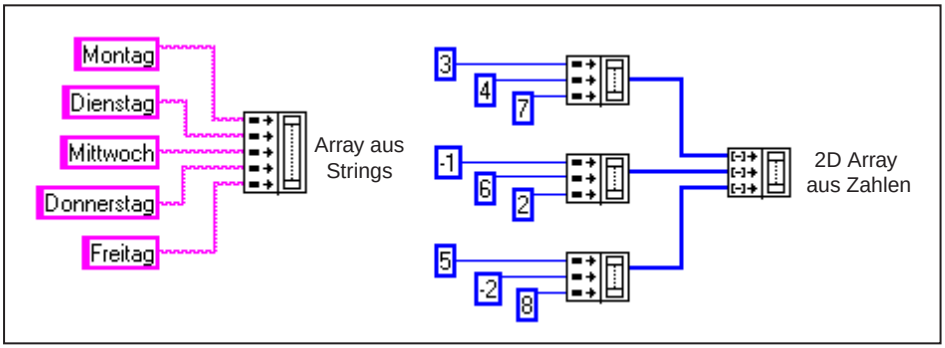
Array erstellen



Array erstellen (**Funktionen»Array**)—Sie können diese Funktion verwenden, um ein Array auf der Grundlage skalarer Werte oder anderer Arrays zu erstellen. Die Funktion “Array erstellen” wird zu Beginn mit einem skalaren Eingang angezeigt.

Sie können beliebig viele Eingänge zur Funktion “Array erstellen” hinzufügen. Jeder einzelne Eingang kann dabei entweder ein skalarer Eingang oder ein Array sein. Wenn Sie weitere Eingänge hinzufügen möchten, können Sie das Popup-Menü auf der linken Seite der Funktion aufrufen und **Elementeingang hinzufügen** oder **Array-Eingang hinzufügen** wählen. Sie können den Knoten zum Erstellen eines Arrays auch mit dem Skaliercursor vergrößern (plazieren Sie das Positionierwerkzeug in der Ecke eines Objekts, um den Cursor in den Skaliercursor zu verwandeln). Sie können Eingänge entfernen, indem Sie den Knoten mit dem Skaliercursor verkleinern oder **Eingang entfernen** wählen.

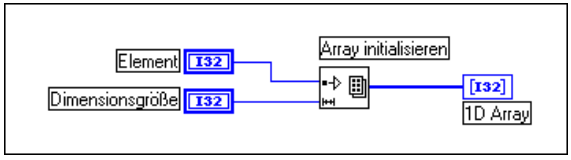
Die folgende Abbildung illustriert zwei Wege zum Erstellen und Initialisieren von Arrays mit Werten von Blockdiagrammkonstanten. Auf der linken Seite werden fünf String-Konstanten in ein 1D Array mit Strings integriert. Auf der rechten Seite werden drei Gruppen mit numerischen Konstanten in drei numerische 1D Arrays integriert. Die drei Arrays werden anschließend in einem numerischen 2D Array zusammengefaßt. Das Ergebnis ist ein 3 x 3 Array mit den folgenden Reihen: 3/4/7, –1/6/2 und 5/–2/8.



Sie können ein Array auch erstellen, indem Sie andere Arrays und skalare Elemente zusammenfassen. Beispiel: Sie haben zwei Arrays und drei skalare Elemente, die Sie in einem neuen Array in der Reihenfolge “Array 1”, “Skalares Element 1”, “Skalares Element 2”, “Array 2” und “Skalares Element 3” zusammenfassen möchten.

Array initialisieren

Verwenden Sie diese Funktion, um ein Array zu erstellen, dessen Elemente alle denselben Wert aufweisen. Die folgende Abbildung illustriert diesen Vorgang anhand eines 1D Arrays.

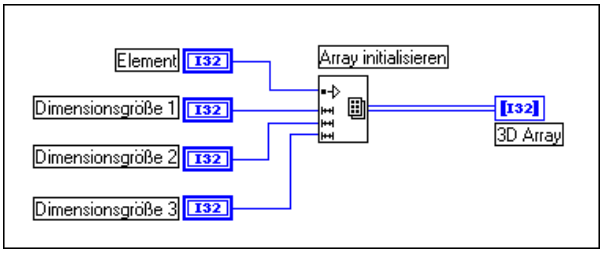


Der Elementeingang bestimmt den Datentyp und den Wert jedes einzelnen Elements. Der Dimensionsgrößeneingang bestimmt die Länge des Arrays. Wenn z.B. `Element` ein Long-Integer-Wert ist, der dem Wert 5 entspricht, und als `Dimensionsgröße` der Wert 100 verwendet wird, ist das Ergebnis ein 1D Array mit 100 Long-Integer-Werten, die alle auf 5 eingestellt sind. Sie können die Eingänge mit den Bedienelementterminals des Frontpanels (wie in der vorherigen Abbildung gezeigt), mit Blockdiagramm-Konstanten oder mit anderen Berechnungen in anderen Teilen des Diagramms verbinden.

Rufen Sie das Popup-Menü auf der unteren linken Seite der Funktion auf, und wählen Sie **Dimension hinzufügen**, um ein Array zu erstellen und zu initialisieren, das mehrere Dimensionen hat. Sie können auch den Array-Initialisierungsknoten mit Hilfe des Skaliercursors vergrößern und

weitere Dimensionsgrößeneingänge hinzufügen, d.h. jeweils einen Eingang für jede weitere Dimension. Durch Verkleinern des Knotens können Sie Dimensionen entfernen. Wählen Sie dazu “Dimension entfernen” im Popup-Menü für die Funktion, oder benutzen Sie den Skaliercursor.

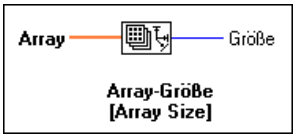
Das folgende Blockdiagramm illustriert die Initialisierung eines 3D Arrays.

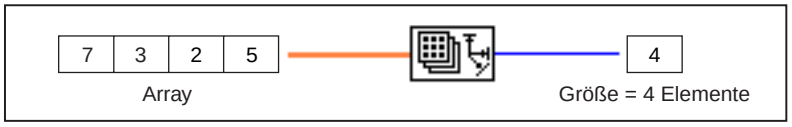


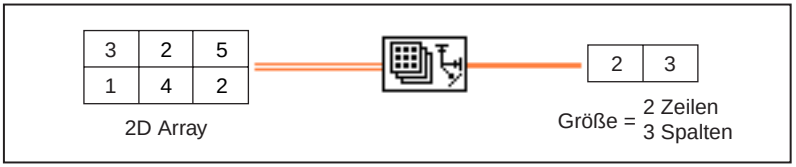
Wenn alle Dimensionsgrößeneingänge den Wert Null aufweisen, wird durch diese Funktion ein leeres Array des angegebenen Typs und in der angegebenen Dimension erstellt.

Array-Größe

Durch die Funktion “Array-Größe” wird die Anzahl der Elemente im Eingabearray ausgegeben.

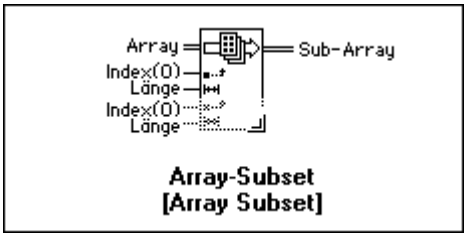




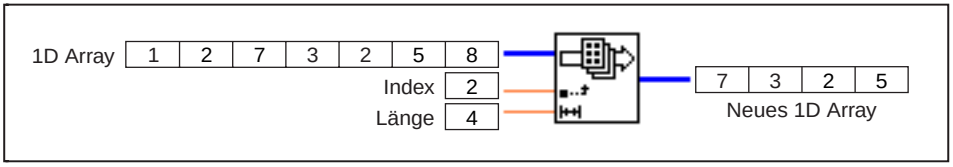


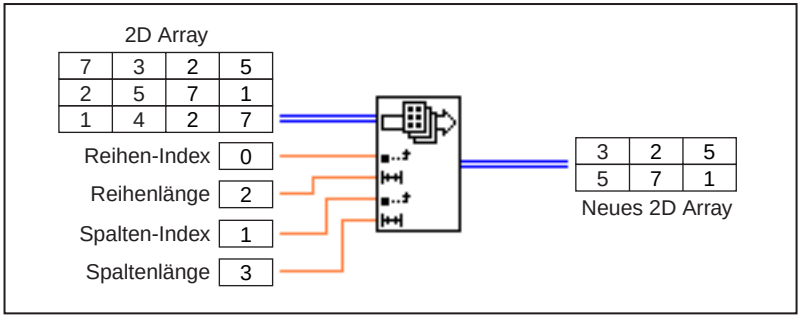
Array-Subset

Mit dieser Funktion können Sie einen Teil eines Arrays oder einer Matrix extrahieren.



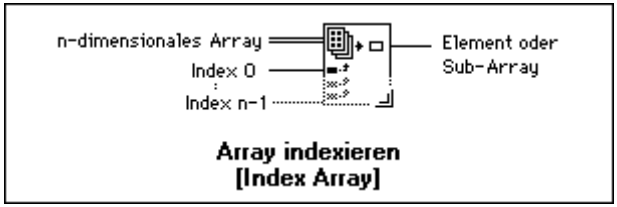
Die Funktion “Array-Subset” gibt einen Teil eines Arrays aus, der mit dem Index beginnt und Längenelemente enthält. Die folgenden Abbildungen zeigen Beispiele für Array-Subsets. Beachten Sie, daß der Array-Index mit 0 beginnt.



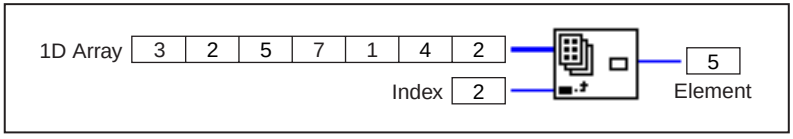


Array indexieren

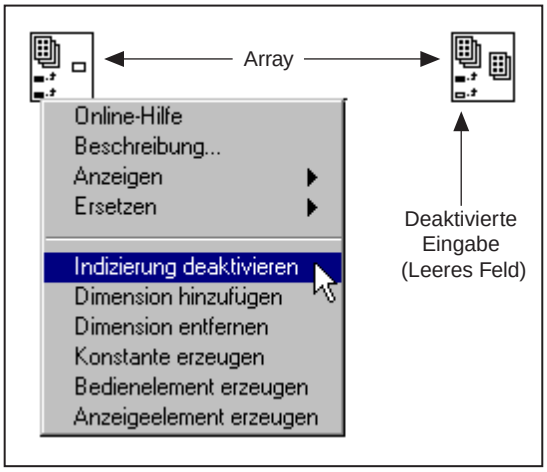
Die Funktion “Array indexieren” greift auf ein Element eines Arrays zu.



Die folgende Abbildung zeigt ein Beispiel für die Funktion “Array indexieren”, mit der auf das dritte Element in einem Array zugegriffen wird. Beachten Sie, daß der Index für das dritte Element 2 ist, weil das erste Element den Index 0 hat.

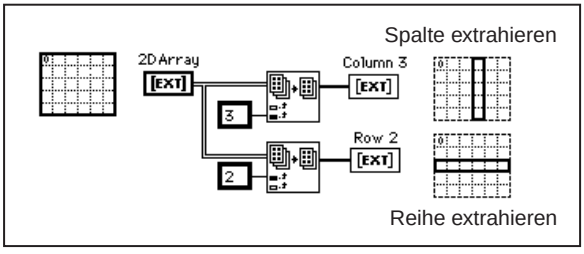


Sie können diese Funktion auch dazu verwenden, eine oder mehrere Dimensionen eines multi-dimensionalen Arrays abzutrennen, um auf diese Weise ein Subarray des Originals zu erstellen. Strecken Sie dazu die Funktion “Array indexieren” in die Länge, damit zwei Index-Eingänge Platz haben, und wählen Sie den Befehl **Indizierung deaktivieren** im Popup-Menü des zweiten Index-Terminals, wie in der folgenden Abbildung gezeigt. Sie haben damit nun den Zugang zu einer speziellen Array-Spalte deaktiviert. Durch Angabe eines Reihen-Indexes erhalten Sie als Ergebnis ein Array, dessen Elemente den Elementen in der angegebenen Reihe des 2D Arrays entsprechen. Sie können die Indizierung auch über das Reihen-Terminal deaktivieren.

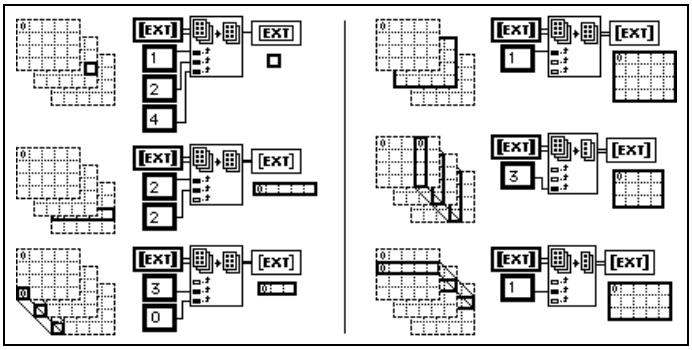


Beachten Sie, daß sich das Symbol für das Index-Terminal aus einem ausgefüllten Feld in ein leeres Feld verwandelt, wenn Sie die Indizierung deaktivieren. Wählen Sie im selben Menü den Befehl **Indizierung aktivieren**, um einen deaktivierten Index wiederherzustellen.

Sie können Subarrays für beliebige Dimensionskombinationen extrahieren. Die folgende Abbildung illustriert, wie eine 1D Reihe oder Spalten-Arrays aus einem 2D Array extrahiert werden.



Sie können ein 2D Array aus einem 3D Array extrahieren, indem Sie zwei Index-Terminals deaktivieren. Durch Deaktivieren eines einzelnen Index-Terminals erhalten Sie ein 1D Array. Die folgende Abbildung zeigt mehrere Möglichkeiten zum Auftrennen eines 3D Arrays.



Für das Auftrennen von Arrays mit Hilfe der Funktion “Array indexieren” gelten die folgenden Regeln:

- Die Dimension des Ausgangsobjekts muß der Anzahl der deaktivierten Index-Terminals entsprechen. Beispiel:
  - Kein Terminal deaktiviert = Skalares Element
  - Ein Terminal deaktiviert = 1D Komponente
  - Zwei Terminals deaktiviert = 2D Komponente
- Die Werte, die mit aktivierten Terminals verbunden werden, müssen die Ausgangselemente identifizieren.

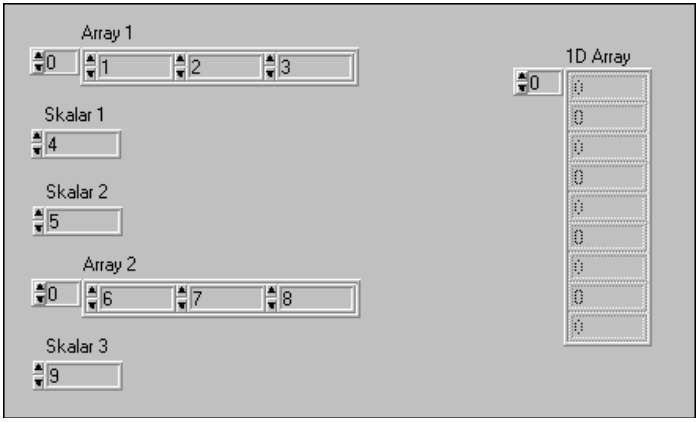
Sie können also das untere linke Beispiel in der obigen Abbildung als einen Befehl interpretieren, durch den ein 1D Array aller Elemente in Spalte 0 und Reihe 3 erzeugt wird. Das obere rechte Beispiel kann entsprechend als Befehl zum Erzeugen eines 2D Arrays von Seite 1 interpretiert werden. Wie in der vorherigen Abbildung zu sehen ist, entspricht das neue 0-Element am ehesten dem Original.

Übung 5-3. Die Funktion “Array erstellen” verwenden

Übungsziel ist das Verwenden der Funktion “Array erstellen”, um Elemente und Arrays in einem größeren Array zusammenzufassen.

Frontpanel

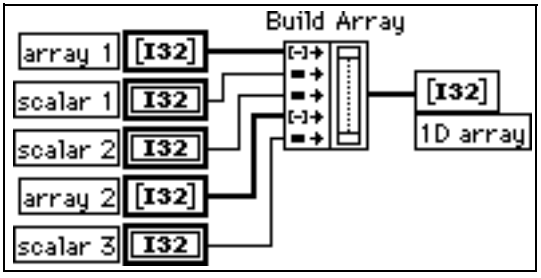
1. Erstellen Sie ein neues Frontpanel entsprechend der folgenden Abbildung.



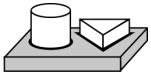
2. Plazieren Sie ein numerisches Eingabefeld mit Hilfe der Palette **Elemente»Numerisch**, und beschriften Sie es mit `Skalar 1`. Ändern Sie seine Repräsentierung in `I32`.
3. Kopieren Sie das Element, und fügen Sie es erneut ein, um zwei weitere numerische Eingabefelder zu erstellen. Beschriften Sie sie mit `Skalar 2` und `Skalar 3`.
4. Erstellen Sie ein Array der numerischen Eingaben, und beschriften Sie es mit `Array 1`. Kopieren Sie es, und fügen Sie es erneut ein. Nennen Sie das neue Array `Array 2`.
5. Erweitern Sie die Arrays, und geben Sie wie in der obigen Abbildung die Werte 1 bis 9 in `Array 1`, `Skalar 1`, `Skalar 2`, `Array 2` und `Skalar 3` ein.
6. Kopieren Sie das Array, und fügen Sie es erneut ein. Ändern Sie es in ein Anzeigeelement um. Beschriften Sie es mit `1D Array`. Erweitern Sie das Anzeigeelement, so daß es neun Werte anzeigt.

Blockdiagramm

7. Plazieren Sie die Funktion “Array erstellen” (**Funktionen»Array**) auf dem Blockdiagramm. Erweiteren Sie die Funktion mit dem Positionierwerkzeug, so daß sie fünf Eingänge hat.
8. Rufen Sie das Popup-Menü für den ersten Eingang im Knoten “Array erstellen” auf, und wählen Sie **In Array ändern**. Gehen Sie auf dieselbe Weise für den vierten Eingang vor.
9. Verbinden Sie die Arrays und skalaren Elemente mit dem Knoten. Das Ausgabearray ist ein 1D Array, das sich aus den Elementen von `Array 1` sowie `Skalar 1`, `Skalar 2` und den Elementen von `Array 2` und `Skalar 3` im Anschluß daran zusammensetzt. Die folgende Abbildung illustriert diese Anordnung.



10. Führen Sie das VI aus. Sie können sehen, daß die Werte in `Skalar 1`, `Skalar 2`, `Skalar 3`, `Array 1` und `Array 2` in einem einzelnen 1D Array erscheinen.
11. Speichern Sie das VI unter dem Namen `Build Array.vi` im Verzeichnis `LabVIEW\Activity`.



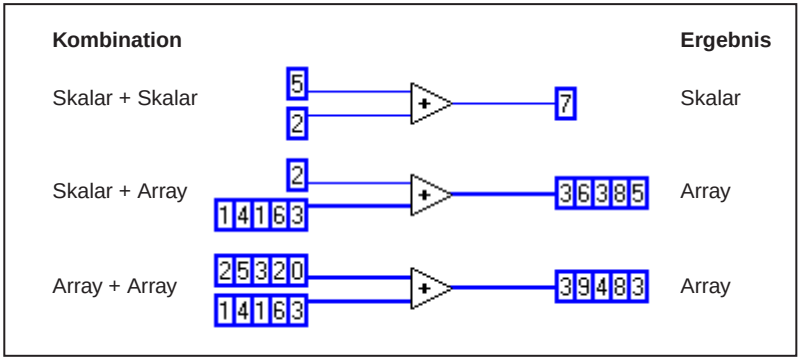
Ende der Übung 5-3.

Effiziente Speicherausnutzung: Datenkopien minimieren

Um Speicherplatz zu sparen, können Sie Arrays in einfacher Genauigkeit anstatt in Double-Präzision verwenden. Sie finden Informationen zum Zuordnen des Speichers im Abschnitt *Überwachung der Speicherauslastung* in Kapitel 28, *Leistung*, im *Referenzhandbuch zur Programmierung in G*.

Was ist Polymorphismus?

Als *Polymorphismus* wird die Fähigkeit einer Funktion bezeichnet, entsprechend des Typs, der Dimension oder der Repräsentierung von Eingangsdaten Anpassungen durchzuführen. Die meisten G-Funktionen sind polymorph. Die folgende Abbildung zeigt z.B. einige der polymorphen Kombinationen für die Funktion “Addieren”.



In der ersten Kombination werden die beiden skalaren Elemente addiert, und als Ergebnis entsteht ein skalares Element. In der zweiten Kombination wird das skalare Element zu jedem Element des Arrays addiert, und es ergibt sich ein Array. Ein Array stellt eine Ansammlung von Daten dar. In der dritten Kombination wird jedes Element eines Arrays zu dem entsprechenden Element des anderen Arrays addiert. Sie können auch andere Kombinationen benutzen, wie z.B. Cluster mit numerischen Werten oder Arrays, die aus Clustern bestehen.

Diese Prinzipien können auf andere Funktionen und Datentypen in G angewendet werden. G-Funktionen sind in unterschiedlichem Umfang polymorph. Manche Funktionen akzeptieren unter Umständen nur numerische und Boolesche Eingaben; andere Funktionen akzeptieren dagegen eventuell eine Kombination aller anderen Datentypen. Weitere Informationen über Polymorphismus erhalten Sie über **Online-Referenz» Funktion und VI Referenz**.

Cluster

Ein Cluster ist ein Datentyp, der Datenelemente unterschiedlichen Typs enthalten kann. Das Cluster im Blockdiagramm, das Sie in Übung 5-4 anfertigen werden, stellt verwandte Datenelemente von mehreren Stellen auf dem Blockdiagramm in Gruppen zusammen. Dadurch werden verwirrende Verbindungen vermieden. Wenn Sie Cluster verwenden, benötigen die SubVIs eine geringere Anzahl an Verbindungsterminals. Ein Cluster entspricht einem “Record” in Pascal oder einem “Struct” in C. Sie können sich ein Cluster als ein Bündel aus Verbindungen vorstellen, ähnlich einem Telefonkabel. Jede Verbindung in dem Kabel repräsentiert ein anderes Element des Clusters. Zu den Komponenten des Clusters gehören der Anfangswert für *X* (0), der Wert für Delta *X* (1) und das Y Array (Signalverlaufsdaten, die von den numerischen Konstanten auf dem Blockdiagramm geliefert werden). Verwenden Sie die Bündelfunktion, um ein Cluster in G zusammenzustellen. Weitere Information zu Clustern finden Sie in Kapitel 14, *Bedien- und Anzeigeelemente vom Typ Array und Cluster*, im *Referenzhandbuch zur Programmierung in G*.

Graphen

Ein *Graph* ist eine zweidimensionale Anzeige für einen oder mehrere Arrays, die als Plots bezeichnet werden. Die Palette **Elemente»Graph** enthält drei verschiedene Typen von Graphen:

- XY-Graph
- Kurvengraph
- Intensitätsgraph

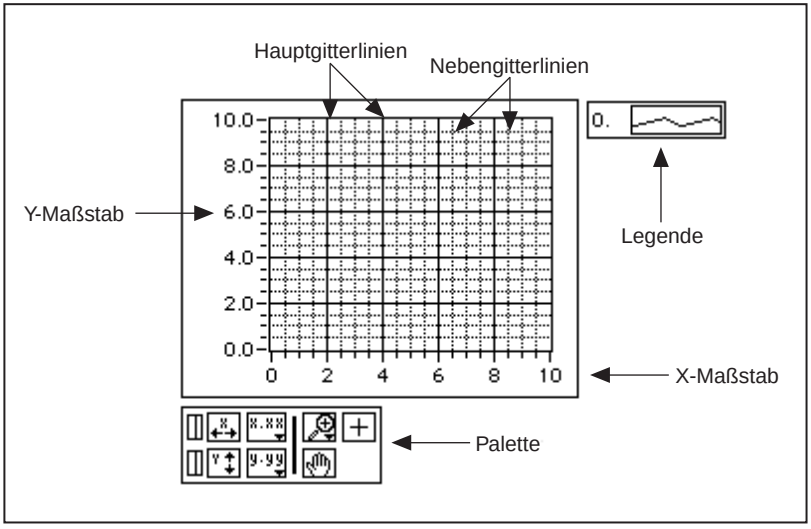
Der Unterschied zwischen einem Graphen und einem Diagramm besteht darin, daß ein Graph die Daten als Block zeichnet, während ein Diagramm die Daten Punkt für Punkt bzw. Array für Array zeichnet.

Sie finden Beispiele für Graphen-VIs in `Examples\General\Graphs`.

Graphen anpassen

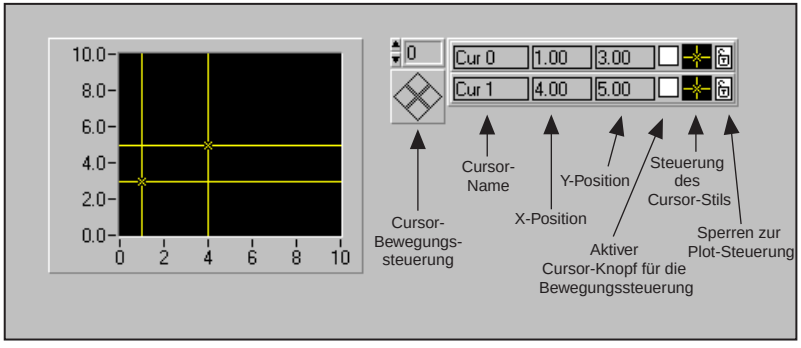
Sowohl Kurvengraphen als auch XY-Graphen enthalten eine Anzahl optionaler Bestandteile, die Sie mit Hilfe des Untermenüs **Anzeigen** im Popup-Menü für den Graphen ein- bzw. ausblenden können. Zu diesen Optionen gehören eine Legende, über die Sie die Farbe und den Stil für einen einzelnen Plot definieren können, eine Palette, über die Sie Skalier-

und Formatoptionen ändern können, während das VI ausgeführt wird, sowie eine Cursor-Anzeige. Die folgende Abbildung eines Graphen zeigt alle optionalen Komponenten mit Ausnahme der Cursor-Anzeige.



Cursor in einem Graphen

In G können Sie Cursor und eine Cursor-Anzeige auf allen Graphen plazieren, und Sie können den Cursor auf dem Plot beschriften. Ein Cursor kann so eingestellt werden, daß er auf dem Plot fixiert ist, und mehrere Cursor können gleichzeitig verschoben werden. Die Anzahl der möglichen Cursor für einen Graphen ist unbeschränkt. Die folgende Abbildung zeigt einen Kurvengraphen mit der Cursor-Anzeige.



Detaillierte Informationen zum Anpassen von Graphen finden Sie in Kapitel 15, *Bedien- und Anzeigeelemente vom Typ Graph und Diagramm*, im *Referenzhandbuch zur Programmierung in G*.

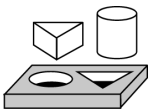
Sehen Sie sich das ZoomGraph-VI in `Examples\General\Graphs\zoom.llb` als Beispiel für ein VI an, das Cursor-Werte liest und mit den Cursorn einen Graphen programmatisch vergrößert und verkleinert.

Achsen eines Graphen

Sie können die Skalen eines Graphen so formatieren, daß entweder die absolute oder eine relative Zeit angegeben wird. Verwenden Sie das absolute Zeitformat, um die Zeit, das Datum oder beides auf der Skala anzuzeigen. Wenn Sie vermeiden möchten, daß G ein Datum annimmt, sollten Sie das relative Zeitformat verwenden. Rufen Sie zur Auswahl des absoluten bzw. relativen Zeitformats das Popup-Menü für das Diagramm auf, und wählen Sie die Skala aus, die Sie ändern möchten. Wählen Sie die Option **Formatieren…**. Daraufhin wird das Dialogfeld **Formatieren** aktiviert, in dem Sie verschiedene Attribute für das Diagramm festlegen können.

Datenerfassungs-Arrays

Die Daten, die von einer DAQ-Einsteckkarte mit Hilfe der Datenerfassungs-VIs ausgegeben werden, können als Einzelwert, als 1D Array oder als 2D Array vorliegen. Im Verzeichnis `Examples\General\Graphs`, das VIs für das Ausführen unterschiedlicher Funktionen mit Arrays und Graphen enthält, finden Sie eine Reihe von Beispielen für Graphen.

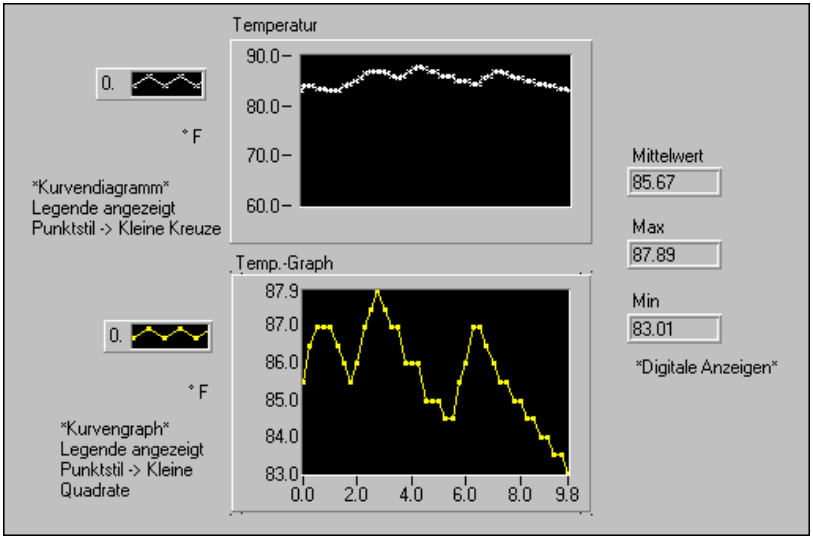


Übung 5-4. Graphen- und Analyse-VIs verwenden

Übungsziel ist das Erstellen eines VIs, das die Temperatur mißt und die Werte in Echtzeit anzeigt. Es zeigt außerdem die durchschnittliche, höchste und tiefste Temperatur an.

Frontpanel

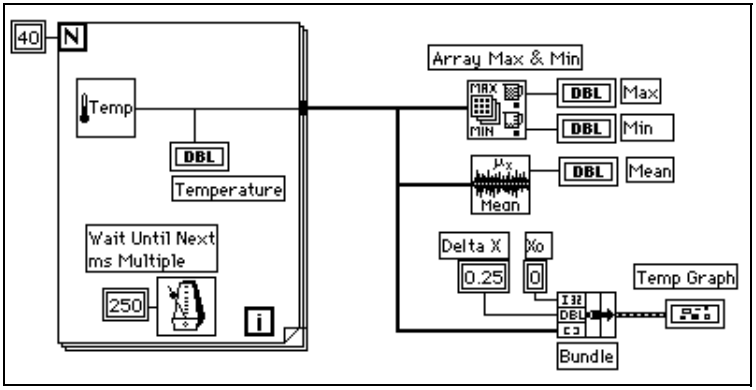
1. Erstellen Sie ein neues Frontpanel entsprechend der folgenden Abbildung. Sie können den Punktstil des Kurvendiagramms und des Kurvengraphen ändern, indem Sie das Popup-Menü für die jeweilige Legende aufrufen. Skalieren Sie die Diagramme in der dargestellten Form.



Das Temperatur-Kurvendiagramm zeigt die erfaßte Temperatur an. Nach der Erfassung zeichnet das VI die Daten im Bereich `Temp.-Graph`. Die digitalen Anzeigen `Mittelwert`, `Max` und `Min` zeigen die durchschnittliche, höchste und tiefste Temperatur an.

Blockdiagramm

2. Erstellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm:



Digital-Thermometer-VI (**Funktionen»VI auswählen** im Verzeichnis `LabVIEW\Activity`)—Gibt einen Temperaturmeßwert aus.

Wartet bis zum nächsten Vielfachen von ms (**Funktionen» Zeit & Dialog**)—In dieser Übung wird durch diese Funktion sichergestellt, daß die For-Schleife in Abständen von 0,25 Sekunden (250 Millisekunden) ausgeführt wird.

Numerische Konstante (**Funktionen»Numerisch**)— Sie können auch das Popup-Menü für die Funktion “Wartet bis zum nächsten Vielfachen von ms” aufrufen und die Option **Konstante erzeugen** wählen, um die numerische Konstante automatisch zu erstellen und zu verbinden.

Max & Min Array (**Funktionen»Array**)—In dieser Übung wird durch diese Funktion die Höchst- und Tiefsttemperatur ausgegeben, die während der Erfassung gemessen wurde.

Mittelwert-VI (**Funktionen»Analysis»Wahrscheinlichkeit und Statistik oder Funktionen»Basis-Analyse** für Benutzer des LabVIEW-Basispakets) — Gibt den Durchschnitt der Temperaturmeßwerte aus.

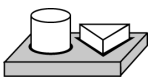
Elemente bündeln (**Funktionen»Cluster**)—Faßt die Plot-Komponenten in einem Cluster zusammen. Zu den Komponenten gehören der Anfangswert für *X* (0), der Wert für Delta *X* (0,25) und das Y Array (Temperaturdaten). Ziehen Sie mit dem Positionierwerkzeug an einer Ecke der Funktion, um die Funktion in der Größe zu verändern.

Die For-Schleife wird 40mal ausgeführt. Die Funktion “Wartet bis zum nächsten Vielfachen von ms” bewirkt dabei, daß jeweils nach 250 Millisekunden eine Iteration durchgeführt wird. Das VI speichert die Temperaturmessungen in einem Array, das am Rand der For-Schleife erstellt wird (Auto-Indizierung). Sobald die Ausführung der For-Schleife abgeschlossen ist, wird das Array an die SubVIs und den `Temp.-Graphen` weitergeleitet.

Die Funktion “Max & Min Array” ermittelt die Höchst- und Tiefsttemperatur, und das Mittelwert-VI gibt den Durchschnitt der Temperaturmeßwerte aus.

Das fertiggestellte VI bündelt das Daten-Array mit dem Wert 0 als Anfangswert für X und 0,25 als Wert für Delta-X. Das VI benötigt einen Delta X-Wert von 0,25, um die Punkte für das Temperatur-Array alle 0,25 Sekunden auf dem Kurvengraphen zu zeichnen.

3. Kehren Sie zum Frontpanel zurück, und führen Sie das VI aus.
4. Speichern Sie das VI unter dem Namen `Temperature Analysis.vi` im Verzeichnis `LabVIEW\Activity`.



Ende der Übung 5-4.

Intensitäts-Plots

LabVIEW bietet zwei Methoden zum Anzeigen dreidimensionaler Daten: das Intensitätsdiagramm und den Intensitätsgraphen. Beide Intensitäts-Plots akzeptieren zweidimensionale Zahlen-Arrays, in denen jede Zahl einer Farbe zugeordnet ist. Sie können die Farbskalierung interaktiv durch eine optionale Farbrampenskala oder programmatisch durch einen Attributknoten für das Diagramm definieren. Beispiele für die Verwendung von Intensitätsdiagrammen und -graphen finden Sie in `intgraph.llb` im Verzeichnis `Examples\General\Graphs`.

6

Strings und Datei I/O

In diesem Kapitel werden die Bedien- und Anzeigeelemente für Strings und die Eingabe- und Ausgabevorgänge für Dateien erklärt. Außerdem enthält dieses Kapitel Übungen, durch die die folgenden Aufgaben illustriert werden:

- Bedien- und Anzeigeelemente für Strings erstellen
- String-Funktionen verwenden
- Datei-Eingabe- und Ausgabevorgänge durchführen
- Datendateien in Tabellenformat speichern
- Daten in Textdateien schreiben und von Textdateien lesen

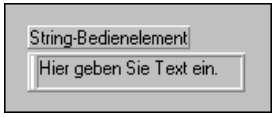
Strings

Ein String ist eine Ansammlung von ASCII-Zeichen. Bei der Instrumentensteuerung können Sie numerische Daten als Zeichen-Strings weiterleiten und diese Strings dann in Zahlen konvertieren. Auch beim Speichern numerischer Daten in einem Speichermedium können Strings eingesetzt werden. Wenn Sie Zahlen in einer ASCII-Datei speichern möchten, müssen Sie die Zahlen zuerst in Strings umwandeln, bevor Sie die Zahlen in eine Datei auf dem Speichermedium schreiben können.

Beispiele für Strings finden Sie in `Examples\General\strings.llb`.

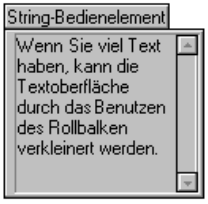
String-Bedien- und Anzeigeelemente erstellen

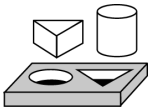
Sie finden das links abgebildete String-Bedien- und Anzeigeelement in **Elemente»String & Tabelle**. Mit dem Bedienwerkzeug oder dem Beschriftungswerkzeug können Sie Text in ein String-Bedienelement eingeben oder vorhandenen Text bearbeiten. Ziehen Sie mit dem Positionierwerkzeug an einer Ecke des String-Bedien- oder Anzeigeelements, wenn Sie es vergrößern möchten.



Strings und Datei I/O

Wenn Sie den Bereich, den ein String-Bedien- oder Anzeigeelement auf dem Frontpanel beansprucht, verkleinern möchten, sollten Sie **Anzeigen» Rollbalken** wählen. Falls diese Option abgeblendet ist, müssen Sie das Fenster in vertikaler Richtung vergrößern, damit die Option verfügbar wird.



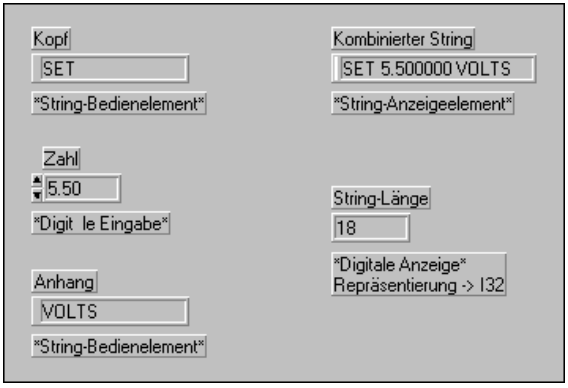


Übung 6-1. Strings verknüpfen

LabVIEW bietet viele Funktionen, mit denen Strings manipuliert werden können. In dieser Übung besteht das Übungsziel darin, mit einigen dieser String-Funktionen eine Zahl in einen String umzuwandeln und diesen String mit anderen Strings so zu verknüpfen, daß ein einzelner Ausgabe-String entsteht.

1. Öffnen Sie ein neues Frontpanel, und fügen Sie die in der folgenden Abbildung gezeigten Objekte hinzu. Achten Sie darauf, daß Sie die Bedien- und Anzeigeelemente so verändern, daß sie mit der Abbildung übereinstimmen.

Frontpanel

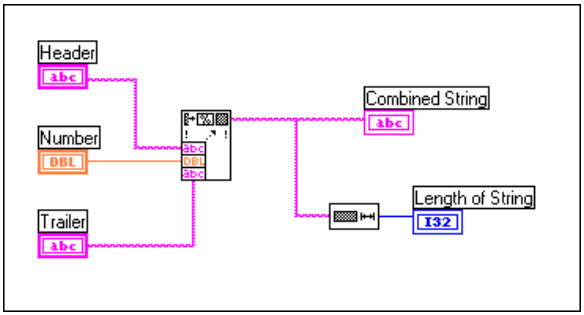


Die beiden String-Bedienelemente und die digitale Eingabe können zu einem einzelnen Ausgabe-String kombiniert und im String-Anzeigeelement angezeigt werden. Die digitale Eingabe zeigt die String-Länge an.

Das Ausgabeformat des kombinierten Strings ähnelt in dieser Übung dem Format von Command-Strings, die zur Kommunikation mit GPIB-(IEEE 488) und seriellen (RS-232 oder RS-422) Geräten verwendet werden. In Teil II, *I/O-Schnittstellen*, dieses Handbuchs erfahren Sie mehr über Strings, die für Gerätebefehle verwendet werden.

Blockdiagramm

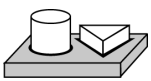
2. Stellen Sie ein Blockdiagramm entsprechend der folgenden Abbildung zusammen.



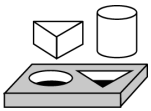
In String formatieren (**Funktionen»String**)—Verknüpft und formatiert Zahlen und Strings zu einem einzelnen Ausgabe-String. Verwenden Sie den Skaliercursor auf dem Symbol, um drei Argumenten-Eingänge hinzuzufügen.

String-Länge (**Funktionen»String**)—Gibt die Anzahl der Zeichen in dem verknüpften String aus.

3. Führen Sie das VI aus. Beachten Sie, daß die Funktion “In String formatieren” die beiden String-Bedienelemente und die digitale Eingabe zu einem einzelnen Ausgabe-String verknüpft.
4. Speichern Sie das VI unter dem Namen `Build String.vi`. Sie benötigen dieses VI für die nächste Übung.



Ende der Übung 6-1.



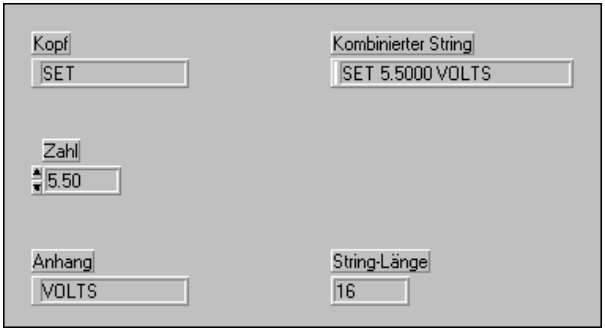
Übung 6-2. Format-Strings verwenden

Übungsziel ist das Anfertigen eines Strings in dem von Ihnen vorgegebenen Format.

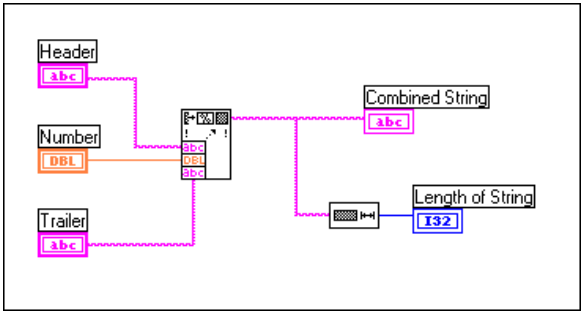
Sie verwenden in dieser Übung das VI, das Sie in Übung 6-1 zusammengestellt haben, um einen Format-String zu erstellen. Mit Format-Strings können Sie das Format von Argumenten festlegen, einschließlich der Feldbreite, der Basis (hexadezimal, oktal usw.) und eines Textes, der die Argumente voneinander trennt.

Frontpanel

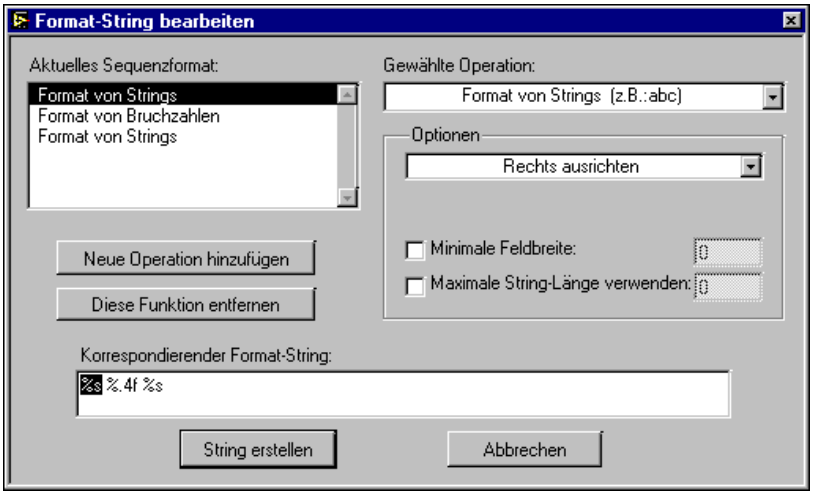
1. Öffnen Sie das VI zum Erstellen eines Strings, das Sie in Übung 6-1 zusammengestellt haben.



Blockdiagramm



2. Rufen Sie das Popup-Menü für das VI “In String formatieren” auf, und wählen Sie **Format-String bearbeiten**. Daraufhin wird das folgende Dialogfeld angezeigt:

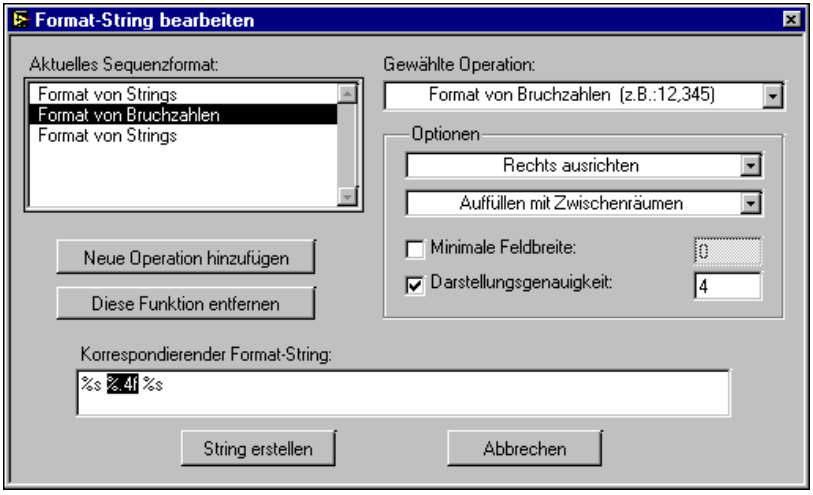


☞ **Hinweis** ***Sie können auch auf den Knoten doppelklicken, um auf das Dialogfeld “Format-String bearbeiten” zuzugreifen.***

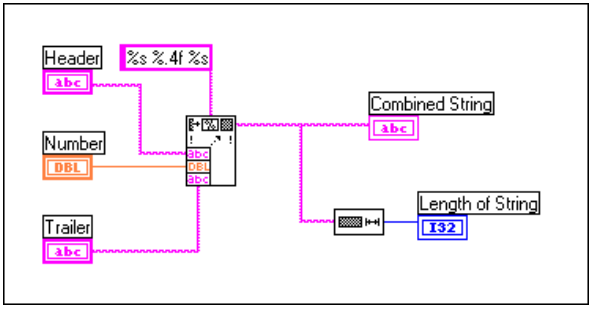
Beachten Sie, daß im Feld “Aktuelles Sequenzformat” die Argumententypen in der Reihenfolge aufgeführt werden, in der Sie sie verbunden haben.

3. Stellen Sie 4 als Genauigkeit für den numerischen Wert ein.
   a. Markieren Sie `Format von Bruchzahlen` im Listenfeld “Aktuelles Sequenzformat”.
   b. Klicken Sie auf das Kontrollkästchen “Angegebene Darstellungsgenauigkeit verwenden”.
   c. Markieren Sie den numerischen Wert neben dem Kontrollkästchen “Spezifizierte Genauigkeit verwenden”, geben Sie 4 ein, und drücken Sie auf `<Eingabe>` (**Windows**); `<Return>` (**Macintosh**); `<Return>` (**Sun**) oder `<Enter>` (**HP-UX**).

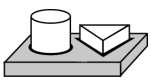
Die folgende Abbildung zeigt die Optionen, die zur Einstellung der Genauigkeit von Zahlen gewählt werden müssen.



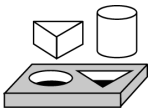
4. Drücken Sie auf die Taste **String erstellen**. Durch Drücken dieser Taste werden automatisch die korrekten Informationen für das String-Format eingegeben und der Format-String wie in der folgenden Abbildung mit der Funktion verbunden.



5. Kehren Sie zum Frontpanel zurück, und geben Sie Text in den beiden String-Bedienelementen und eine Zahl in der digitalen Eingabe ein. Führen Sie das VI aus.
6. Speichern Sie das VI, und schließen Sie es. Benennen Sie es `Format String.vi`.



Ende der Übung 6-2.

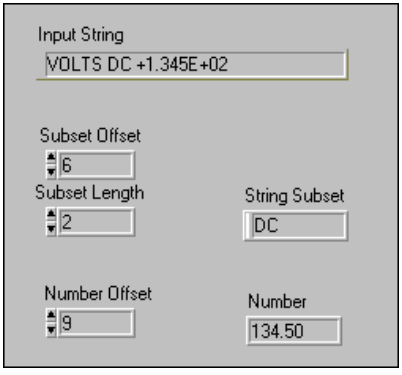


Übung 6-3. String-Subsets und Zahlen-Extraktion

Übungsziel ist das Umwandeln eines Subsets eines Strings, das die String-Repräsentierung einer Zahl enthält, in einen numerischen Wert.

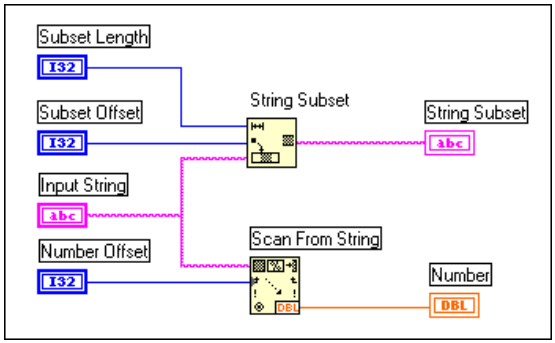
Frontpanel

1. Öffnen Sie das VI `Parse String.vi` im Verzeichnis `Examples\General\strings.llb`. Führen Sie das VI mit den Standardeingaben aus. Beachten Sie, daß das String-Subset *DC* als Eingangs-String gewählt wird. Außerdem können Sie erkennen, daß der numerische Teil des Strings ausgegliedert und in eine Zahl umgewandelt wurde. Sie können unterschiedliche Steuerwerte ausprobieren (denken Sie daran, daß die Indizierung von Strings wie von Arrays bei Null beginnt), oder Sie können das Blockdiagramm anzeigen, um festzustellen, wie die Komponenten aus dem Eingangs-String auszugliedern sind.



Blockdiagramm

2. Öffnen Sie das Blockdiagramm für das VI zum Ausgliedern von Strings, das in der folgenden Abbildung dargestellt ist.



LabVIEW verwendet die Funktionen “String-Subset” und “Aus String suchen”, um den Eingangs-String auszugliedern.

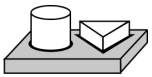
String-Subset (**Funktionen»String**)—Gibt den Substring aus, der am **Offset** beginnt und die als **Länge** angegebene Anzahl an Zeichen enthält. Der erste Zeichen-Offset ist Null.

In vielen Fällen müssen Sie Strings in Zahlen umwandeln, so z.B. bei der Umwandlung eines Daten-Strings, der von einem Instrument empfangen wurde, in die Datenwerte.

Aus String suchen (**Funktionen»String**)—Tastet einen String ab und wandelt gültige numerische Zeichen (0 bis 9, +, –, e, E und Punkt) in Zahlen um. Wenn Sie eine Verbindung mit einem Format-String herstellen, führt die Funktion “Aus String suchen” die Umwandlungen in Übereinstimmung mit dem Format durch. Wenn kein Format-String verbunden ist, führt die Funktion Standardumwandlungen für jedes Standard-Eingangsterminal in der Funktion durch. Dabei beginnt die Funktion mit dem Abtasten beim **String** am **Offset**. Der erste Zeichen-Offset ist Null.

Die Funktion “Aus String suchen” ist besonders nützlich, wenn Ihnen die Kopfzeilenlänge bekannt ist (`VOLT/Gleichstrom` im vorliegenden Beispiel) oder wenn der String nur gültige numerische Zeichen enthält.

3. Schließen Sie das VI, indem Sie **Datei»Schließen** wählen. Speichern Sie das VI nicht.



Ende der Übung 6-3.

Datei I/O

Mit den Datei I/O-Funktionen in G (**Funktionen»Datei I/O**) verfügen Sie über leistungsstarke und flexible Mittel für die Arbeit mit Dateien. Mit den Datei I/O-Funktionen von LabVIEW können Sie nicht nur Daten lesen und schreiben, sondern auch Dateien und Verzeichnisse verschieben und umbenennen, tabellenartige Dateien mit lesbarem ASCII-Text erstellen und zur Verkürzung der Bearbeitungszeit und zum kompakteren Arbeiten Daten in binärer Form schreiben.

Sie können Daten in drei verschiedenen Formaten speichern oder aus Dateien abrufen.

- ASCII Byte-Stream — Sie sollten Daten, auf die Sie von einem anderen Softwarepaket, z.B. von einem Textverarbeitungs- oder Tabellenkalkulationsprogramm, aus zugreifen möchten, in ASCII-Format speichern. Um die Daten auf diese Weise zu speichern, müssen Sie alle Daten in ASCII-Strings umwandeln.
- Datenprotokolldateien — Diese Dateien liegen in einem binären Format vor, auf das nur G zugreifen kann. Datenprotokolldateien sind mit Datenbankdateien vergleichbar, da mehrere verschiedene Datentypen in einem (Protokoll-)Datensatz für eine Datei gespeichert werden können.
- Binär-Byte-Stream — Diese Dateien bieten die kompakteste und schnellste Methode zum Speichern von Daten. Sie müssen die Daten in binäres String-Format umwandeln, und Sie müssen zum Speichern und Abrufen der Daten in diesen Dateien genau wissen, welche Datentypen Sie verwenden.

In diesem Abschnitt werden ASCII Byte-Stream-Dateien besprochen, da es sich dabei um das am häufigsten verwendete Datendateiformat handelt. Sie finden Beispiele für Datei I/O in `Examples\File`.

Datei I/O-Funktionen

Die meisten Datei I/O-Vorgänge bestehen aus drei Grundschritten: dem Öffnen einer vorhandenen Datei bzw. dem Erstellen einer neuen Datei; dem Schreiben in die Datei bzw. dem Lesen aus der Datei und dem Schließen der Datei. Aus diesem Grund enthält LabVIEW zahlreiche Utility-VIs in **Funktionen»Datei I/O**. In diesem Abschnitt werden die neun übergeordneten Utility-VIs beschrieben. Diese Utility-Funktionen beruhen auf darunterliegenden VIs der mittleren Ebene, die für die Fehlerüberprüfung und -bearbeitung im Rahmen der Datei I/O-Funktionen sorgen.

Das VI “Zeichen in Datei schreiben” schreibt einen Zeichen-String in eine neue Byte-Stream-Datei oder hängt den String an eine vorhandene Datei an. Dieses VI öffnet oder erstellt die Datei, schreibt die Daten und schließt anschließend die Datei wieder.

Das VI “Zeichen aus Datei lesen” liest ab einem festgelegten Zeichen-Offset eine festgelegte Anzahl von Zeichen aus einer Byte-Stream-Datei. Dieses VI öffnet zuerst die Datei und schließt sie nach Ende des Vorgangs wieder.

Das VI “Zeilen aus Datei lesen” liest ab einem festgelegten Zeichen-Offset eine festgelegte Anzahl von Zeilen aus einer Byte-Stream-Datei. Dieses VI öffnet zuerst die Datei und schließt sie nach Ende des Vorgangs wieder.

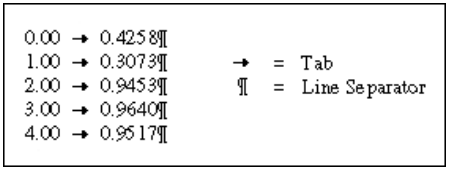
Das VI “In Spreadsheet-Datei schreiben” wandelt ein aus Single-Zahlen bestehendes 1D oder 2D Array in einen Text-String um und schreibt den String in eine neue Byte-Stream-Datei oder hängt ihn an eine vorhandene Datei an. Sie haben dabei die Möglichkeit, die Daten umzustellen. Dieses VI öffnet oder erstellt zuerst die Datei und schließt sie nach Ende des Vorgangs wieder. Mit diesem VI können Sie Textdateien erstellen, die mit den meisten Tabellenkalkulationsprogrammen gelesen werden können.

Das VI “Aus Spreadsheet-Datei lesen” liest ab einem festgelegten Zeichen-Offset eine festgelegte Anzahl von Zeilen oder Reihen aus einer numerischen Textdatei und wandelt die Daten in ein 2D Array mit Zahlen in einfacher Genauigkeit um. Sie haben dabei die Möglichkeit, die Daten umzustellen. Dieses VI öffnet zuerst die Datei und schließt sie nach Ende des Vorgangs wieder. Sie können mit diesem VI Spreadsheet-Dateien lesen, die in Textformat gespeichert wurden.

Wählen Sie **Funktion»Datei I/O»Binärdatei-VIs** oder **Funktion» Datei I/O»Fortgeschrittene Dateifunktionen**, wenn Sie weitere Datei I/O-Funktionen sehen möchten.

In Spreadsheet-Datei schreiben

Eine häufig beim Speichern von Daten in eine Datei verwendete Methode besteht darin, die Textdatei so zu formatieren, daß sie in einer Tabelle (Spreadsheet) geöffnet werden kann. In den meisten Tabellen werden wie in der folgenden Abbildung Spalten durch Tabulatoren und Reihen durch EOL-Zeichen (End of line = Ende der Zeile) getrennt.



Beim Öffnen der Datei mit einem Tabellenkalkulationsprogramm entsteht die folgende Tabelle.

	A	B	C
1	0	0.4258	
2	1	0.373	
3	2	0.9453	
4	3	0.964	
5	4	0.9517	
6			

Übung 6-4. In Spreadsheet-Datei schreiben

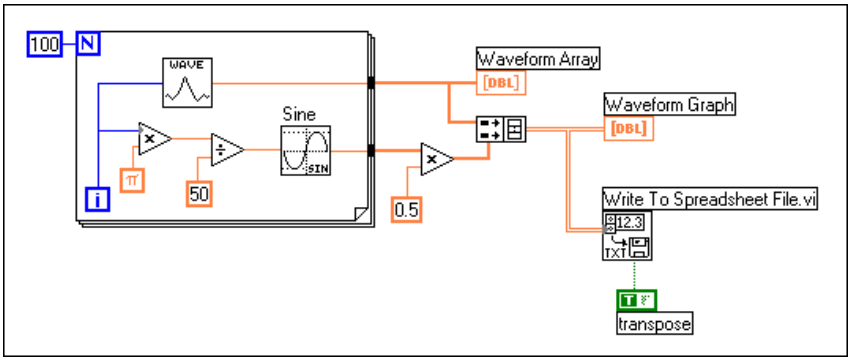
Übungsziel ist das Verändern eines vorhandenen VIs, so daß eine Datei I/O-Funktion das Speichern von Daten in einer neuen Datei in ASCII-Format ermöglicht. Auf diese Datei können Sie anschließend von einem Tabellenkalkulationsprogramm aus zugreifen.

Frontpanel

1. Öffnen Sie das VI `Graph Waveform Arrays.vi`, das Sie in Übung 5-1 erstellt haben. Sie werden sich daran erinnern, daß dieses VI zwei Daten-Arrays erzeugt und sie in einen Graphen überträgt. Sie verändern dieses VI nun, so daß es die beiden Arrays in eine Datei schreibt, in der jede Spalte ein Daten-Array enthält.

Blockdiagramm

2. Öffnen Sie das Blockdiagramm des VIs `Graph Waveform Arrays.vi`, und modifizieren Sie das VI, indem Sie die Blockdiagrammfunktionen hinzufügen, die in der unteren rechten Ecke der folgenden Abbildung ergänzt wurden.



Das VI “In Spreadsheet-Datei schreiben” (**Funktionen»Datei I/O**) wandelt das 2D Array in einen Spreadsheet-String um und schreibt ihn in eine Datei. Falls Sie noch keinen Pfadnamen angegeben haben, wird ein Dialogfeld eingeblendet, in dem Sie zur Eingabe eines Dateinamens aufgefordert werden. Das VI “In Spreadsheet-Datei schreiben” schreibt entweder ein 1D oder ein 2D Array in eine Datei. Da Sie in diesem Beispiel ein 2D Array mit Daten verwenden, müssen Sie keine Verbindung zum 1D-Eingang herstellen. Mit diesem VI können Sie einen Tabellen-Begrenzer oder einen String mit Begrenzern, wie z.B. Tabulatoren oder Kommas in den Daten, verwenden.

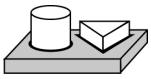
Durch die Funktion “Boolesche Konstante” (**Funktionen»Boolesch**) wird festgelegt, ob G das 2D Array vor dem Schreiben in eine Datei umstellt. Klicken Sie mit dem Bedienwerkzeug auf die Konstante, um den Wert in WAHR zu ändern. Im vorliegenden Fall sollten die Daten umgestellt werden, da die Daten-Arrays reihenspezifisch organisiert sind (jede Reihe des zweidimensionalen Arrays stellt ein Daten-Array dar). Da jede Spalte der Spreadsheet-Datei ein Daten-Array enthält, muß das 2D Array zuerst umgestellt werden.

3. Kehren Sie zum Frontpanel zurück, und führen Sie das VI aus. Nachdem die Daten-Arrays erzeugt wurden, werden Sie in einem Dialogfeld aufgefordert, den Dateinamen für die neu zu erstellende Datei einzugeben. Geben Sie einen Dateinamen ein, und klicken Sie auf **OK**.

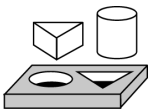
Vorsicht ***Versuchen Sie nicht, Daten in VI-Bibliotheken wie z.B.*** `mywork.llb` ***zu speichern. Unter Umständen überschreiben Sie dadurch die vorhandene Bibliothek, und Sie verlieren die bereits eingegebenen Daten.***

4. Speichern Sie das VI unter dem Namen `Waveform Arrays to File.vi`, und schließen Sie es.
5. Sie können die gerade erstellte Datei nun mit einem Tabellenkalkulationsprogramm oder einem Text-Editor öffnen und einsehen. Es sollten zwei Spalten mit jeweils 100 Elementen vorhanden sein.

In diesem Beispiel wurden die Daten erst umgewandelt oder in eine Datei geschrieben, nachdem alle Daten-Arrays vorhanden waren. Wenn Sie große Datenpuffer abrufen oder die Datenwerte in ein Speichermedium schreiben möchten, während sie erzeugt werden, müssen Sie ein anderes Datei I/O-VI verwenden.



Ende der Übung 6-4.

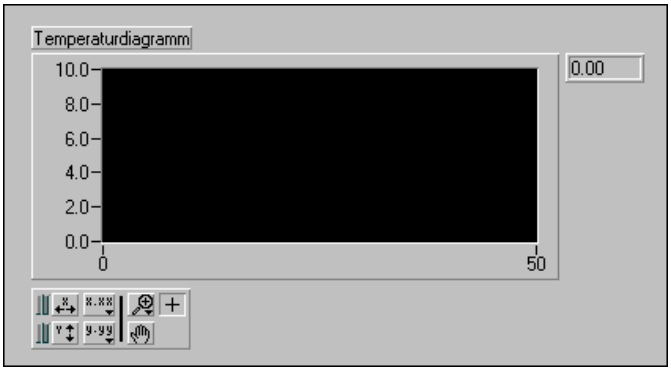


Übung 6-5. Daten an eine Datei anhängen

Übungsziel ist das Erstellen eines VIs zum Anhängen von Temperaturdaten an eine Datei in ASCII-Format. In diesem VI wird eine For-Schleife verwendet, um Temperaturwerte zu erzeugen und in einer Datei zu speichern. Bei jeder Iteration werden die Daten in einen String umgewandelt, ein Komma als Begrenzer eingefügt und der String an eine Datei angehängt.

Frontpanel

1. Öffnen Sie ein neues Frontpanel, und setzen Sie die in der folgenden Abbildung gezeigten Objekte ein.

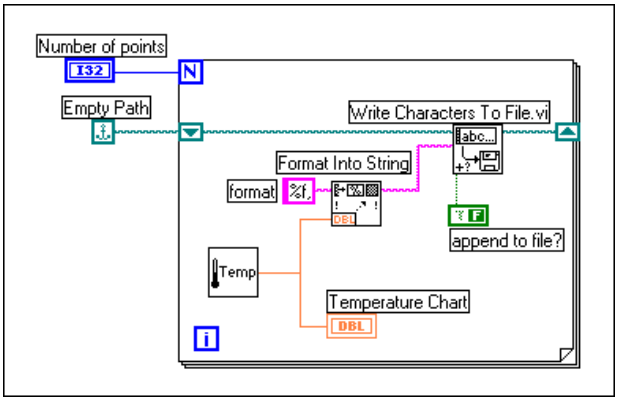


Das Frontpanel weist ein numerisches Eingabefeld und ein Kurvendiagramm auf. Wählen Sie **Anzeigen»Digitale Anzeige**. Durch das Bedienfeld `Anzahl der Punkte` wird festgelegt, wie viele Temperaturwerte abgerufen und in eine Datei geschrieben werden. Das Diagramm zeigt die Temperaturkurve. Ändern Sie die Skala für die Y-Achse, so daß Werte zwischen 70,0 und 90,0 angezeigt werden, und richten Sie die Skala für die X-Achse auf Werte zwischen 0 und 20 ein.

2. Rufen Sie das Popup-Menü für das numerische Eingabefeld “Anzahl der Punkte” auf, und wählen Sie **Repräsentierung»I32**.

Blockdiagramm

3. Öffnen Sie das Blockdiagramm.



4. Fügen Sie die For-Schleife hinzu, und vergrößern Sie sie. Dieses VI erzeugt die Anzahl an Temperaturwerten, die über das Bedienelement `Anzahl der Punkte` festgelegt wird.
5. Fügen Sie ein Schieberegister zu der Schleife hinzu, indem Sie das Popup-Menü für den Schleifenrand aufrufen. Dieses Schieberegister enthält die Pfadangabe für die Datei.
6. Stellen Sie die Verbindungen für die Objekte fertig.

Leerer Pfad (Konstante) (**Funktionen»Datei I\O»Dateikonstanten**). Durch die Funktion “Leerer Pfad” wird das Schieberegister so initialisiert, daß beim ersten Versuch, einen Wert in eine Datei zu schreiben, keine Pfadangabe vorhanden ist. Durch ein Dialogfeld werden Sie aufgefordert, einen Dateinamen einzugeben.

Das VI “Digitales Thermometer” gibt die simulierte Temperaturmessung eines Temperatursensors aus.

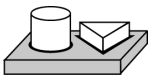
Die Funktion “In String formatieren” (**Funktionen»String**) wandelt die Temperaturmessung (eine Zahl) in einen String um und verknüpft das nachfolgende Komma.

String-Konstante (**Funktionen»String**). Durch diesen Format-String können Sie festlegen, daß eine Zahl in einen String in Bruchformat umgewandelt wird und daß ein Komma auf den String folgt.

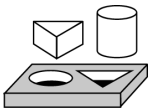
Das VI “Zeichen in Datei schreiben” (**Funktionen»Datei I/O)** schreibt einen String aus Zeichen in eine Datei.

Durch die Funktion “Boolesche Konstante” (**Funktionen»Boolesch**) wird die Eingabe `an Datei anhängen?` des VIs “Zeichen in Datei schreiben” auf “Wahr” gesetzt, so daß die neuen Temperaturwerte bei jeder Schleifenwiederholung an die ausgewählte Datei angehängt wird. Klicken Sie mit dem Bedienwerkzeug auf die Konstante, um ihren Wert auf “Wahr” zu setzen.

7. Kehren Sie zum Frontpanel zurück, und führen Sie das VI mit der Einstellung 20 für `Anzahl der Punkte` aus. Durch ein Dateidialogfeld werden Sie zur Eingabe eines Dateinamens aufgefordert. Wenn Sie einen Dateinamen eingeben, beginnt das VI damit, die Temperaturwerte für jeden erzeugten Punkt in die angegebene Datei zu schreiben.
8. Speichern Sie das VI unter dem Namen `Write Temperature to File.vi` im Verzeichnis `LabVIEW\Activity`.
9. Verwenden Sie ein beliebiges Textverarbeitungsprogramm, wie z.B. Write für Windows, TeachText für Macintosh oder einen Text-Editor in UNIX, um die Datendatei zu öffnen und den Inhalt einzusehen. Die geöffnete Datei sollte zwanzig Datenwerte (mit einer Genauigkeit von drei Stellen hinter dem Dezimalzeichen) enthalten, die durch Kommas voneinander getrennt sind.



Ende der Übung 6-5.

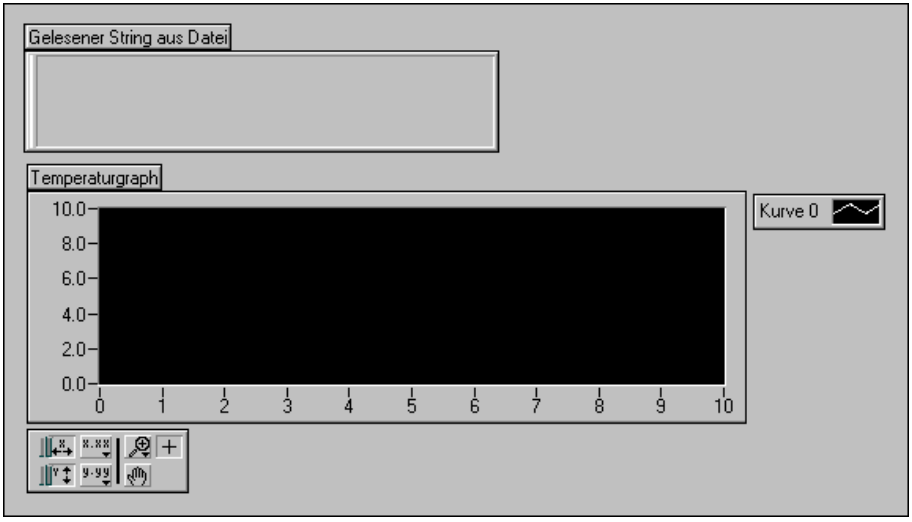


Übung 6-6. Daten aus einer Datei lesen

Übungsziel ist das Erstellen eines VIs, das die im vorherigen Beispiel angelegte Datei liest und die Daten in einem Kurvengraphen anzeigt. Die Daten müssen im selben Datenformat gelesen werden, in dem sie gespeichert wurden. Da Sie die Daten ursprünglich unter Verwendung von String-Datentypen in ASCII-Format gespeichert haben, müssen Sie sie auch als String-Daten mit einem der Datei I/O-VIs einlesen.

Frontpanel

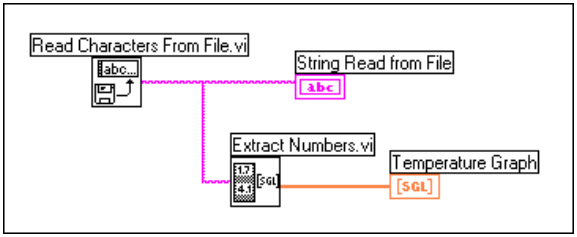
1. Öffnen Sie ein neues Frontpanel, und gestalten Sie es entsprechend der folgenden Abbildung.



Auf dem Frontpanel befinden sich eine String-Anzeige und ein Kurvengraph. Die Anzeige “Gelesener String aus Datei” zeigt die kommabegrenzten Temperaturdaten aus der Datei an, die Sie in der vorherigen Übung eingerichtet haben. Der Graph stellt die Temperaturkurve dar.

Blockdiagramm

2. Stellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm zusammen.

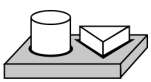


Das VI “Zeichen aus Datei lesen” (**Funktionen»Datei I/O**) liest die Daten aus der Datei und gibt die Informationen in einem String aus. Wenn keine Pfadangabe vorhanden ist, werden Sie in einem Dialogfeld aufgefordert, einen Dateinamen einzugeben. In diesem Beispiel ist es nicht notwendig, die Anzahl der zu lesenden Zeichen zu bestimmen, da die Datei weniger als die Standardanzahl von 512 Zeichen enthält.

Um die Daten aus einer Datei lesen zu können, müssen Sie wissen, wie die Daten gespeichert wurden. Wenn Ihnen die Länge einer Datei bekannt ist, können Sie das VI “Zeichen aus Datei lesen” verwenden, um die Ihnen bekannte Anzahl an Zeichen zu lesen.

Das VI “Zahlen extrahieren” (`Examples\General\strings.llb`) bearbeitet einen ASCII-String, der durch Komma, Zeilenvorschübe oder andere nicht-numerische Zeichen voneinander getrennte Zahlen enthält, und wandelt die Zahlen in ein Array aus numerischen Werten um.

3. Kehren Sie zum Frontpanel zurück, und führen Sie das VI aus. Wählen Sie in dem eingeblendeten Datendialogfeld die Datendatei aus, die Sie gerade in das Speichermedium geschrieben haben. Im Graphen sollten dieselben Datenwerte angezeigt werden, die Sie auch im Beispiel für das VI “Temperatur an Datei schreiben” gesehen haben.
4. Speichern Sie das VI unter dem Namen `Temperature from File.vi`, und schließen Sie es.



Ende der Übung 6-6.

Datei I/O-Funktionen verwenden

Die übergeordneten Datei I/O-Funktionen bieten manchmal nicht die Funktionalität, die Sie eventuell zum Speichern von Daten in einem Speichermedium benötigen. In diesem Fall müssen Sie die Funktionen verwenden, die über **Funktionen»Datei I/O»Fortgeschritten** zur Verfügung stehen.

Datei bestimmen

Es gibt zwei Möglichkeiten, eine Datei zu bestimmen — programmatisch oder über ein Dialogfeld. Bei der programmatischen Methode geben Sie den Dateinamen und den Pfadnamen für das VI vor.

(Windows) Ein Pfadname besteht aus dem Laufwerksnamen (z.B. `C`), einem anschließenden Doppelpunkt, den durch einen Backslash voneinander getrennten Verzeichnisnamen und abschließend dem Dateinamen. Der Pfadname `C:\DATADIR\TEST1` beschreibt z.B. eine als `TEST1` bezeichnete Datei im Verzeichnis `DATADIR` auf dem Laufwerk `C`.

(Macintosh) Ein Pfadname besteht aus dem Laufwerksnamen, einem anschließenden Doppelpunkt, den durch Doppelpunkt voneinander getrennten Ordnernamen und abschließend dem Dateinamen.
Der Pfadname `HardDrive:DataFolder:Test1` beschreibt z.B. eine als `Test1` bezeichnete Datei im Ordner `DataFolder` auf dem Volume mit der Bezeichnung `HardDrive`.

(UNIX) Ein Pfadname besteht aus den durch Schrägstrich voneinander getrennten Verzeichnisnamen, auf die der Dateiname folgt. Der Pfadname `/usr/datadirectory/test1` beschreibt z.B. eine als `test1` bezeichnete Datei im Verzeichnis `/usr/datadirectory`.

(Alle Plattformen) In der Dialogfeldmethode wird durch die Dateidialogfunktion ein Dialogfeld geöffnet, mit dem Sie interaktiv nach einem Verzeichnis suchen und dann einen Dateinamen eingeben können.

Pfade und Refnums

Ein Pfad ist ein G-Datentyp, der zur Identifikation von Dateien dient. Mit dem Pfad-Bedienelement und der Pfad-Anzeige können Sie einen Dateipfad in der Standardsyntax für eine spezielle Plattform anzeigen. In vieler Hinsicht entsprechen das Pfad-Bedienelement bzw. die Pfad-Anzeige einem String-Bedien- bzw. -Anzeigeelement; G formatiert

allerdings den Pfad entsprechend den Anforderungen für alle Plattformen, die unterstützt werden.

Ein Refnum besteht aus einem G-Datentyp, durch den offene Dateien gekennzeichnet werden. Wenn Sie eine Datei öffnen, gibt G ein Refnum zurück, das mit der Datei assoziiert ist. Alle Bearbeitungsvorgänge, die an geöffneten Dateien durchgeführt werden, verwenden die Datei-Refnums zur Identifikation der jeweiligen Datei. Ein Refnum ist nur gültig, solange die Datei geöffnet ist. Wenn Sie die Datei schließen, hebt G die Assoziation zwischen dem Refnum und der Datei auf. Wenn Sie die Datei anschließend erneut öffnen, ist das neue Refnum möglicherweise nicht mit dem zuvor von G verwendeten Refnum identisch.

G assoziiert nicht nur einen Bearbeitungsvorgang mit einer Datei, sondern speichert auch zusätzliche Informationen für jedes Refnum, so z.B. die aktuelle Position beim Lesen der Datei und die Zugangsberechtigung, die andere Benutzer dieser Datei haben. Sie können also gleichzeitig, aber unabhängig voneinander verschiedene Bearbeitungsvorgänge an einer einzelnen Datei durchführen. Wenn Sie eine Datei mehrmals öffnen, wird bei jedem Öffnen der Datei ein anderes Refnum ausgegeben.

Datei I/O-Beispiele

Anhand der folgenden Beispiele können Sie erkennen, wie die Datei I/O-Funktionen mit korrekt angewendeten Methoden zur Fehlerprüfung und Bedienung verwendet werden:

Das VI “In Textdatei schreiben” (in `Examples\File\smplfile.llb`) erstellt eine ASCII-Textdatei, die Datenwerte mit Zeitstempeln enthält.

Das VI “Aus Textdatei lesen” (in `Examples\File\smplfile.llb`) liest eine ASCII-Textdatei, die Datenwerte mit Zeitstempeln enthält.

Datenprotokolldateien

Die in diesem Kapitel gezeigten Beispiele illustrieren einfache Methoden für den Umgang mit Dateien, in denen die Daten als Sequenz von ASCII-Zeichen gespeichert sind. Das ASCII-Format wird häufig verwendet, um Dateien zu erstellen, die mit anderen Software-anwendungen gelesen werden, z.B. mit einem Tabellenkalkulations-programm. G bietet außerdem ein anderes Dateiformat, das als *Datenprotokolldatei* bezeichnet wird. In einer Datenprotokolldatei werden die Daten als Sequenz von Datensätzen in einem einzigen arbiträren Datentyp gespeichert, den Sie beim Erstellen der Datei

bestimmen. G indexiert die Daten in einer Datenprotokolldatei aufgrund dieser Datensätze. Alle Datensätze in einer Datenprotokolldatei müssen zwar einem einzigen Typ angehören; dieser Typ kann jedoch komplex gestaltet sein. Sie können z.B. jeden Datensatz so einrichten, daß der Datensatz ein Cluster aus einem String, einer Zahl und einem Array enthält.

Wenn Sie die Daten mit einem VI abrufen, empfiehlt es sich, die Daten nicht in ASCII-Dateien zu schreiben, da die Umwandlung von Daten in und aus Strings viel Zeit in Anspruch nehmen kann. Die Umwandlung eines zweidimensionalen Arrays in einen String in einem Tabellenformat mit Kopfzeilen und Zeitstempeln ist ein komplizierter Vorgang. Wenn die Daten nicht unbedingt in einem Format gespeichert werden müssen, das für andere Anwendungen lesbar ist, ist es sinnvoll, die Daten in eine Datenprotokolldatei zu schreiben. In diesem Format genügt ein geringes Maß an Manipulation, und das Schreiben und Lesen wird beschleunigt. Auch das Abrufen von Daten wird dadurch vereinfacht, da die ursprünglichen Datenblocks als Protokoll oder Datensatz zurückgelesen werden können, ohne daß bekannt sein muß, wie viele Datenbytes die Datensätze enthalten. G hält die Datenmenge für jeden Datensatz in einer Datenprotokolldatei fest.

Das Beispiel-VI “Datenprotokolldatei schreiben” (in `Examples\File\datalog.llb`) erstellt eine neue Datenprotokolldatei und schreibt die angegebene Anzahl an Datensätzen in die Datei. Jeder Datensatz stellt ein Cluster dar, das einen String und ein Array mit Zahlen in einfacher Genauigkeit enthält.

Wenn Sie eine Datenprotokolldatei lesen wollen, müssen Sie denselben Datentyp verwenden, der beim Schreiben in die Datei benutzt wurde. Das Beispiel-VI “Datenprotokolldatei lesen” (in `Examples\File\datalog.llb`) liest in der Datenprotokolldatei, die durch das Beispiel-VI “Datenprotokolldatei schreiben” erstellt wurde, einen Datensatz nach dem anderen. Der gelesene Datensatz besteht aus einem Cluster, das einen String und ein Array mit Zahlen in einfacher Genauigkeit enthält.

Teil II

I/O-Schnittstellen

Dieser Teil des Handbuchs enthält grundsätzliche Informationen zu den Schnittstellen, über die Daten ein- und ausgegeben werden können. Dazu gehören: Datenerfassung, GPIB, Seriell und VXI. Im LabVIEW-Handbuch *Grundlagen der Datenerfassung* finden Sie einführende Informationen zur Echtzeit-Datenerfassung. VISA (Virtual Instrument Software Architecture) bietet die Möglichkeit, über eine einzige Software eine Schnittstelle zur Programmierung von Anwendungsprogrammen, eine sogenannte API, für GPIB-, serielle und VXI-Instrumente zu schaffen. Die LabVIEW-Anwendungen, die für ein spezielles Instrument entwickelt wurden, werden als Instrumententreiber bezeichnet. National Instruments stellt mehrere Instrumententreiber zur Verfügung, die die VISA-Bibliothek benutzen. Sie können aber ebenso Ihre eigenen Instrumententreiber anfertigen.

Teil II, *I/O-Schnittstellen*, enthält die folgenden Kapitel:

- Kapitel 7, *Erste Schritte mit einem LabVIEW-Gerätetreiber*, erläutert, wie Sie die Instrumententreiber von National Instruments erstellen und verwenden können.
- Kapitel 8, *Tutorial für LabVIEW VISA*, zeigt Ihnen, wie Sie gebräuchliche VISA-Anwendungen durch nachrichten- und registerorientierte Kommunikation implementieren können und wie Sie mit Ereignissen und der Sperrfunktion umgehen.
- Kapitel 9, *Einführung in die LabVIEW GPIB-Funktionen*, erläutert, wie GPIB arbeitet und welche Unterschiede zwischen den Schnittstellen IEEE 488 und IEEE 488.2 bestehen.
- Kapitel 10, *VIs für den seriellen Anschluß*, beschreibt die VIs für die Kommunikation mit dem seriellen Port und erläutert die wichtigen Faktoren, die die serielle Kommunikation beeinflussen.

7

Erste Schritte mit einem LabVIEW-Gerätetreiber

Zu Beginn dieses Kapitels wird erläutert, wie Sie Gerätetreiber aus der Gerätetreiberbibliothek installieren und verwenden können, und am Ende des Kapitels erhalten Sie Anleitungen für das Erstellen Ihres eigenen Gerätetreibers. Dieses Kapitel behandelt Schritt für Schritt das allgemeine Verfahren zur Verifikation der Kommunikation mit einem Gerät, zur Entwicklung einer Anwendung, die Gerätetreiber verwendet, und zum Anfertigen eines Gerätetreibers.

Was ist ein LabVIEW-Gerätetreiber?

Ein Gerätetreiber ist ein Set von LabVIEW-VIs, die mit Hilfe der Standard-VISA-I/O-Funktionen von LabVIEW mit einem Instrument oder Gerät kommunizieren. Jedes VI entspricht einem programmatischen Vorgang, z.B. dem Konfigurieren, dem Lesen und Schreiben von Daten und der Triggerung eines Geräts. Dank der LabVIEW-Gerätetreiber ist es nicht notwendig, die komplexen, auf unteren Ebenen wirksamen Programmierbefehle für jedes Gerät zu beherrschen.

Die LabVIEW-Gerätetreiberbibliothek enthält Gerätetreiber für eine Reihe von programmierbaren Geräten und Instrumenten, die die GPIB-, VXI- oder serielle Schnittstelle verwenden. Sie können einen Bibliothekstreiber in der vorliegenden Form für Ihr Gerät verwenden. Die Gerätetreiber werden jedoch mit dem Quellcode für das Blockdiagramm bereitgestellt, so daß Sie sie gegebenenfalls für Ihre spezielle Anwendung anpassen können.

Wo finde ich Gerätetreiber?

Die Gerätetreiber können von einer Gerätetreiber-CD installiert oder von der Website von National Instruments heruntergeladen werden. Sie erhalten das aktuellste Bestellformular für Gerätetreiber, wenn Sie Faxabruf, das automatisierte Faxsystem von National Instruments, über ein Telefon mit Frequenzwahlverfahren unter folgender Rufnummer anrufen:

+1 (512) 418-1111 oder (800) 329-7177 (nur in Nordamerika). Sie können die Treiber auch über das Instrument Driver Network im World Wide Web herunterladen. Sie erhalten Zugriff auf dieses Netzwerk über `http://www.natinst.com/idnet`.

Wenn für das von Ihnen verwendete Gerät keine Gerätetreiber vorhanden sind, können Sie folgendermaßen vorgehen:

1. Versuchen Sie, einen Treiber für ein ähnliches Gerät zu verwenden. Ähnliche Geräte vom selben Hersteller verwenden oft ähnliche oder sogar identische Befehlsätze.
2. Erstellen Sie einen Gerätetreiber, und befolgen Sie dabei die Richtlinien im Abschnitt *Ein schnell und einfach zu entwickelnder LabVIEW-Gerätetreiber* in diesem Kapitel.
3. Entwickeln Sie einen vollständigen, voll funktionsfähigen Gerätetreiber. Die Anwendungshinweise 006, *Developing a LabVIEW Instrument Driver (Entwickeln eines LabVIEW-Gerätetreibers)*, die Sie von unserer Website herunterladen können, erläutern, wie Sie einen Qualitätstreiber, der den Anforderungen von National Instruments entspricht, entwickeln können. Diese Anwendungshinweise helfen Ihnen dabei, einen vollständigen Gerätetreiber herzustellen.

Wo sollte der LabVIEW-Gerätetreiber installiert werden?

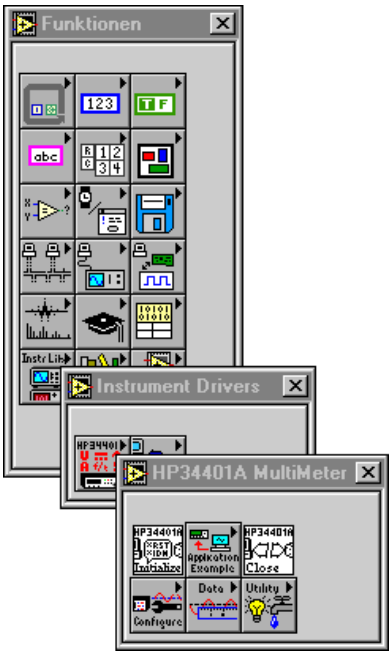
Die Gerätetreiber sollten in einem Unterverzeichnis von `LabVIEW/instr.lib` installiert werden. Der Gerätetreiber HP34401A, der mit LabVIEW geliefert wird, ist z.B. in folgendem Verzeichnis installiert:

```
Labview/instr.lib/hp34401a
```

In diesem Verzeichnis finden Sie die Menüdateien und die VI-Bibliotheken, aus denen sich ein Gerätetreiber zusammensetzt. Die Menüdateien ermöglichen das Anzeigen der Gerätetreiber-VIs über die Palette **Funktionen**. Die VI-Bibliotheken enthalten die Gerätetreiber-VIs.

Wie erfolgt der Zugriff auf die Gerätetreiber-VIs?

Sie finden die Gerätetreiber-VIs im unteren Bereich der Palette **Funktionen** in der Unterpalette **Gerätetreiber**.



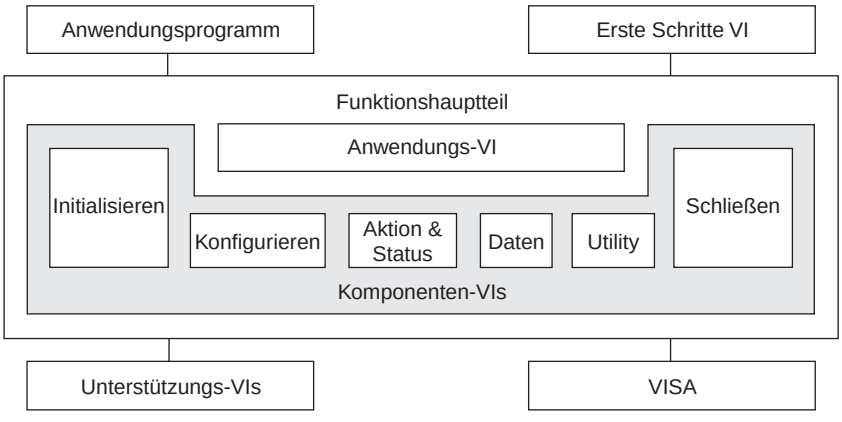
Viele der Gerätetreiber sind mit Menüpaletten ausgestattet, die die folgenden Komponenten enthalten:

- VI initialisieren
- VI schließen
- Anwendungsbeispiel-Unterpalette
- Konfigurations-Unterpalette
- Aktion/Status-Unterpalette
- Daten-Unterpalette
- Utility-Unterpalette

Der Zugriff auf Gerätetreiber ist auch über die Option **VI auswählen** in der Palette **Funktionen** möglich. Wenn Sie die vollständige Gerätetreiber-Hierarchie sehen möchten, können Sie das VI “VI-Baum” öffnen. Es handelt sich dabei um ein nicht ausführbares VI, das zum Anzeigen der funktionalen Struktur des Gerätetreibers dient.

Gerätetreiber-Struktur

Die folgende Abbildung zeigt, wie ein Standard-Gerätetreiber organisiert ist. Wenn Sie sich mit diesem Modell vertraut gemacht haben, werden Sie feststellen, daß es sich auf zahlreiche Gerätetreiber übertragen läßt.

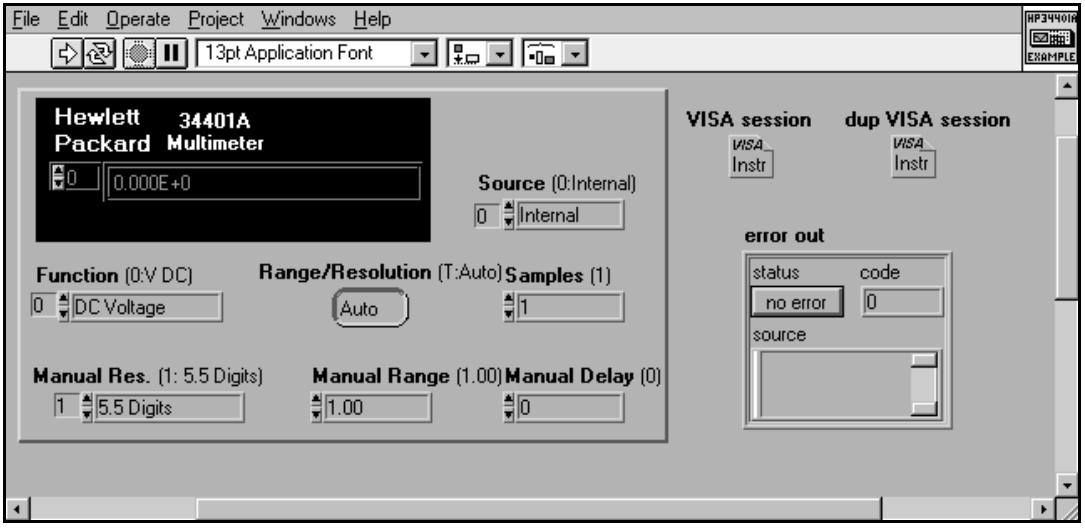


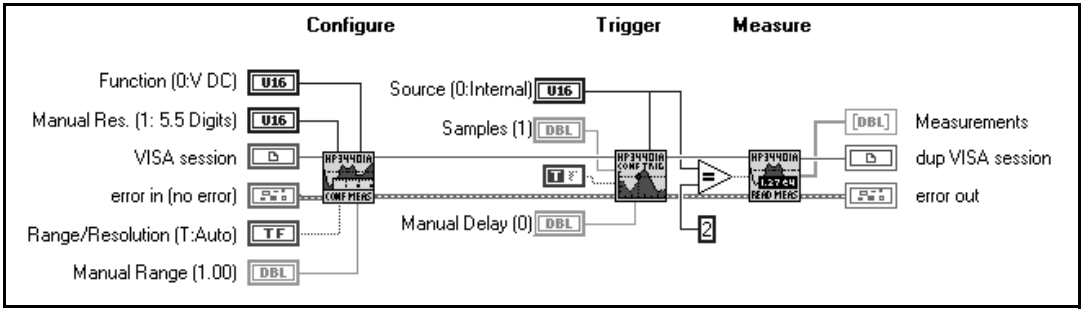
Modell eines Gerätetreibers

Die VIs “Getting Started” (“Erste Schritte”) stellen einfache Anwendungs-VIs dar, die Sie ohne weitere Veränderung verwenden können. Führen Sie dieses VI aus, um die Kommunikation mit dem Gerät zu überprüfen. In der Regel müssen Sie lediglich die Geräteadresse ändern, bevor Sie das VI vom Frontpanel aus ausführen. Bei einigen VIs müssen Sie jedoch zusätzlich den VISA-Ressourcennamen angeben (z.B. GPIB::2). Weitere Informationen zu VISA-Ressourcennamen finden Sie in Kapitel 8, *Tutorial für LabVIEW VISA*. Das VI “Getting Started” besteht im allgemeinen aus drei SubVIs: dem Initialisieren-VI, einem Anwendungs-VI und dem Schließen-VI.

Die Anwendungs-VIs sind High-Level-Beispiele für das Zusammenfassen von Low-Level-Komponentenfunktionen, wodurch das Ausführen eines typischen programmatischen Gerätevorgangs ermöglicht wird. In den Anwendungs-VIs können VIs zusammengefaßt sein, durch die die am häufigsten verwendeten Gerätekonfigurationen und -messungen gesteuert werden. Diese VIs dienen als Code-Beispiel für das Ausführen eines häufig wiederholten Vorgangs, z.B. des Konfigurierens eines Geräts, der Triggerung und des eigentlichen Meßvorgangs.

Da es sich bei den Anwendungs-VIs um Standard-VIs mit Icons und Anschlußfeldern handelt, können Sie sie von jeder High-Level-Anwendung aus aufrufen, wenn Sie den Treiber mit einer meßorientierten Einzel-Schnittfläche versehen möchten. Viele Benutzer benötigen außer den Anwendungs-VIs keine weiteren Gerätetreiber-VIs zur Gerätesteuerung. Das HP34401-Beispiel-VI, das in der folgenden Abbildung dargestellt ist, zeigt das Frontpanel und Blockdiagramm eines Anwendungs-VIs.





Das *Initialisieren-VI*, d.h. das zuerst aufgerufene Gerätetreiber-VI, stellt die Kommunikation mit dem Gerät her. Darüber hinaus kann es alle erforderlichen Aufgaben ausführen, um das Gerät entweder in den Standard-Einschaltzustand oder in einen anderen speziellen Zustand zu schalten. In der Regel muß das *Initialisieren-VI* nur einmal zu Beginn des Anwendungsprogramms aufgerufen werden.

Die *Konfigurations-VIs* sind eine Ansammlung von Software-Routinen, die das Gerät so konfigurieren, daß es den gewünschten Vorgang ausführt. Je nach Gerät können zahlreiche *Konfigurations-VIs* vorhanden sein. Nachdem diese VIs aufgerufen wurden, ist das Gerät bereit, Messungen durchzuführen oder ein System zu stimulieren.

Die Kategorie *Aktion/Status* enthält zwei VI-Typen. *Aktion*-VIs initiieren oder terminieren Test- und Meßvorgänge. Zu diesen Vorgängen können das Aktivieren des Trigger-Systems oder das Erzeugen eines Stimulus zählen. Diese VIs unterscheiden sich dadurch von den *Konfigurations-VIs*, daß sie die Geräteeinstellungen nicht verändern, sondern nur bewirken, daß das Gerät eine Aktion aufgrund seiner aktuellen Konfiguration ausführt. Die *Status-VIs* ermitteln den aktuellen Status des Geräts oder den Status der momentan ausgeführten Vorgänge.

Die *Daten-VIs* übertragen Daten in das Gerät oder aus dem Gerät. Beispiele sind VIs zum Lesen eines durch ein Meßgerät gemessenen Werts oder eines Signalverlaufs, VIs zum Herunterladen von Signalverläufen oder digitalen Mustern in ein Quellgerät.

Die *Utility-VIs* führen eine Reihe von Vorgängen aus, die die am häufigsten benutzten Gerätetreiber-VIs unterstützen. Zu diesen VIs zählen die Mehrzahl der Gerätetreiber-Vorlagen-VIs, wie z.B. VIs für das Zurücksetzen, das Durchführen von Selbsttests, die Revision, die Fehlerabfrage und das Ausgeben von Fehlermeldungen; zu dieser Gruppe können auch noch andere angepaßte Gerätetreiber-VIs gehören, die solche Vorgänge wie das Kalibrieren oder Speichern und das Wiederaufrufen von Setup-Einstellungen ausführen.

Das *Schließen-VI* terminiert die Softwareverbindung zum Gerät und gibt dadurch Systemressourcen wieder frei. In der Regel muß das *Schließen-VI* nur einmal am Ende des Anwendungsprogramms oder nach Abschluß der Kommunikation mit dem Gerät aufgerufen werden. Sie sollten sicherstellen, daß Sie jedesmal, wenn Sie das Initialisieren-VI erfolgreich aufrufen, auch über ein entsprechendes Schließen-VI verfügen — andernfalls werden Speicherressourcen unnötig blockiert.

☞ **Hinweis** ***Die Anwendungsfunktionen rufen das Initialisieren- und Schließen-VI nicht direkt auf. Wenn Sie eine Anwendungsfunktion ausführen möchten, müssen Sie zuerst das Initialisieren-VI ausführen. Das VI "Getting Started" initialisiert und schließt selbständig.***

Hilfe bei der Arbeit mit Gerätetreiber-VIs

Im **Hilfe**-Fenster von LabVIEW finden Sie Hinweise für die Arbeit mit LabVIEW-Gerätetreibern. Für jedes Gerätetreiber-VI sowie für jedes Bedienelement auf dem Frontpanel wird eine **Hilfe**-Beschreibung angeboten. Wählen Sie **Hilfe anzeigen** im Menü **Hilfe**, um das **Hilfe**-Fenster anzuzeigen. Ziehen Sie den Cursor auf das VI-Symbol, wenn Sie Hilfe zu einem VI benötigen. Um Hilfe für ein Bedienelement des Frontpanels zu erhalten, müssen Sie den Cursor auf das entsprechende Bedienelement ziehen. Wenn das **Hilfe**-Fenster keine vollständige Beschreibung enthält, können Sie weitere Hilfe zu Bedien- oder Anzeigeelementen aufrufen, indem Sie **Datenoperationen» Beschreibung...** im Popup-Menü für das Bedien- oder Anzeigeelement wählen.

Interaktiv das VI “Getting Started” ausführen
(Auswahl der GPIB-Adresse, des seriellen Anschlusses und der logischen Adresse)

Wenn Sie die Kommunikation mit einem Gerät oder Instrument überprüfen und einen typischen programmatischen Gerätevorgang testen möchten, sollten Sie zuerst das VI “Getting Started” öffnen. Kontrollieren Sie die verschiedenen Bedienelemente, und legen Sie angemessene Einstellungen fest. Mit Ausnahme des Adreßfeldes sind die Standardeinstellungen der meisten Bedienelemente im allgemeinen für den ersten Durchlauf ausreichend. Die Adresse muß den Gegebenheiten angepaßt werden. Wenn Ihnen die Adresse Ihres Geräts nicht bekannt ist, erhalten Sie Hilfe durch den Instrument Wizard. Nachdem Sie das VI ausgeführt haben, sollten Sie überprüfen, ob angemessene Daten ausgegeben wurden und ob nicht eventuell ein Fehler im Fehler-Cluster gemeldet wurde. In den häufigsten Fällen sind Probleme beim Ausführen des VIs “Getting Started” auf die folgenden Ursachen zurückzuführen:

1. NI-VISA ist nicht installiert. Falls Sie diese Option nicht bereits bei der Installation von LabVIEW gewählt haben, müssen Sie NI-VISA installieren, bevor Sie das VI “Getting Started” erneut ausführen.
2. Die Geräteadresse ist falsch. Das VI “Getting Started” benötigt die korrekte Adresse für Ihr Gerät. Wenn Sie nicht sicher sind, wie die Adresse für das Gerät lautet, sollten Sie den Instrument Wizard oder die Funktion “Find Resource” ausführen. Wenn Sie mit der Syntax für den Adreß-String nicht vertraut sind, finden Sie Hilfe in Kapitel 8, *Tutorial für LabVIEW VISA*. Sie können auf den Instrument Wizard

zugreifen, indem Sie **Solution Wizard** im LabVIEW-Dialogfeld wählen. Wählen Sie **Instrument Wizard**, wenn Sie zur Auswahl aufgefordert werden.

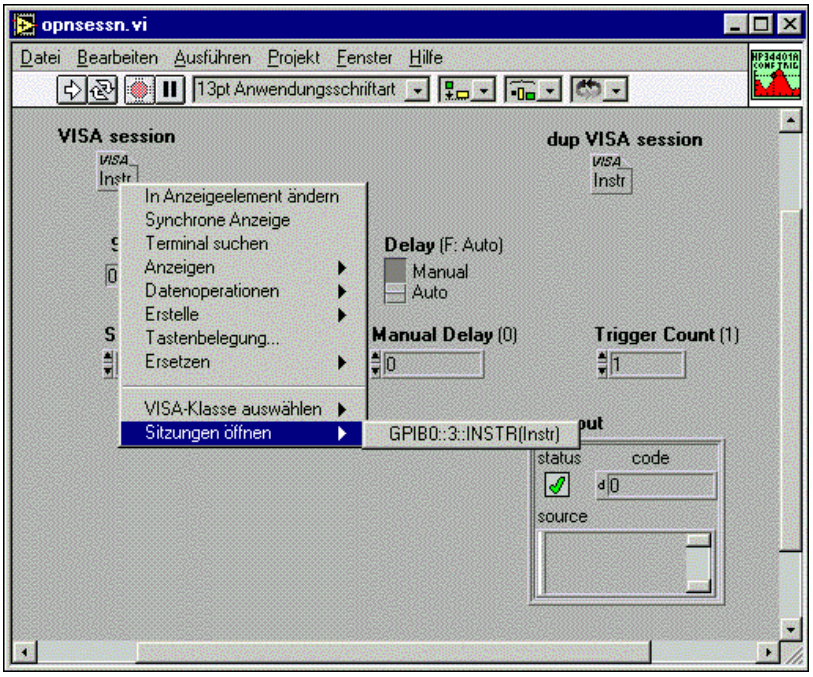
3. Der Gerätetreiber unterstützt nicht das spezielle Modell, das Sie verwenden. Sie sollten noch einmal überprüfen, ob der Gerätetreiber tatsächlich das verwendete Gerätemodell unterstützt.

Nachdem Sie die Kommunikation mit dem Gerät mit dem VI “Getting Started” grundsätzlich überprüft haben, müssen Sie wahrscheinlich die Gerätesteuerung Ihren Anforderungen anpassen. Falls für Ihre Anwendung ähnliche Anforderungen erfüllt sein müssen wie für das VI “Getting Started”, besteht die einfachste Möglichkeit zum Erstellen eines benutzerspezifischen VIs darin, eine Kopie des VIs “Getting Started” zu speichern. Wählen Sie dazu **Speichern unter...** im Menü **Datei**.
Sie können die Standardwerte auf dem Frontpanel ändern, indem Sie **Aktuelle Werte als Standard übernehmen** im Menü **Ausführen** wählen. Bei der Änderung des Blockdiagramms können Sie unter anderem die Konstanten ändern, die mit dem Anwendungs-VI oder anderen SubVIs verbunden sind. Wie oben bereits erwähnt, besteht das Blockdiagramm des VIs “Getting Started” im allgemeinen aus drei VIs: dem Initialisieren-VI, einem Anwendungsfunktions-VI und dem Schließen-VI.

Komponenten-VIs interaktiv testen

Viele Benutzer ziehen es vor, Komponenten-VIs interaktiv zu testen, bevor Sie sie in ihre Anwendung übernehmen. Dadurch wird das Auswählen der passenden Konfigurationseinstellungen für das Gerät erleichtert. Wenn Sie Komponenten-VIs von dem entsprechenden Frontpanel aus ausführen möchten, müssen Sie zuerst das Initialisieren-VI aufrufen. Für alle weiteren VIs müssen Sie zuerst wie in der folgenden Abbildung das

Popup-Menü für das VISA Session-Bedienelement aufrufen und im Untermenü **Sitzungen öffnen...** den entsprechenden Ressourcennamen wählen:



Wenn Sie die Ressource gewählt haben, können Sie das Komponenten-VI interaktiv vom Frontpanel aus mehrfach ausführen, ohne die Ressourcenauswahl jedesmal neu einstellen zu müssen.

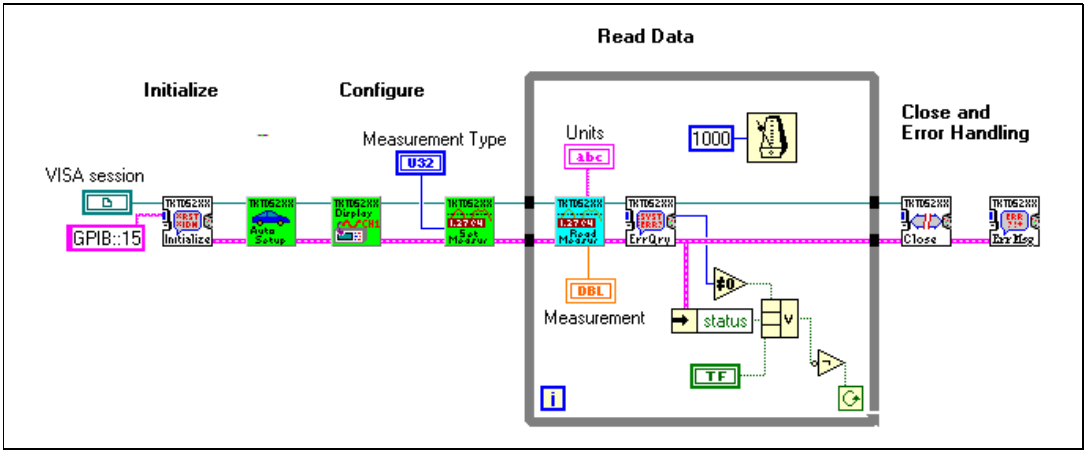
In der Regel sollten Sie die Komponenten-VIs in der Reihenfolge ausführen, in der sie in der Anwendung aufgerufen werden sollen. Führen Sie dabei zuerst das Initialisieren-VI aus und anschließend ein oder mehrere Konfigurations-VIs. Wenn Sie einen Trigger für die Messung verwenden, müssen Sie unter Umständen ein Aktions-VI aufrufen, um den Trigger zu aktivieren. Durch Aufrufen der Daten-VIs werden danach die gemessenen Werte gesammelt. Wenn Sie den Test der Komponenten-VIs für den Gerätetreiber abgeschlossen haben, sollten Sie das Schließen-VI ausführen, um die Ressourcen wieder freizugeben.

Anwendungen erstellen

Nachdem Sie die benötigten Komponenten-VIs ausgewählt und die Reihenfolge ihrer Ausführung bestimmt haben, können Sie Ihre Anwendung erstellen. Falls die Ausführreihenfolge ähnlich wie bei dem Anwendungs-VI ist, können Sie einfach das Blockdiagramm des Anwendungs-VIs abändern. Falls Ihre Anwendung deutliche Unterschiede zum Anwendungs-VI aufweist, ist es sinnvoller, wenn Sie ein ganz neues VI zusammenstellen.

Sie sollten die VIs auf Ihrem Blockdiagramm in der gewünschten Reihenfolge plazieren und sie dann mit Hilfe der VISA Session- und der Fehler-Cluster-Parameter miteinander verbinden. Es ist dabei nicht notwendig, eine Verbindung für alle Eingänge von allen Komponenten-VIs herzustellen. Wenn die Standardwerte für Ihre Anwendung ausreichen, müssen Sie keine Verbindung für die Eingangsterminals herstellen. Wichtige Eingänge sollten Sie allerdings trotzdem verbinden, da Sie auf diese Weise die Arbeit des VIs dokumentieren können. Falls Sie einzelne Messungen wiederholt durchführen möchten, sollten Sie die Datenmessungs-VIs in eine Schleife einsetzen. Denken Sie daran, daß Sie die Indizierung für die VISA Session- und die Fehler-Cluster-Verbindungen, die in die Schleife führen bzw. aus ihr herausführen, deaktivieren müssen, wenn Sie ein Komponenten-VI in eine Schleife einsetzen.

Sie sollten in regelmäßigen Abständen das Fehlerabfrage-VI aufrufen, um das Gerät auf Fehler zu überprüfen. Die folgende Abbildung zeigt ein Oszilloskop, das zur Frequenzmessung in Abständen von jeweils einer Sekunde eingesetzt wird und die Meßergebnisse dem Benutzer anzeigt. Beachten Sie, daß drei Bedingungen zum Abbruch der Schleife führen können: Der Benutzer hält das VI über ein Bedienelement auf dem Frontpanel an; das Fehlerabfrage-VI stellt einen Fehler fest, oder in der VISA I/O-Schnittstelle tritt ein Fehler auf. Falls in der Schleife ein Fehler auftritt, wird durch das Fehlermeldungs-VI eine Meldung für den Benutzer eingeblendet. Das Fehlermeldungs-VI ähnelt dem Allgemeiner-Error-Handler-VI von LabVIEW. Es unterscheidet sich jedoch darin, daß es zusätzlich gerätespezifische Fehler melden kann. Das Fehlermeldungs-VI sollte nach dem Ausführen mehrerer Gerätetreiber-VIs eingesetzt werden, um aufgetretene Fehler festzustellen und anzuzeigen.



Verwandte Themen

Monitor-VI für offene VISA Sessions

Das Monitor-VI für offene VISA Sessions ist praktisch, wenn Sie beabsichtigen, das Debugging für eine Gerätertreiber-Anwendung interaktiv oder programmatisch durchzuführen. Möglicherweise bemerken Sie während des Debugging-Verfahrens, daß viele VISA Sessions geöffnet sind, die geschlossen werden sollten. Wenn Sie eine größere Anzahl von VISA Sessions öffnen, ohne sie wieder zu schließen, verringern sich die verfügbaren Speicherressourcen. Wenn Sie alle offenen Sessions schnell schließen möchten, können Sie das Monitor-VI für offene VISA Sessions in der Bibliothek `labview/vi.lib/utility/visa.llb` aufrufen. Alternativ dazu können Sie aber auch Ihre Arbeit speichern, das Programm beenden und dann LabVIEW neu starten. Beim Beenden von LabVIEW werden alle offenen VISA Sessions geschlossen.

Fehlerbehandlung

Da es mehrere verschiedene Quellen für Fehler geben kann, ist es wichtig, daß Sie Fehler in Anwendungen zur Gerätesteuerung auf angemessene Weise behandeln.

- Die VISA-Funktionen können Fehler melden, weil VISA oder die zugrundeliegende Software oder Hardware nicht korrekt installiert ist. Zur Kommunikation mit GPIB-Geräten muß z.B. NI-488.2 installiert sein, damit die GPIB-Controller-Karte von National Instruments korrekt verwendet werden kann. Wenn die Karte nicht installiert oder

nicht korrekt konfiguriert ist, kann es ebenfalls zu Fehlern mit den Gerätetreiber-VIs kommen. Fehler dieser Art können durch das Fehlermeldungs-VI oder durch das Allgemeiner-Error-Handler-VI von LabVIEW festgestellt werden.

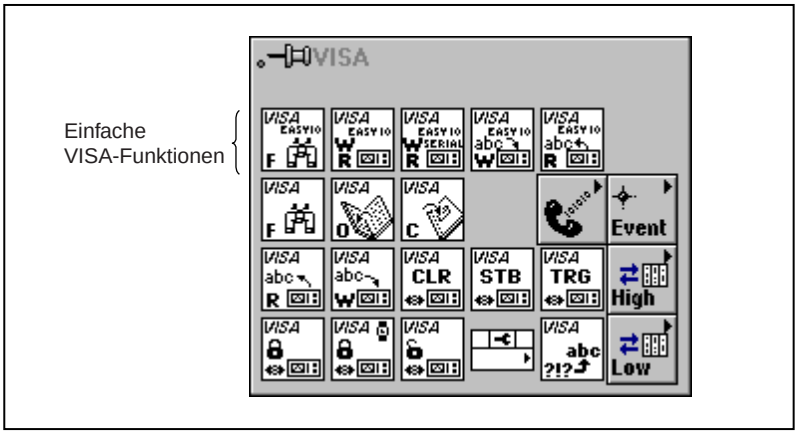
- Die VISA-Funktionen können Fehler melden, wenn das Gerät, auf das Sie zugreifen, nicht auf die gesendeten Befehle reagiert. Unter Umständen werden solche Fehler dadurch verursacht, daß Sie eine falsche Adresse für das Gerät verwenden, daß Fehlfunktionen des Geräts auftreten oder daß das Gerät nicht in der Lage ist, die gesendeten Befehle zu verstehen. Fehler dieser Art können durch das Fehlermeldungs-VI oder durch das Allgemeiner-Error-Handler-VI von LabVIEW festgestellt werden.
- Das Gerät meldet Fehler. In der Regel weist ein Gerät auf eine Reihe von Fehlern hin. Als mögliche Ursachen kommen ungültige Befehle ebenso wie Einstellungen, die außerhalb des gültigen Bereichs liegen, oder fehlende Hardwareoptionen in Frage. Sie können solche Gerätefehler dadurch feststellen, daß Sie das Fehlerabfrage-VI des Gerätetreibers und anschließend das Fehlermeldungs-VI aufrufen.

Weitere Informationen zur Fehlerbehandlung finden Sie im Abschnitt *Die Fehlerbehandlung mit VISA* in Kapitel 8, *Tutorial für LabVIEW VISA.*

Kommunikation mit dem Gerät testen

Wenn sich bei Einsatz der Gerätetreiber-VIs Probleme in der Kommunikation mit dem Gerät ergeben, können Sie einfache Lese- und Schreibvorgänge interaktiv mit den einfachen VISA IO-Funktionen testen, ohne die Funktionen zum Öffnen und Schließen von VISA ausführen zu müssen. Für die VISA Write-Funktion müssen Sie z.B. nur den Ressourcennamen und die an das Gerät zu sendende Meldung angeben. Es stehen die folgenden einfachen VISA IO-Funktionen zur Verfügung, die im Anschluß an die Liste auch in einer Abbildung zu sehen sind:

- VISA Find Resources
- VISA Write
- VISA Read
- VISA Write & Read
- VISA Serial Write & Read
- Register Write
- Register Read



Diese einfachen VISA IO-Funktionen können zwar als nützliche Hilfsmittel beim Testen eingesetzt werden; sie sollten jedoch nur mit Vorsicht bei der Entwicklung von Anwendungen verwendet werden. Bei jedem Schreib- oder Lesevorgang, der das Gerät betrifft, wird durch diese Funktionen jedesmal eine VISA Session geöffnet und geschlossen. Bei wiederholtem Aufrufen kann dadurch die Anwendung deutlich verlangsamt werden. Aus diesem Grund empfiehlt es sich, die VISA-Standardfunktionen zur Anwendungsentwicklung zu verwenden.

Ein schnell und einfach zu entwickelnder LabVIEW-Gerätetreiber

Obwohl National Instruments ständig neue Gerätetreiber entwickelt, reichen die verfügbaren Ressourcen bei weitem nicht aus, Treiber für alle erforderlichen Geräte anzufertigen. Es können sich Situationen ergeben, in denen Sie ein Gerät anschließen möchten, für das kein Treiber vorhanden ist. In diesem Abschnitt wird beschrieben, wie Sie in solchen Fällen einen einfachen Gerätetreiber für eine Anwendung herstellen können.

Vorhandenen Treiber verändern

Bevor Sie mit der Entwicklung eines vollständig neuen Treibers beginnen, sollten Sie feststellen, ob tatsächlich kein Treiber für das Gerät vorhanden ist. Überprüfen Sie dazu sowohl die Website des Herstellers als auch die Website von National Instruments. Bei der Durchsicht der Websites sollten Sie besonders darauf achten, ob nicht unter Umständen Gerätetreiber zur Verfügung stehen, die ein ähnliches Gerät unterstützen. Geräte derselben Modellserien reagieren oft auf dieselben Befehlsätze. SCPI-Geräte mit

vergleichbarer Funktionalität verwenden ebenfalls oft dieselben Befehlssätze. Rufen Sie diese Treiber ab, und überprüfen Sie, ob Sie Ähnlichkeiten zu dem für Ihr Gerät benötigten Befehlssatz feststellen können. Bei Geräten, die zur selben Modellserie gehören, müssen Sie sich unter Umständen direkt an den Hersteller wenden, um detaillierte Informationen zu den Unterschieden in den Befehlen zu erhalten. Wenn Sie ähnliche SCPI-Geräte miteinander vergleichen, müssen Sie die Gerätetreiberbefehle mit den Befehlen vergleichen, die im Programmierhandbuch für Ihr Gerät aufgeführt werden. Unter Umständen müssen Sie einen vorhanden Treiber abwandeln, um den Code zu optimieren. Es ist möglich, daß ein Komponenten-VI, das für den Einsatz durch eine Vielzahl von Benutzern vorgesehen ist, eine Einstellung zu verändern versucht, die nicht für Ihre Anwendung benötigt wird. In der Regel brauchen Sie nur diejenigen VIs zu optimieren, die wiederholt in einer Schleife aufgerufen werden. Konfigurations-VIs werden im allgemeinen nur einmal aufgerufen und haben geringe Auswirkung auf die Geschwindigkeit der Anwendung.

Die einfachste Methode, einen Gerätetreiber abzuändern, besteht darin, ihn durch Auswahl der Option **Speichern unter...** im Menü **Datei** umzubenennen. Zur Kennzeichnung des neuen VIs sollten Sie das Präfix oder die Beschreibung in seinem Namen ändern. Wenn Sie z.B. einen Gerätetreiber für das Oszilloskop Tektronix TDS so modifizieren, daß er für ein anderes Gerät eingesetzt werden kann, sollten Sie die VI-Präfixe von TKTDS7XX in eine Bezeichnung ändern, die Ihrem Gerät angemessen ist. Nachdem Sie den Namen geändert haben, müssen Sie wahrscheinlich das Blockdiagramm und die Bedienelemente auf dem Frontpanel modifizieren. Die meisten Änderungen im Blockdiagramm hängen vermutlich mit den String-Funktionen zusammen. Wenn Sie nicht mit den String-Funktionen (z.B. “Zeile auswählen & anhängen” oder “Auswählen & Anhängen”) vertraut sind, erhalten Sie weitere Informationen hierzu über die *Online-Referenz* von LabVIEW.

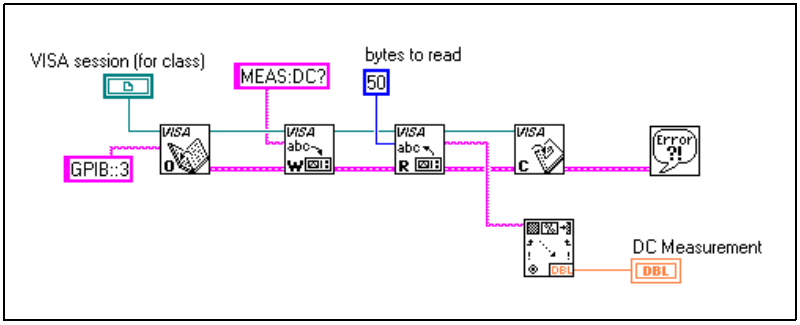
Jedes Initialisieren-VI verfügt über die Option, eine Identifikationsabfrage durchzuführen, die dem speziellen Gerätemodell oder der Modellserie angepaßt ist. Sie müssen entweder diese Option ausschalten oder die Antwort auf den Identifikationsabfragebefehl verändern. Für SCPI-Geräte lautet dieser Befehl “*IDN?”.

Das Ausmaß der Änderungen, die an einem Gerätetreiber durchgeführt werden müssen, hängt davon ab, wie groß die Ähnlichkeit zwischen den Geräten und ihren Befehlssätzen ist. Wenn die Befehlssätze deutlich voneinander abweichen, ist es ratsam, einen vollständig neuen Treiber anzufertigen.

Einfachen Treiber entwickeln

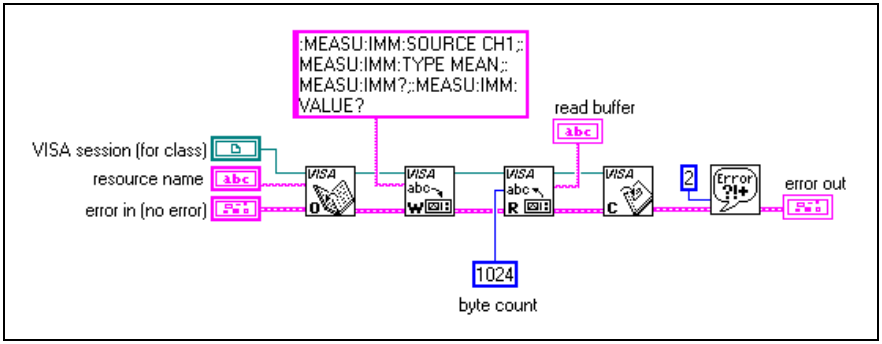
Die meisten Geräte, die Meldungen ausgeben, werden programmatisch durch eine Reihe von Schreib- und Lesevorgängen, die für das Gerät durchgeführt werden, gesteuert. Für die meisten einfachen Geräte reichen vier VISA-Funktionen aus: “VISA Open”, “VISA Write”, “VISA Read” und “VISA Close”.

Das einfache Gerätetreiber-VI, das in der folgenden Abbildung dargestellt ist, führt nur jeweils einen Schreib- und Lesevorgang für das Gerät durch. Dieses VI verwendet zuerst das VI “VISA Open”, um Ressourcen für das Gerät zu öffnen. Danach sendet es den Befehl “MEAS:DC?” (Beschreibung im Benutzerhandbuch für das Gerät), um das Gerät zur Ausgabe einer Gleichstrommessung zu veranlassen. Die Funktion “VISA Read” gibt die Messung in Form eines Strings wieder. Damit die Messung für andere numerische Funktionen verwendbar wird, wird der String mit Hilfe der Funktion “Aus String suchen” in einen numerischen Wert umgewandelt. Nachdem der letzte Lese- bzw. Schreibvorgang für das Gerät durchgeführt wurde, wird die Funktion “VISA Close” aufgerufen. Im Anschluß daran werden Fehler, die eventuell in den Geräte I/O-Funktionen aufgetreten sind, durch das VI “Einfacher Error-Handler” bearbeitet.



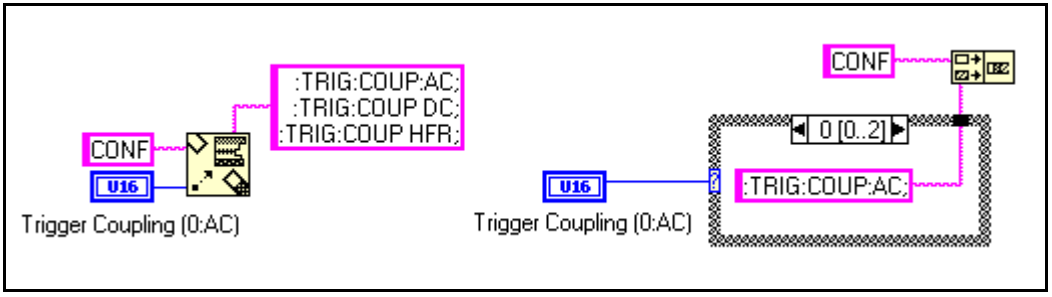
Wenn Sie einen Gerätetreiber entwickeln möchten, der deutlicher modular organisiert ist, können Sie versuchen, die Lese- und Schreibvorgänge für das Gerät so aufzuteilen, daß sie der jeweils von Ihnen beabsichtigten Operation entsprechen. Dabei können Sie z.B. die Lese- und Schreibvorgänge, die für die Konfigurationseinstellungen benötigt werden, in einem VI zusammenfassen, während ein anderes VI für das Lesen der Messungen zuständig ist. Zur Wiederholung von Messungen können Sie die Messungs-VIs in einer Schleife unterbringen. Wenn Ihnen alle Konfigurationseinstellungen genau bekannt sind, können Sie unter

Umständen alle Konfigurationsbefehle wie in der folgenden Abbildung in eine String-Konstante einbeziehen.



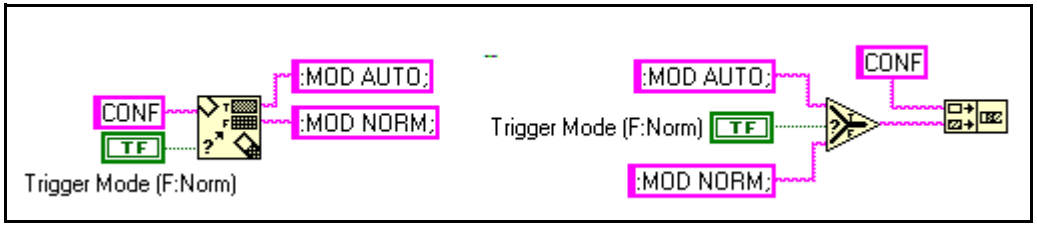
Wenn Sie jedoch Wert darauf legen, daß ein Benutzer unter verschiedenen Konfigurationen auswählen kann, müssen Sie Befehls-Strings programmatisch erstellen. Sie können die Funktion “Zeile auswählen & anhängen” verwenden, um eine Auswahl aus einer Reihe von Strings zu treffen und im selben Schritt eine Verknüpfung mit einem anderen String durchzuführen. Dieses Verfahren ist einfacher als die Verwendung einer Case-Struktur und der Funktion zur Verknüpfung von Strings.

Das Blockdiagramm auf der linken Seite der folgenden Abbildung besitzt eine einfachere Struktur als das Blockdiagramm auf der rechten Seite.

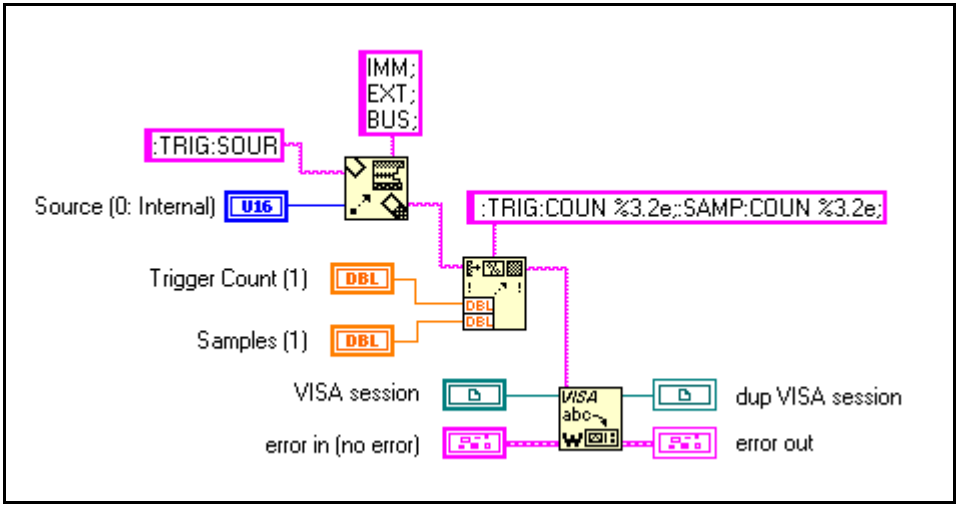


Mit der Funktion “Auswählen & Anhängen” können Sie eine String-Konstante auswählen und sie im selben Schritt mit einem anderen String verknüpfen. Dieses Verfahren ist einfacher als die Verwendung der Auswahlfunktion und der Verknüpfungsfunktion.

Das Blockdiagramm auf der linken Seite der folgenden Abbildung besitzt eine einfachere Struktur als das Blockdiagramm auf der rechten Seite.

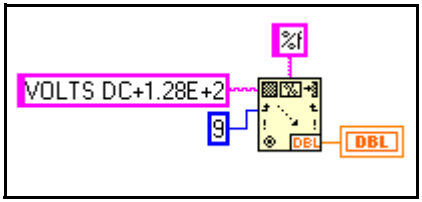


Andere nützliche Funktionen, die beim Erstellen von Befehls-Strings eingesetzt werden können, sind die Funktionen “In String formatieren” und “Formatieren & Anhängen”. Diese Funktionen bieten eine Vielzahl von Formatierungsoptionen für das Umwandeln einzelner oder mehrerer numerischer Werte in einen String. Das folgende Blockdiagramm illustriert die beiden Funktionen “Zeile auswählen & anhängen” und “In String formatieren”:

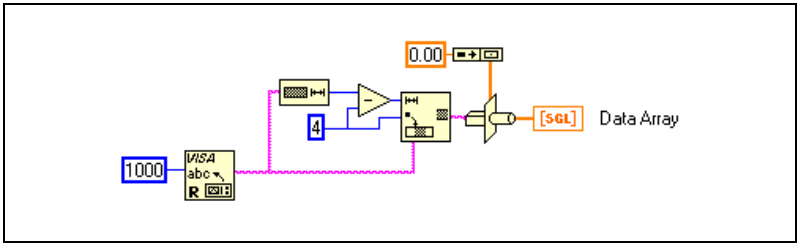


Nachdem Sie das Gerät abgelesen haben, sollten Sie die Messung in einen numerischen Wert aufgliedern. Die Funktion “Aus String suchen” ist ein nützliches Hilfsmittel zur Umwandlung von ASCII-Zahlen in numerische Werte. Der folgende Code trennt den Textteil “VOLT (Gleichstrom)” in der String-Eingabe ab und wandelt die Angabe “+1.28E+2” in einen

numerischen Wert mit Double-Präzision um. Die String-Eingabe entspricht einem typischen Meßwert eines Mehrfach-Meßgeräts.



Wenn das Gerät binäre Daten ausgibt, können Sie die Typenformungsfunktion verwenden. Diese Funktion ändert den Datentyp einer Verbindung, ohne die Art und Weise, wie die Daten gespeichert werden, zu beeinflussen. Durch “VISA Reads” werden String-Daten ausgegeben; dabei spielt es keine Rolle, ob die Daten im ASCII-Format oder in binärem Format vorliegen. Zum Umwandeln eines binären Strings in einen numerischen Wert oder in ein numerisches Array, müssen Sie den String in einen anderen Datentyp umformen. Im folgenden Beispiel wird eine 4-Byte-Kopfzeile von einem 1000-Byte-Antwortstring abgetrennt, und die verbleibenden Werte werden anschließend in ein Array mit einfacher Genauigkeit umgewandelt.



Treiber mit vollständigen Funktionen erstellen

Wenn Sie einen Treiber erstellen, der von anderen Benutzern eingesetzt wird, sollten Sie in Erwägung ziehen, einen Treiber mit vollständigen Funktionen herzustellen. Diese Treiber sind deutlicher modular organisiert und zeichnen sich durch eine ähnliche Architektur wie die Treiber in der Gerätetreiberbibliothek von National Instruments aus, d.h., sie bieten Funktionen für das Melden von Fehlern sowie Utility-Funktionen. Weitere Detailinformationen zur Entwicklung eines detaillierteren Treibers erhalten Sie in den Anwendungshinweisen 006, *Developing a LabVIEW Instrument Driver (Entwickeln eines LabVIEW-Gerätetreibers)*, die Sie über die Website von National Instruments abrufen können.

LabVIEW mit IVI-Gerätetreibern verwenden

Außer den über 600 LabVIEW-Quellcodetreibern stehen zur Steuerung von Geräten auch IVI-Treiber (Intelligent Virtual Instruments/Intelligente virtuelle Instrumente) zur Verfügung. IVI-Gerätetreiber sind Treiber auf DLL-Basis, die in LabWindows/CVI erstellt werden und Anwendern im Produktionstestbereich die folgenden zusätzlichen Vorteile bieten:

- Gerätezustands-Caching zur Leistungsverbesserung
- Simulation
- Multithread-Sicherheit
- Geräteattributzugriff

Weitere Informationen zu IVI-Gerätetreibern und deren Einsatz erhalten Sie über die *Online-Referenz* von LabVIEW.

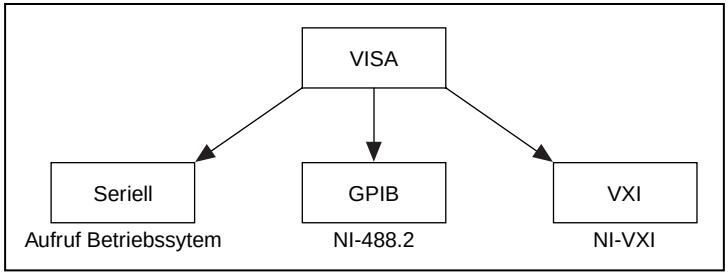
8

Tutorial für LabVIEW VISA

Dieses Kapitel bietet einen Überblick über die Implementierung von VISA in LabVIEW. Es erklärt die grundlegenden Konzepte zur Programmierung von Instrumenten mittels VISA und bietet Ihnen Beispiele, die einfache Konzepte von VISA demonstrieren.

Was ist VISA?

VISA ist ein standardmäßiges I/O Application Programming Interface (API) für die Instrumentationsprogrammierung. VISA selbst bietet keine Fähigkeiten zur Instrumentationsprogrammierung an. VISA ist eine High-Level API, die Treiber auf niedrigeren Ebenen aufruft. Die Hierarchie von NI-VISA ist aus der untenstehenden Abbildung ersichtlich.



VISA kann VXI-, GPIB-, oder serielle Instrumente steuern, wobei je nach verwendetem Instrument die geeigneten Treiberaufrufe ausgegeben werden. Beim Debugging von VISA-Problemen ist es wichtig, auf die Existenz von dieser Hierarchie zu achten. Was dem Anschein nach ein VISA-Problem ist, könnte in Wirklichkeit eine Schwierigkeit mit einem der von VISA aufgerufenen Treibern sein.

Unterstützte Plattformen und Umgebungen

Da VISA der Standard der Branche für die Entwicklung von Instrumenten-Treibern darstellt, benutzen die meisten gegenwärtig von National Instruments geschriebenen Gerätetreiber auch VISA und unterstützen folglich Macintosh, Windows 3.*x*, Windows 95, Windows NT, Solaris 1, Solaris 2 und HP-UX, sofern Treiber auf Systemniveau für die betreffende Plattform verfügbar sind.

Warum VISA benutzen?

VISA ist der Standard

VISA ist die standardmäßige API für Gerätetreiber überall in der Instrumentationsbranche. Darüber hinaus können Sie mit nur einer API eine Reihe von Instrumenten verschiedener Art, einschließlich VXI, GPIB und serielle, steuern.

Schnittstellenunabhängigkeit

VISA verwendet die gleichen Operationen zur Kommunikation mit Instrumenten für jeden Schnittstellentyp. Zum Beispiel: Der VISA-Befehl um einen ASCII-String auf ein meldungsbasiertes Instrument zu schreiben ist der gleiche, egal ob das Instrument eine serielle, GPIB- oder VXI-Schnittstelle benutzt. VISA bietet daher Schnittstellenunabhängigkeit. So wird es leichter, Bus-Schnittstellen zu wechseln, weshalb Benutzer, die Instrumente für verschiedene Schnittstellen programmieren sollen, nur eine API lernen müssen.

Plattformunabhängigkeit

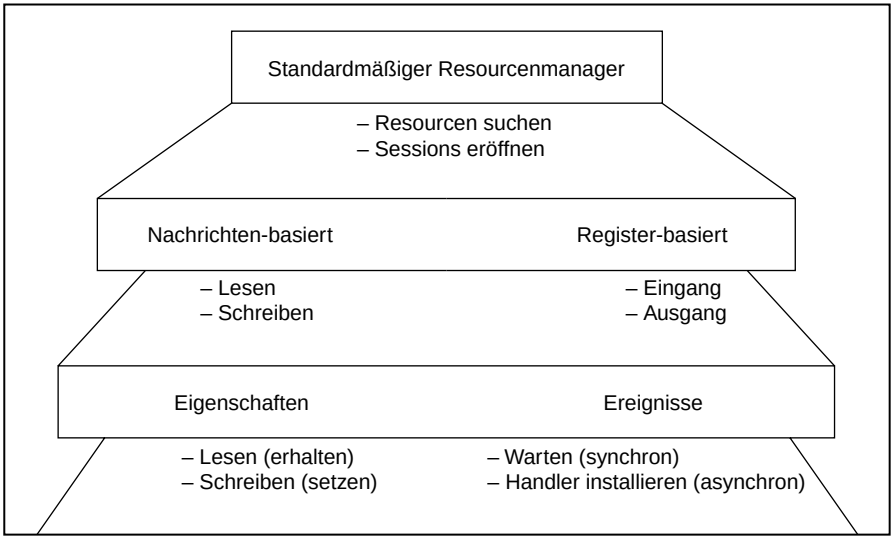
VISA ist derart ausgelegt, daß Programme, die mit VISA-Funktionsaufrufen geschrieben wurden, leicht von einer Plattform zur anderen zu übertragen sind. Um die Plattformunabhängigkeit zu sichern, definiert VISA seine eigenen Datentypen streng. Fragen, wie z.B. die von einer Plattform zur anderen variable Größe eines Integers in Byte, sollten daher ein VISA-Programm nicht beeinflussen. Die VISA-Funktionsaufrufe und die damit verwandten Parameter sind über alle Plattformen hinweg einheitlich. Software kann auf andere Plattformen portiert und anschließend neu kompiliert werden. Ein LabVIEW-Programm kann auf jede Plattform portiert werden, die LabVIEW unterstützt.

Leicht an die Zukunft anzupassen

Ein weiteres Vorteil von VISA ist, daß es eine objektorientierte API ist, die leicht an in der Zukunft neu entwickelte Instrumentationsschnittstellen anzupassen ist, wodurch die Migration von Anwendungen auf neue Schnittstellen erleichtert wird.

Grundkonzepte von VISA

Ein vereinfachter Umriß der internen Struktur der VISA-API ist in dem untenstehenden Diagramm zu sehen.



Standardmäßiger Ressourcenmanager, Session und Instrumenten- Deskriptoren

Der standardmäßige Ressourcenmanager befindet sich auf der höchsten Ebene der VISA-Operationen. LabVIEW baut beim ersten VISA VI-Aufruf automatisch eine Kommunikation mit dem standardmäßigen Ressourcenmanager auf. Hier werden zwei Begriffe erwähnt, die der Definition bedürfen: Ressource und Session.

Ressource Ein Objekt, mit dem Sie kommunizieren können. Beispiele umfassen Instrumenten- (INSTR) und Speicherzugriff-Ressourcen (MEMACC).

Session—Eine Verbindung (Verknüpfung) zu irgendeiner bestehenden Ressource, auch mit dem standardmäßigen Ressourcenmanager.

Sie benutzen den standardmäßigen VISA Ressourcenmanager, um Sessions mit anderen Ressourcen zu eröffnen. Sie müssen Sessions mit den Instrumenten eröffnen, bevor ein VI mit ihnen kommunizieren kann.

Der standardmäßige VISA Ressourcenmanager kann auch nach verfügbaren Ressourcen im System suchen. Sie können dann mit allen diesen Ressourcen Sessions eröffnen.

Wie suche ich nach Ressourcen?

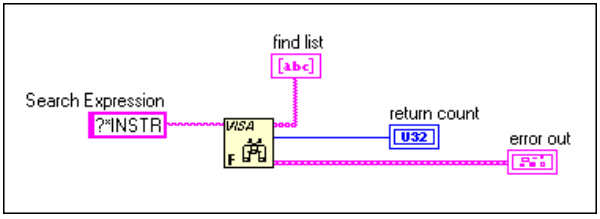
Die unten gezeigte Funktion VISA Find Resource sucht nach verfügbaren Ressourcen im System. Diese Funktion ist ein häufiger Startpunkt für ein VISA-Programm. Sie können sie verwenden, um festzustellen, ob alle nötigen Ressourcen für die auszuführende Anwendung verfügbar sind.

Die einzige nötige Eingabe zur Funktion VISA Find Resource ist ein String mit der Bezeichnung Such-**Ausdruck**. Dieser bestimmt die Art von Ressourcen, die das VI Find Resource zurückgibt. Mögliche Strings für den Such-**Ausdruck** werden in der untenstehenden Tabelle gezeigt und sind auch in der LabVIEW *Online Referenz* zu finden.

Instrumenten-Ressourcen	Ausdruck
GPIB	GPIB[0 – 9]*::? *INSTR
GPIB-VXI	GPIB-VXI?*INSTR
GPIB oder GPIB-VXI	GPIB?*INSTR
VXI	VXI?*INSTR
Alle VXI	?*VXI[0 – 9]*::?*INSTR
Serielle	ASRL[0 – 9]*::?*INSTR
Alle	?*INSTR

Speicherzugriff-Ressourcen	Ausdruck
VXI	VXI?*MEMACC
GPIB-VXI	GPIB-VXI?*MEMACC
Alle VXI	?*VXI[0–9]*::?*MEMACC
Alle	?*MEMACC

Die Rückgabewerte der Funktion sind der **zurückgelieferte Zählwert** (der die Anzahl der gefundenen Ressourcen meldet) und die **Suchliste**. Die **Suchliste** ist ein Array von Strings. Jeder String enthält die Beschreibung von einer der gefundenen Ressourcen. Diese Strings heißen Instrumenten-Deskriptoren. Ein VI, das alle verfügbaren Ressourcen im System findet, ist in der untenstehenden Abbildung dargestellt.



Instrumenten-Deskriptor—Die genaue Bezeichnung und Adresse von einer VISA Ressource. Dieser String hat folgendes Format:

Schnittstellentyp[Kartenindex]::Adresse::VISA-Klasse

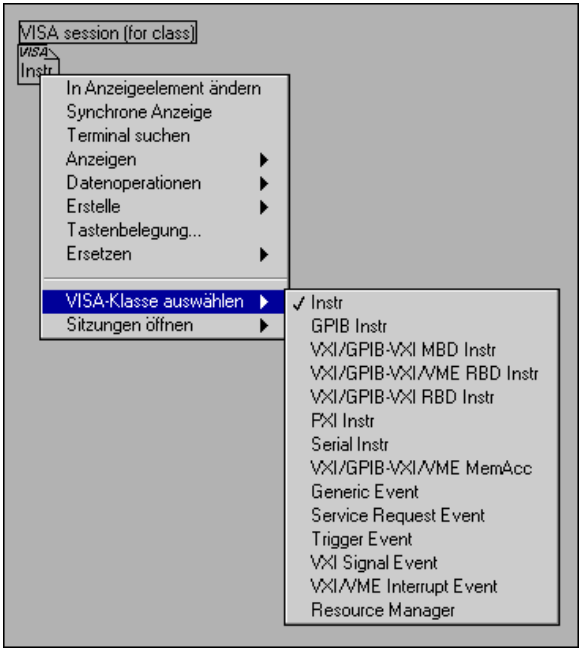
Die Instrumenten-Deskriptoren sind einfach spezifische Instrumente, die durch die Suchabfrage gefunden wurden. Der Kartenindex muß nur benutzt werden, wenn mehr als ein Schnittstellentyp im System vorhanden ist. Wenn das System beispielsweise zwei GPIB-Steckkarten enthält, könnte eine als GPIB0 und eine andere als GPIB1 bezeichnet werden. In diesem Fall muß der Kartenindex in Instrumenten-Deskriptoren benutzt werden. Für VXI-Instrumente ist der Adreßparameter die Logik-Adresse des Instruments. Für GPIB-Instrumente ist er die GPIB-Hauptadresse. Serielle Instrumente benutzen den Adreßparameter nicht. ASRL1::INSTR ist z.B. der Deskriptor des seriellen Anschlusses COM 1 auf einem Personalcomputer.

Was ist eine VISA-Klasse?

Die VISA-Klasse ist eine Gruppierung, die einige oder alle VISA-Operationen verkapselt. INSTR ist die allgemeinste Klasse, die alle VISA-Operationen für ein Instrument umfaßt. In der Zukunft könnten weitere Klassen der VISA-Spezifikation hinzugefügt werden. Zur Zeit muß die VISA-Klasse nicht als Bestandteil des Instrumenten-Deskriptors eingefügt werden, aber Sie sollten sie einfügen, um die zukünftige Kompatibilität zu sichern. Wenn die VISA-Klasse gegenwärtig leergelassen wird, wird sie automatisch in die INSTR-Klasse umgewandelt.

Ein Popup-Menü auf einem VISA-Bedienelement aufrufen

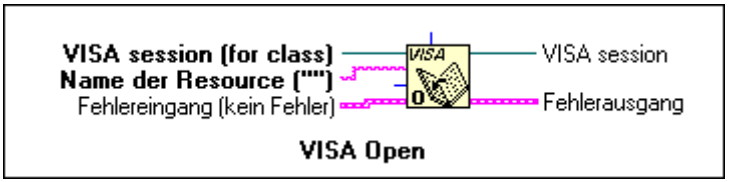
LabVIEW bietet eine andere Möglichkeit, die Klasse für eine VISA Session zu bestimmen, die jetzt benutzt werden kann. Sie können auf dem Visa Session-Bedienelement am Frontpanel ein Popup-Menü aufrufen und die VISA-Klasse auswählen, wie in der folgenden Abbildung gezeigt.



Wird eine andere Klasse als die standardmäßige Instr-Klasse ausgewählt, so können in dieser Session nur Funktionen für mit dieser Geräteklasse assoziierten Operationen erfolgreich verbunden werden. Wenn beispielsweise GPIB Instr als VISA-Klasse ausgewählt wird, können die Funktionen für VXI Registerzugriff in der Session nicht verbunden werden.

Eine Session eröffnen

Die Instrumenten-Deskriptoren dienen zur Eröffnung von Sessions zu den Ressourcen im System. Die Funktion VISA Open ist unten dargestellt.



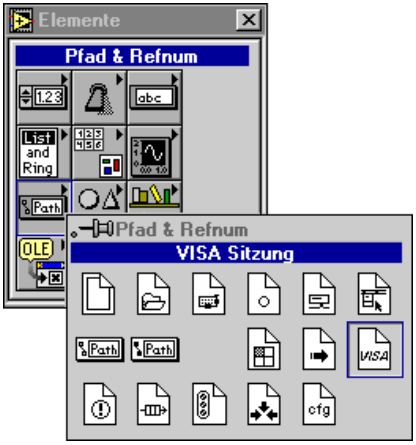
Der Eingabenamen der Ressource ist der VISA Instrumenten-Deskriptor für die Ressource, zu der eine Session eröffnet wird.

☞ **Hinweis** ***Um Instrumenten-Deskriptoren für Ressourcen zu finden, ist es nicht nötig, das VI Find Resource zu benutzen. Das VI VISA Open kann mit einem vom Anwender oder vom Programmierer eingegebenen Instrumenten-Deskriptor verwendet werden. Um der Syntax für den Deskriptor sicher zu sein, empfiehlt es sich jedoch, zuerst Find Resource auszuführen.***

☞ **Hinweis** ***In den meisten Anwendungen brauchen Sessions nur einmal für jedes Instrument, mit dem die Anwendung kommunizieren wird, eröffnet zu werden. Diese Session kann durch die Anwendung gerichtet und dann am Ende der Anwendung geschlossen werden.***

Bemerken Sie auch, daß es einen VISA Sessionseingang zum VI VISA Open gibt. Um Sessions zu Ressourcen in LabVIEW zu eröffnen, ist ein Frontpanel-Bedienelement für die VISA Session erforderlich.

Das Frontpanel-Bedienelement für die VISA Session ist in der Palette **Elemente** in der Subpalette **Pfad & Refnum** zu finden.



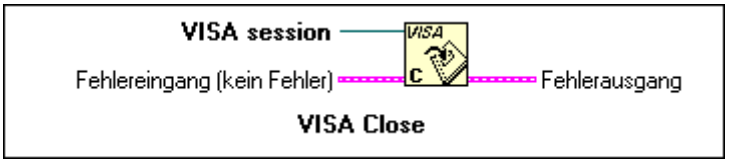
Was ist die Beziehung zwischen dem standardmäßigen Ressourcenmanager, den Instrumenten-Deskriptoren und den Sessions?

Hier ist es wichtig, ein klares Verständnis des standardmäßigen Ressourcenmanagers, der Instrumenten-Deskriptoren, und der Sessions zu besitzen. Es läßt sich eine gute Analogie zwischen dem standardmäßigen VISA Ressourcenmanager und einem Telefonbeamten ziehen. Die Eröffnung einer Session zum standardmäßigen Ressourcenmanager (was in LabVIEW automatisch geschieht) ist wie das Abheben eines Hörers und der Anruf zum Telefonbeamten, um eine Kommunikationsverbindung zwischen einem Programm und dem VISA-Treiber aufzubauen.

Der Telefonist kann seinerseits eine Rufnummer wählen, um Kommunikationsverbindungen mit Ressourcen im System aufzubauen. Die vom Ressourcenmanager verwendeten Nummern sind die Instrumenten-Deskriptoren. Die Kommunikationsverbindungen sind die Sessions, die zu VISA Ressourcen eröffnet werden. Der Ressourcenmanager kann ferner nach allen verfügbaren Rufnummern suchen. Dies ist die Operation VISA Find Resource.

Eine Session schließen

Eine offene Session zu einer VISA Resource verwendet auch Systemressourcen im Computer. Um ein VISA-Programm richtig zu beenden, sollten alle eröffnete VISA Sessions geschlossen werden. Zu diesem Zweck gibt es das VI VISA Close, das unten gezeigt ist.



Die Eingabe VISA Session zum VI VISA Close ist die zu schließende Session. Diese Session kommt ursprünglich aus der Terminal Output Session des VIs VISA Open.

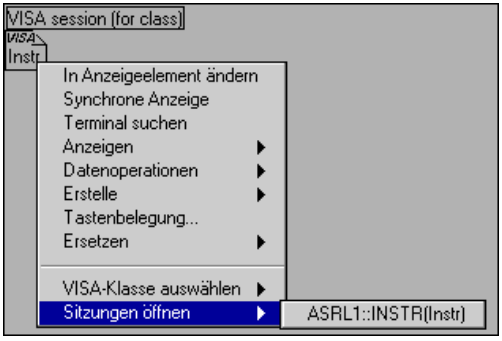
Wird eine Session bei der Ausführung von einem VI nicht geschlossen, so bleibt sie offen. Es ist ratsam, Sessions, die in einer Anwendung eröffnet werden, zu schließen, damit offene Sessions sich nicht aufbauen und Probleme mit Systemressourcen verursachen. Es gibt aber auch Fälle, in denen es nützlich sein kann, Sessions offen zu lassen.

☞ **Hinweis** ***Wenn ein VI abgebrochen wird (wenn Sie beispielsweise ein VI testen) wird die VISA-Session nicht automatisch geschlossen. Sie können das VI Open VISA Session Monitor (in*** `vi.lib/Utility` ***zu finden) zur Hilfe beim Schließen solcher Sessions ziehen.***

Wann empfiehlt es sich, eine Session offenzulassen?

Wenn ein VI ausgeführt wird und eine Session offenläßt, ohne sie abzuschließen, kann diese Session in nachfolgenden VIs benutzt werden. Auf eine offene Session kann durch das Aufrufen einer VISA-Session Frontpanel-Bedienelement durch Wählen von **Sitzungen öffnen** zugegriffen werden. Die Ausgabe des VISA-Session Frontpanel-Bedienelements wird dann die ausgewählte offene Session sein. Auf dieser Weise können Sessions, die von früheren Ausführungen von einem VI offengelassen wurden, geschlossen werden. Diese Methode kann auch interaktiv benutzt werden, um Teile einer Anwendung zu überprüfen.

Ein Beispiel von der Auswahl einer offenen Session ist in der folgenden Abbildung dargestellt.



Sie können das VISA Session Bedienelement verwenden, um nach offenen Sessions zu erkundigen. Zugriff auf offene Sessions durch Aufrufen von VISA Session Frontpanel-Bedienelemente bietet außerdem eine bequeme Möglichkeit, Teile von einem Gerätetreiber interaktiv auszuführen.

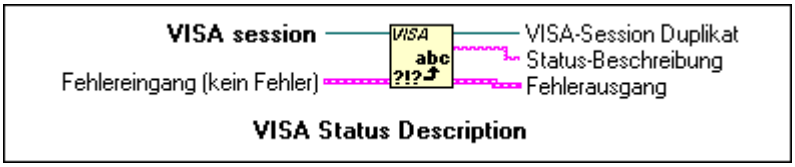
Die Fehlerbehandlung mit VISA

Die Fehlerbehandlung mit VISA VI ähnelt der Fehlerbehandlung mit anderen I/O VIs in LabVIEW. Jedes VISA VI enthält Fehlereingangs- und Fehlerausgangsterminals, die zur Weitergabe von Fehler-Clustern von einem VI zum anderen innerhalb eines Diagramms dienen. Der Fehler-Cluster enthält einen Booleschen Flag zur Angabe, daß ein Fehler eingetreten ist, einen numerischen VISA-Fehlercode und einen String mit der Adresse des VIs, wo der Fehler eingetreten ist. Wenn ein Fehler eintritt, versuchen spätere VIs gar nicht erst abzulaufen, sondern geben den Fehler-Cluster einfach weiter. Eine Frontpanelanzeige für Fehler-Cluster, die die Ausgabe vom Fehlerausgangsterminal des VISA VIs anzeigt, ist in der folgenden Abbildung dargestellt.

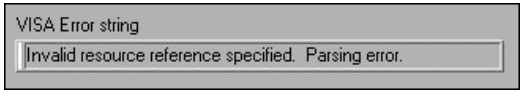
Bemerken Sie, daß in diesem Fall ein Fehler eingetreten ist. Bemerken Sie gleichfalls, daß der VISA-Fehlercode in der Code-Anzeige abgeschnitten ist. VISA-Fehlercodes sind 32-Bit-Integer, üblicherweise im Hexadezimalformat. Der LabVIEW Fehler-Cluster zeigt den Code im Dezimalformat an. Das Thema *VISA-Fehlercodes* in der LabVIEW *Online-Referenz* und der Abschnitt *Numerische Fehlercodes* im Anhang A, *Fehlercodes*, im *LabVIEW Funktionen- und VI-Referenzhandbuch* geben die Fehlercodes auch im Dezimalformat an. Wie aus der obigen Abbildung aber hervorgeht, sind diese Fehlercodes im Fehler-Cluster abgeschnitten. Die Größe der Code-Anzeige muß also geändert werden, um den gesamten Fehlercode anzuzeigen.

Die LabVIEW VIs Simple und General Error Handler sind in der Subpalette **Zeit & Dialog** unter der Palette **Funktionen** zu finden. Wenn ein Fehler eintritt, stellen diese VIs ein kontextsensitives Dialogfeld zur Verfügung, das mögliche Gründe für den Fehler anzeigt. Das VI Einfacher Fehlerbehandler zeigt denselben Fehler wie das im vorigen Fehler-Cluster verwendete Beispiel, liefert aber detailliertere Information über den Fehler, wie aus der folgenden Abbildung hervorgeht.

Bemerken Sie, das die Codebeschreibung unter den möglichen Gründen steht. Es ist nicht immer bequem, Fehler mit kontextsensitiven Dialogfeldern mittels der Fehlerbehandlungs-VIs von LabVIEW zu behandeln. VISA stellt außerdem eine Operation zur Verfügung, die einen VISA-Fehlercode nimmt und einen dem Code entsprechenden Fehlermeldungs-String ausgibt. Dieses VI ist in der untenstehenden Abbildung dargestellt.



Die Eingänge zu diesem VI sind eine VISA Session und einen VISA Fehler-Cluster. Das VI überprüft den VISA-Code in dem eingegebenen Fehler-Cluster und gibt die textuelle Beschreibung des Codes in **Status Description** aus. Die folgende Abbildung zeigt eine LabVIEW String-Anzeige des vom VI VISA Status Description zurückgegebenen Fehlerstrings.



Die Methode zur Implementierung der Fehlerbehandlung hängt im Einzelfall vom jeweiligen Programm ab. Irgendeine Art von Fehlerbehandlung sollte jedoch in jedem Programm, das mit VISA etwas zu tun hat, implementiert werden.

Easy VISA VI

Sie können die VIs Easy VISA verwenden, um nachzuweisen, daß Sie mit Ihrem Instrument die Kommunikation aufgenommen haben. Beim Entwickeln von Ihren Anwendungen aber sollten Sie die anderen VISA VIs in der Palette verwenden, weil sie mehr Kontrolle über Ihrem Instrument anbieten. Für weitere Information über die Easy VISA VIs sehen Sie bitte den Abschnitt *Kommunikation mit dem Gerät testen* in Kapitel 7, *Erste Schritte mit einem LabVIEW-Gerätetreiber*. Die Beispiele in den folgenden Abschnitten machen keinen Gebrauch von Easy VISA VIs.

Meldungsbasierte Kommunikation

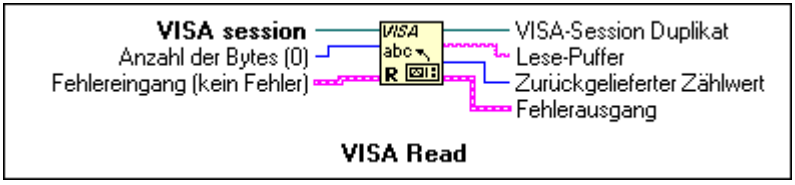
Serielle, GPIB-, und viele VXI-Geräte erkennen eine Vielfalt von meldungsbasierten Befehlsstrings. Das eigentliche Protokoll, das zur Sendung von einem String an ein Instrument verwendet wird, ist auf VISA-Ebene transparent. Ein Benutzer braucht nur zu wissen, daß er eine Meldung an ein meldungsbasiertes Gerät schreiben bzw. von einem meldungsbasiertem Gerät lesen möchte. Die zur Ausführung dieser Operationen eingesetzten VIs heißen VISA Write und VISA Read.

☞ **Hinweis** ***Dieselben VIs dienen dem Schreiben von meldungsbasierten Befehlen an GPIB-, serielle und meldungsbasierte VXI-Instrumente. VISA weiß aufgrund der verwendeten Art der Ressource automatisch, welche Treiberfunktionen aufzurufen sind.***

Das VI VISA Write ist unten dargestellt.

Die einzige Eingabe, abgesehen von der Session, ist der an das Instrument zu sendende String.

Das VI VISA Read ist ebenso leicht zu benutzen. Es ist unten dargestellt.

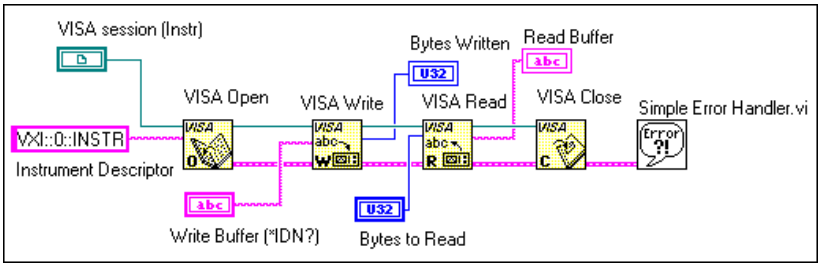


Dem VI VISA Read muß eine Byteanzahl gleich der Höchstanzahl der vom Instrument abzulesenden Bytes eingegeben werden. Das VI hört zu lesen auf, sobald die Anzahl der Bytes gelesen worden ist oder das Ende der Übertragung angezeigt wird.

Die spezifischen meldungsbasierten Befehle, die das Instrument erkennt, unterscheiden sich von Hersteller zu Hersteller. IEEE 488.2 und SCPI standardisierten die Befehle für meldungsbasierte Instrumente. Viele Instrumente folgen diesen Standards. Um sich der Befehle für ein bestimmtes Instrument sicher zu sein, liest man in der Dokumentation der Hersteller nach. Instrumententreiber existieren jedoch für viele meldungsbasierte Geräte. Diese Instrumententreiber enthalten Funktionen, die die entsprechenden ASCII Befehlsstrings zusammenstellen und ans Instrument senden. Um die neuesten Treiber zu bekommen, wenden Sie sich an die Webseite oder den ftp-Dienst von National Instruments.

Wie schreibe ich an ein meldungsbasiertes Gerät und wie lese ich davon ab?

Ein einfaches Beispiel, das den *IDN? (Identifikation)-String an ein meldungsbasiertes Instrument schreibt und die Antwort abliest, ist in der untenstehenden Abbildung dargestellt.

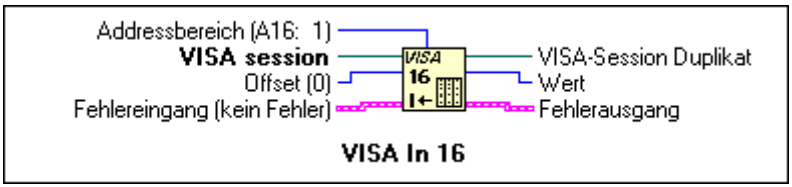


Dieses Programm könnte erfolgreich mit jedem Instrument benutzt werden, das den *IDN? Befehl erkennt. Das Gerät könnte seriell, GPIB oder ein meldungsbasiertes VXI sein. Die einzige Änderung wäre der Instrumenten-Deskriptor.

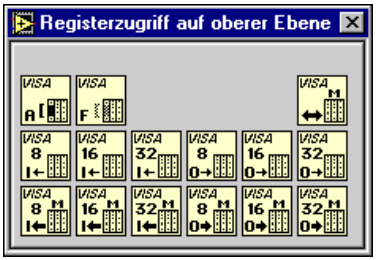
Registerbasierte Kommunikation (nur VXI)

VISA enthält eine Reihe von Registerzugriff-VIs zur Verwendung mit VXI-Instrumenten. Wenn Sie ausschließlich GPIB- oder serielle Geräte verwenden, ist dieser Abschnitt unzutreffend.

Einige VXI-Instrumente unterstützen meldungsbasierte Befehle nicht. Mit derartigen Instrumenten kann die Kommunikation nur durch Registerzugriff erfolgen. Alle VXI-Instrumente besitzen Konfigurationsregister in den obigen 16 Kilobyte vom Speicherbereich A16. Daher können Registerzugriffsfunktionen auch bei meldungsbasierten Geräten verwendet werden, um an die Konfigurationsregister zu schreiben oder davon abzulesen. Die grundlegende VISA-Operation zum Ablesen von einem Wert aus einem Register ist VISA In. Eigentlich gibt es drei verschiedene Versionen von dieser Operation, jeweils zum Einlesen eines 8-, 16- oder 32-Bit-Wertes. Sie müssen das VI für die von ihrem Instrument unterstützte Zugriffsbreite verwenden. Zum Beispiel: Ein bestimmtes Instrument könnte bei 32-Bit-Zugriff einen Busfehler zurückgeben, wenn es auf 16-Bit-Zugriff ausgelegt ist. Das VI VISA In 16 ist in der folgenden Abbildung dargestellt.



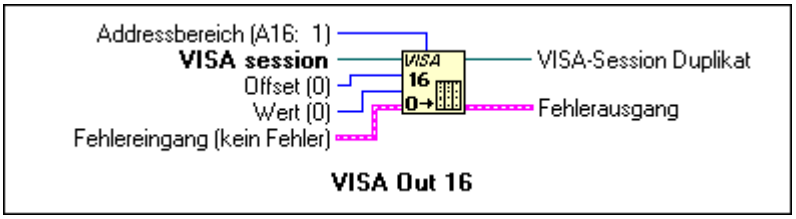
Dieses VI und die anderen grundlegenden Registerzugriff-VIs sind in der Subpalette **High Level Registerzugriff** unter der VISA Hauptfunktionspalette zu finden.



Die Adreßbereich-Eingabe gibt an, welcher VXI-Adreßbereich zu benutzen ist. Die Offset-Eingabe gibt manchmal Anlaß zu Verwirrung. Erinnern Sie sich daran, daß VISA die Basisadresse festhält, die ein Gerät in jedem Adreßbereich verlangt. Die Offset-Eingabe ist relativ zu dieser Basisadresse.

Betrachten Sie folgendes Beispiel. Nehmen Sie an, Sie haben ein Gerät auf Logik-Adresse 1 und möchten das VI VISA In 16 verwenden, um sein Konfigurationsregister der ID/Logik-Adresse zu lesen. Sie wissen, daß dieses Register auf der absoluten Adresse 0xC040 im Bereich A16 liegt und daß die Konfigurationsregister für das Gerät auf Logik-Adresse 1 von 0xC040 bis 0xC07F liegen. VISA weiß dies aber auch und deshalb brauchen Sie in diesem Bereich nur das Offset, auf das Sie zugreifen wollen, anzugeben. In diesem Fall ist das Offset Null.

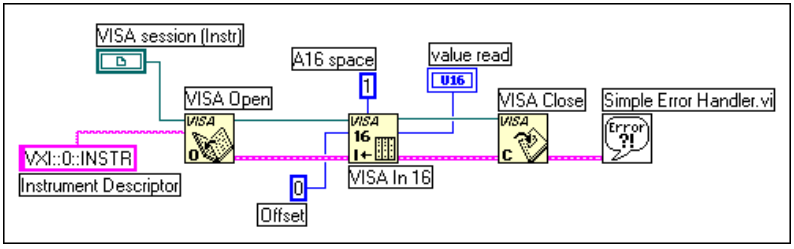
Es gibt einen anderen Satz von Operationen für den High-Level Registerzugriff, die parallel zu den Operationen VISA In liegen, die jedoch für das Schreiben auf Register bestimmt sind. Diese Operationen sind die Operationen VISA Out Operationen. Das VI VISA Out 16 ist unten dargestellt.



Dieses VI ähnelt dem VI VISA In 16 mit der Ausnahme, daß der zu schreibende Wert dem Terminal zur Verfügung gestellt werden muß. Beachten Sie bei dem Einsatz von VISA Out-VIs, daß einige Register auf einen Schreibzyklus möglicherweise nicht ansprechen oder einen Busfehler verursachen.

Einfacher Registerzugriff

Ein Beispiel für die Verwendung von High-Level VISA-Zugriffsfunktionen in einem VI ist im folgenden einfachen Programm ersichtlich.

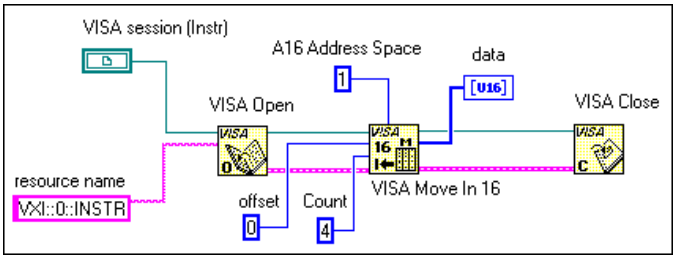


Dieses Blockdiagramm zeigt, wie das VI VISA In 16 verwendet wird, um das erste Konfigurationsregister für ein VXI-Gerät auf Logik-Adresse 0 zu lesen. Der Offsetparameter, Null im vorliegenden Fall, ist relativ zum vom Gerät verlangten Speicherbereich im VXI-Adreßbereich, auf das zugegriffen wird. Der Adreßbereich-Parameter zeigt an, auf welchen VXI-Adreßbereich zugegriffen wird. In diesem Fall liegt das Gerät auf der Logik-Adresse 0. Seine Konfigurationsregister reichen von 0xC000 bis 0xC03F im Adreßbereich A16. Die VISA In 16 Operation mit Offset 0 liest in Wirklichkeit das Register auf 0xC000.

Das Programm liest aus einem 16-bit-Register im Adreßbereich A16 auf dem angegebenen Offset (0) der angegebenen Ressource (VXI::0::INSTR). Wenn ein Fehler in der Folge von den im Programm ausgeführten VISA-VI eintritt, gibt der Simple Error Handler ein Dialogfeld zurück, in dem der Benutzer über den Fehler informiert wird und die mit dem VISA-Fehlercode assoziierte Textmeldung zurückgegeben wird.

Einfache Registerversetzung

Das folgende Blockdiagramm veranschaulicht, wie das VI VISA Move In 16 zum Lesen der ersten vier Konfigurationsregister für ein VXI-Gerät auf Logik-Adresse 0 verwendet wird.



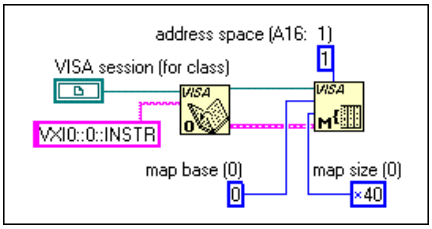
Die VIs VISA Move In dienen zum Lesen von großen Datenblocks aus VXI-Geräten. Die Daten werden als ein Array von vier 16-Bit-Werten zurückgegeben. Ein entsprechender Satz von VIs VISA Move Out existiert zum Übertragen von großen Datenblöcken an VXI-Geräte. Die VIs Move In und Move Out besitzen 8-, 16- und 32-Bit-Versionen. Das geeignete VI ist durch die Größe der Register, auf die zugegriffen werden soll, bestimmt.

Low-Level Access Funktionen

Low-Level Access (LLA) Funktionen bieten eine sehr effiziente Art an, registerbasierte Kommunikation durchzuführen. LLA-Funktionen sind für bestimmte Zugriffsarten mit viel weniger Overhead behaftet als High-Level Access (HLA) Funktionen. LLA- und HLA-Funktionen führen dieselben Schritte aus, abgesehen davon, daß jede einzelne Aufgabe, die durch eine HLA-Funktion ausgeführt wird, eine Einzelfunktion unter LLA ist.

Die Verwendung von VISA für Low-Level Registerzugriffe

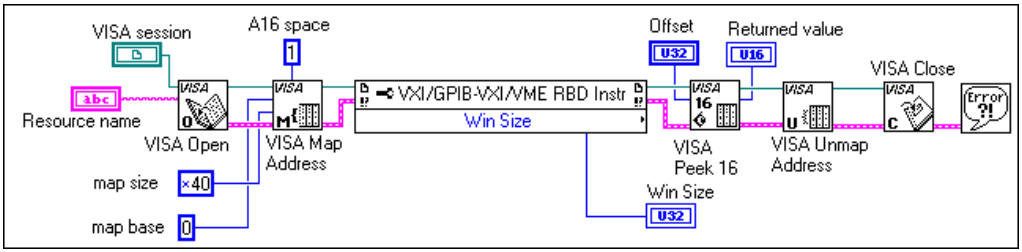
Die erste LLA-Operation, die man aufrufen muß, um auf ein Geräteregister zuzugreifen, ist die Operation `VISA Map Address`, die das Hardware-Fenster einrichtet, um Zugriff auf den VXI-Adreßbereich zu erlauben. Die Operation `VISA Map Address` programmiert die Hardware so, daß lokale CPU-Adressen auf VXI-Adressen abgebildet werden, wie im vorigen Abschnitt beschrieben. Der folgende Code ist ein Beispiel davon, wie die Hardware zum Zugriff auf den Bereich A16 programmiert wird.



Dieser einfache Code richtet die Hardware zur Abbildung vom Bereich A16 für 0x40 Byte, ausgehend vom Offset 0, ein. Erinnern Sie sich daran, daß das Offset relativ zur Basisadresse des angesprochenen Offsets ist und nicht zur Basis vom Bereich A16 selbst. Offset 0 bedeutet also nicht die Adresse 0 im Bereich A16, sondern den Ausgangspunkt des A16-Speichers im Gerät.

☞ **Hinweis** ***Um auf Geräteregister über eine MEMACC-Session zuzugreifen, müssen Sie die absolute VXI-Bus-Adresse (Basisadresse für Gerät + Registeroffset im Gerät-Adreßbereich) angeben.***

Wenn Sie mehr als eine einzige Zuordnung für ein Gerät brauchen, müssen Sie eine zweite Session zum Gerät eröffnen, weil VISA gegenwärtig nur eine einzige Zuordnung pro Session unterstützt. Zwei Sessions zu unterhalten ist mit nur wenig Overhead verbunden, da Sessions selbst nicht viel Speicher brauchen. Sie müssen jedoch zwei Session-Handles verwalten. Beachten Sie, daß dies sich von der Höchstanzahl von Fenstern, die man auf einem System haben darf, unterscheidet. Die Hardware für den von Ihnen benutzten Controller kann eine Begrenzung in der Anzahl der verschiedenen unterstützbaren Fenstern haben. Wenn Sie mit einem Fenster fertig sind oder die Zuordnung auf eine andere Adresse oder einen anderen Adreßbereich wechseln wollen, müssen Sie die Zuordnung des Fensters unter Anwendung der Operation `VISA Unmap Address` freigeben.



Dieses Beispiel ordnet dem Adreßbereich A16 64 Byte (hex 40), ab Offset 0 von der Adresse des Geräts auf Logik-Adresse 1, zu. Beim Gebrauch von LLA-Operationen, geben Sie stets eine Zuordnungsgröße an, die groß genug ist, um den Adreßbereich aufzunehmen, auf den Sie zugreifen werden. Dies können Sie tun, indem Sie einen Eigenschaftsknoten benutzen, um die Menge an Speicherplatz zu bestimmen, der vom Gerät benutzt wird. Eigenschaftsknoten werden später in diesem Kapitel erklärt.

Hinweis ***Das standardmäßige Maximalfenster, das zugeordnet werden kann, ist typischerweise 64kB. Bei Verwendung von einem MITE-basierten Controller können Sie mehr als 64kB anfordern, aber dann müssen Sie die Größe von Ihrem Anwenderfenster erhöhen. Dies erfolgt im Ressource-Editor für Ihren Controller (entweder T&M Explorer, VXIEdit oder VXItedit). Konsultieren Sie bitte die mit Ihrem Controller mitgelieferte Dokumentation.***

Busfehler

Busfehler werden von LLA-Operationen nicht gemeldet. `VISA Peek` und `VISA Poke` melden keine Fehlerzustände. Die HLA-Operationen dagegen melden Busfehler. Wenn Sie LLA-Operationen benutzen, müssen Sie sicherstellen, daß die Adressen, auf die Sie zugreifen, gültig sind.

Vergleich von High-Level- und Low-Level-Zugriff

Geschwindigkeit

In Bezug auf die Entwicklungsgeschwindigkeit für Ihre Anwendung sind die HLA-Operationen wegen der einfacheren Schnittstelle und der nach jedem Zugriff erhaltenen Information viel schneller zu implementieren und zu testen. Zum Beispiel: HLA-Operationen verkapseln die Zuordnung und Freigabe von Hardware-Fenstern, wodurch Sie `VISA Map Address` and `VISA Unmap Address` nicht einzeln aufrufen müssen.

Hinsichtlich der Ausführungsgeschwindigkeit laufen die LLA-Operationen schneller, wenn sie für mehrere I/O-Direktzugriffe auf Register in einem einzelnen zugeordneten Fenster verwendet werden. Wenn Sie wissen, daß die nächsten paar Zugriffe in einem einzelnen Fenster stattfinden, können Sie die Zuordnung nur einmal durchführen und dann hat jeder Zugriff minimales Overhead.

Die HLA-Operation werden langsamer sein, weil sie bei jedem Aufruf eine Zuordnung, einen Zugriff und eine Freigabe durchführen müssen. Selbst wenn das Fenster für den Zugriff korrekt zugeordnet worden ist, muß der HLA-Aufruf zumindest irgendeine Überprüfung durchführen um zu bestimmen, ob er neuzuordnen muß. Weil HLA-Operationen außerdem viele Fähigkeiten zur Zustandsüberprüfung verkapseln, haben HLA-Operationen ein höheres Software-Overhead. Deshalb ist HLA in vielen Fällen langsamer als LLA.

☞ **Hinweis** ***Für Blockübertragungen sind die High-Level VISA Move Operationen am schnellsten.***

Leichter Gebrauch

HLA-Operationen sind leichter zu gebrauchen, weil sie viele in den LLA-Operationen nicht enthaltenen Fähigkeiten zur Zustandsüberprüfung verkapseln, was das höhere Software-Overhead und die niedrigere Ausführungsgeschwindigkeit von HLA-Operationen erklärt. HLA-Operationen verkapseln auch die Zuordnung und Freigabe von Hardware-Fenstern, was bedeutet, daß Sie `VISA Map Address` und `VISA Unmap Address` nicht getrennt aufrufen müssen.

Zugriff auf mehrere Adreßbereiche

Sie können LLA-Operationen nur zum Zugriff auf den Adreßbereich, der gegenwärtig mit einer einzigen VISA-Session zugeordnet ist, verwenden. Um auf einen anderen Adreßbereich für eine einzige Session zuzugreifen, müssen Sie eine Neuzuordnung durchführen, was den Aufruf von `VISA Unmap Address` und `VISA Map Address` beinhaltet. Daher wird die LLA-Programmierung beim gleichzeitigen Zugriff auf mehrere Adressbereichen gleichzeitig komplexer.

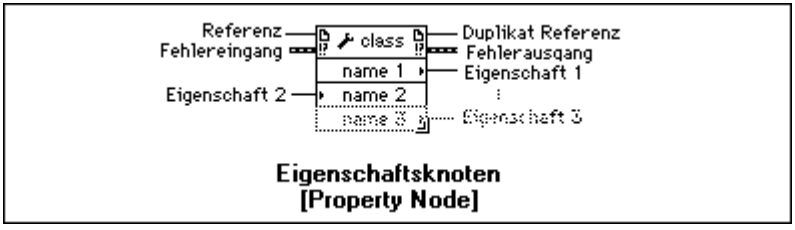
Wenn Sie außerdem mehrere Sessions zum selben Gerät oder zu verschiedenen Geräten haben, die alle Register I/O durchführen, müssen diese um die begrenzte Anzahl von verfügbaren Hardware-Fenstern konkurrieren. Bei Verwendung von LLA-Operationen müssen Sie die Fenster zuweisen und stets sicherstellen, daß das Programm nicht mehr als die verfügbare Anzahl von Fenstern anfordert. Die HLA-Operationen

vermeiden dieses Problem, indem sie das Fenster auf die vorige Einstellung zurücksetzen, sobald sie fertig sind. Selbst wenn alle Fenster gegenwärtig durch LLA-Operationen im Gebrauch sind, kann man noch immer HLA-Funktionen benutzen, da diese den Zustand des Fensters speichern, es neuzuordnen, darauf zugreifen und anschließend das Fenster wiederherstellen. Folglich ist man unter Benutzung von HLA-Operationen nicht eingeschränkt.

VISA-Eigenschaften

Die grundlegenden Operationen, die mit den meldungs- und registerbasierten Ressourcen in VISA assoziiert werden, sind nun eingeführt worden. Diese Operationen erlauben den Registerzugriff und die meldungsbasierte Kommunikation. Neben den grundlegenden Kommunikationsoperationen haben VISA Ressourcen eine Vielfalt von Eigenschaften (Attributen) mit Werten, die in einem Programm abgelesen oder eingestellt werden können.

In einem LabVIEW-Programm werden diese Eigenschaften programmatisch in derselben Weise wie die Eigenschaften der Frontpanel-Bedienelemente behandelt. Eigenschaftsknoten werden zum Lesen oder Einstellen der Werte von VISA-Eigenschaften eingesetzt. Der Eigenschaftsknoten ist in der folgenden Abbildung dargestellt.



Eigenschaftsknoten [Property Node]

☞ **Hinweis** ***Der Eigenschaftsknoten ist ein generischer Knoten, der auch zur Einstellung von ActiveX/OLE- und VI Server-Eigenschaften verwendet werden kann.***

Nach der Plazierung von einem Eigenschaftsknoten auf ein Blockdiagramm können Sie die Eigenschaften für eine VISA-Klasse einstellen. Hierzu können Sie eine von den folgenden Methoden benutzen.

- Verbinden Sie eine VISA-Session mit dem Referenzeingangsterminal des Eigenschaftsknoten.
- Rufen Sie ein kontextsensitives Menü auf dem Eigenschaftsknoten auf, und wählen Sie **Instr** vom Menü **VISA-Klasse auswählen**.

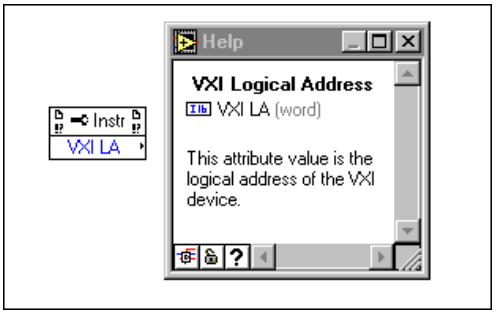
Der Eigenschaftsknoten enthält ein einziges Eigenschaftsterminal wenn er zuerst auf das Blockdiagramm plaziert wird. Seine Größe kann aber geändert werden, damit er soviele Terminals wie nötig enthält. Das anfängliche Terminal am VISA-Eigenschaftsknoten ist ein Leseterminal. Das bedeutet, daß der Wert der in diesem Terminal ausgewählten Eigenschaft gelesen wird. Dies wird durch einen kleinen, nach rechts weisenden Pfeil am rechten Terminalrand angezeigt. Viele Terminals können einzeln von einem Leseterminal zu einem Schreibterminal geändert werden, indem Sie ein kontextsensitives Menü auf der zu ändernden Eigenschaft aufrufen.

☞ **Hinweis** ***Einige Eigenschaften sind schreibgeschützt. Ihre Werte sind nicht einstellbar.***

Um die Eigenschaft in jedem Terminal des Eigenschaftsknoten auszuwählen, rufen sie ein kontextsensitives Menü auf dem Eigenschaftsknotenterminal auf, und wählen Sie **Objekt wählen** aus. Dies ergibt eine Liste von allen möglichen Eigenschaften, die in diesem Programm einzustellen sind. Die Anzahl der verschiedenen Eigenschaften, die unter Objekt wählen vom VISA-Eigenschaftsknoten gezeigt werden, kann durch Änderung der VISA-Klasse des Eigenschaftsknoten geändert werden.

Um die VISA-Klasse zu ändern, rufen Sie ein kontextsensitives Menü auf dem VISA-Eigenschaftsknoten auf, und wählen Sie **VISA-Klasse**. Mehrere verschiedene Klassen neben der standardmäßigen INSTR-Klasse, die alle möglichen VISA-Eigenschaften umfaßt, können unter dieser Option ausgewählt werden. Diese Klassen beschränken die angezeigten Eigenschaften auf jene, die sich auf die ausgewählte Klasse beziehen, statt aller VISA-Eigenschaften. Sobald eine Session mit dem **Session-Eingangs**terminal des Eigenschaftsknoten verbunden ist, wird die VISA-Klasse auf die mit dieser Session assoziierte Klasse eingestellt.

Die VISA-Eigenschaften werden zunächst etwas unvertraut sein und ihre genaue Beschaffenheit wird allein vom Namen her nicht klar sein. Die LabVIEW *Online Referenz* enthält Information über die Eigenschaften. Kurzbeschreibungen von einzelnen Eigenschaften sind auch im einfachen Hilfefenster erhältlich. Um eine Kurzbeschreibung von einer spezifischen Eigenschaft zu erhalten, wählen Sie die Eigenschaft in einem der Terminals von einem Eigenschaftsknoten aus, und öffnen Sie dann das Hilfefenster. Dieses ist unten für die Eigenschaft VXI LA abgebildet.



Beachten Sie, daß das Hilfefenster den spezifischen Variable-Typ anzeigt und eine Kurzbeschreibung der Tätigkeit von dieser Eigenschaft angibt. Falls es unklar ist, welcher Variable-Typ zum Lesen oder Schreiben einer Eigenschaft zu gebrauchen ist, erinnern Sie sich daran, daß der Aufruf von einem kontextsensitivem Menü auf einem Eigenschaftsknoten und die Auswahl von **Konstante erzeugen**, **Bedienelement erzeugen** oder **Anzeigeelement erzeugen** den geeigneten Variable-Typ automatisch auswählen.

Es gibt zwei grundlegende Typen von VISA-Eigenschaften—globale Eigenschaften und lokale Eigenschaften. Globale Eigenschaften sind spezifisch für eine Ressource, während lokale Eigenschaften für eine Session spezifisch sind. Die Eigenschaft VXI LA ist ein Beispiel von einer globalen Eigenschaft. Sie bezieht sich auf alle zu dieser Ressource offenen Sessions. Eine lokale Eigenschaft ist eine Eigenschaft, die für einzelne Sessions zu einer spezifischen Ressource verschieden sein kann. Ein Beispiel einer lokalen Eigenschaft ist der Timeout-Wert. Einige der üblichen Eigenschaften für den jeweiligen Ressource-Typ werden in den untenstehenden Listen angegeben.

Seriell

Serielle Baudrate—Die Baudrate für den seriellen Anschluß.

Serielle Datenbits—Die für serielle Übertragungen verwendete Anzahl von Datenbits.

Serielle Parität—Die für serielle Übertragungen verwendete Parität.

Serielle Stoppbits—Die für serielle Übertragungen verwendete Anzahl von Stoppbits.

GPIB

GPIB-Readressing—Gibt an, ob das Gerät vor jeder Schreiboperation wieder adressiert werden soll.

GPIB-Unadressing—Gibt an, ob das Gerät nach Lese- und Schreiboperation de-adressiert werden soll.

VXI

Mainframe Logical Address—Die niedrigste Logik-Adresse eines Geräts im selben Chassis mit der Ressource.

Manufacturer Identification—Die Hersteller-ID-Nummer von den Konfigurationsregistern des Geräts.

Typcode—Der Typcode von den Konfigurationsregistern des Geräts.

Slot—Der Slot im Chassis, in dem das Gerät sitzt.

VXI Logik-Adresse—Die Logik-Adresse des Geräts.

VXI Speicher-Adreßbereich—Der von der Ressource verwendete VXI-Adreßbereich.

VXI Speicher-Adreßbasis—Die Basisadresse des von der Ressource verwendeten Speicherbereichs.

VXI Speicher-Adreßgröße—Die Größe des von der Ressource verwendeten Speicherbereichs.

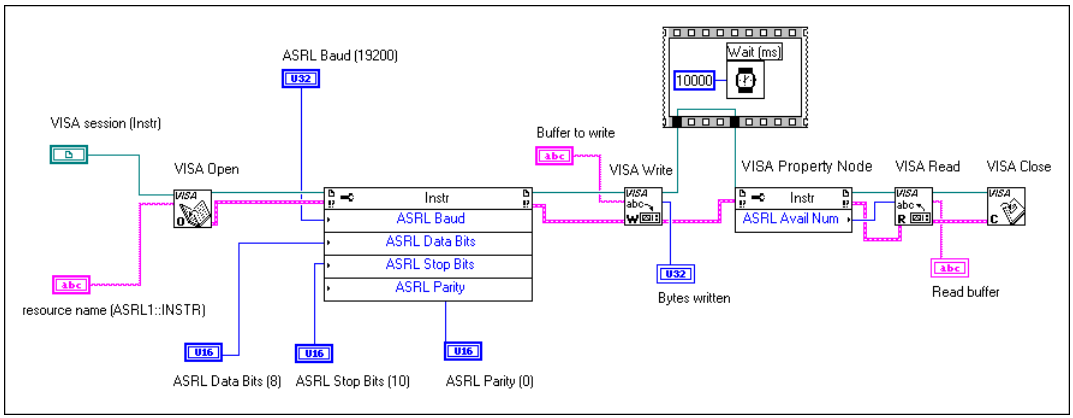
Es gibt zusätzlich zu den oben aufgelisteten Eigenschaften noch viele andere. Es gibt auch Eigenschaften, die für einen gewissen Schnittstellentyp nicht spezifisch sind. Die Timeout-Eigenschaft, d.h., der in meldungsbasierten I/O-Operationen verwendete Timeout, ist ein gutes Beispiel von einer solchen Eigenschaft. Die vollständigste Quelle für Informationen über Eigenschaften ist die LabVIEW *Online-Referenz*, auf die Sie durch Wählen von **Hilfe»Online-Referenz** zugreifen können.

Die Online-Hilfe zeigt den Schnittstellentyp, für den die Eigenschaft gilt, ob die Eigenschaft lokal oder global ist, ihren Datentyp und den gültigen Bereich von Werten für die Eigenschaft. Sie zeigt außerdem verwandte Angaben und bietet eine detaillierte Beschreibung der Eigenschaft.

Beispiele von VISA-Eigenschaften

Serielles Lesen und Schreiben

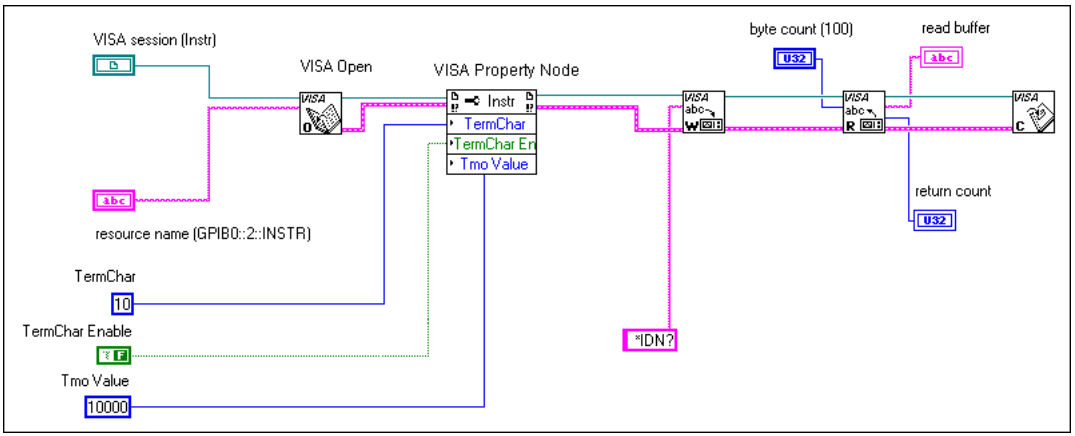
Dieser Abschnitt enthält drei einfache Beispiele zur Verwendung von Eigenschaften in VISA-Programmen. Das erste, in der folgenden Abbildung dargestellt, schreibt einen String an ein serielles Instrument und liest die Antwort.



Das VI eröffnet eine Session zum seriellen Anschluß COM 1 und initialisiert ihn auf 19200 Baud, 8 Datenbit, keine Parität und 1 Stoppbit. Ein String wird dann an den Anschluß geschrieben. Nach dem Schreiben des Strings und einer Wartezeit von 10 Sekunden wird die Anzahl der vom Gerät zurückgegebenen Bytes durch den Gebrauch von einer anderen VISA-Eigenschaft erfaßt. Diese Bytes werden dann vom Anschluß gelesen. Beachten Sie, daß man die Zahl 10 zur Einstellung der Anzahl von Stoppbits auf 1 verwendet. (Dies kommt von der VISA-Spezifikation her. 10 entspricht 1 Stoppbit, 20 entspricht 2 Stoppbits.)

Wie bestimme ich ein Abschlußzeichen für eine Leseoperation?

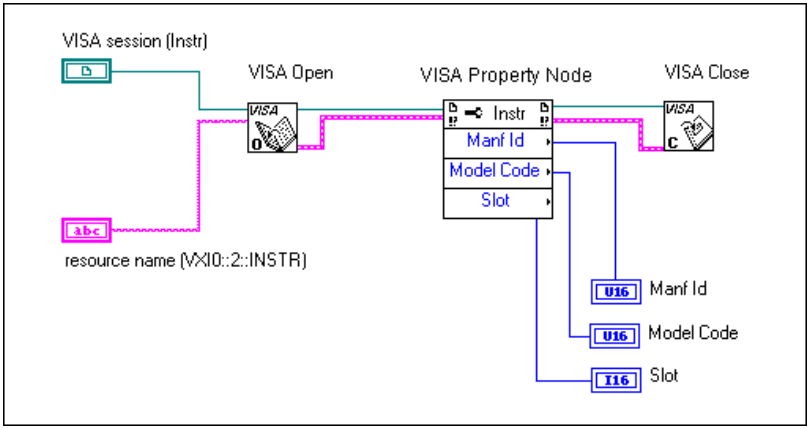
Das folgende Beispiel zeigt, wie Eigenschaften zur Bestimmung von einem Abschlußzeichen für VISA-Leseoperationen verwendet werden. Einige meldungsbasierte Geräte schicken ein besonderes Abschlußzeichen, wenn sie keine Daten mehr zu übertragen haben.



Dieses VI eröffnet eine Session zum GPIB-Instrument auf Hauptadresse 2. Das VI bestimmt das Abschlußzeichen als Zeilenvorschub (Dezimalwert 10) und erlaubt dann mit einer anderen Eigenschaft die Verwendung von einem Abschlußzeichen. Das VI stellt außerdem die Timeout-Eigenschaft auf 10 000 Millisekunden (10 Sekunden). Dann schreibt es den String *IDN? an das Instrument und versucht, eine 100-zeichenlange Antwort zurückzulesen. Das Lesen schließt ab, sobald das Abschlußzeichen empfangen wird. Das VI hält an, wenn das Abschlußzeichen empfangen wird, nachdem 100 Bytes gelesen worden sind oder nach 10 Sekunden.

VXI-Eigenschaften

Das letzte Beispiel zeigt, wie einige der üblichen, mit einem VXI-Instrument assoziierten Eigenschaften gelesen werden.



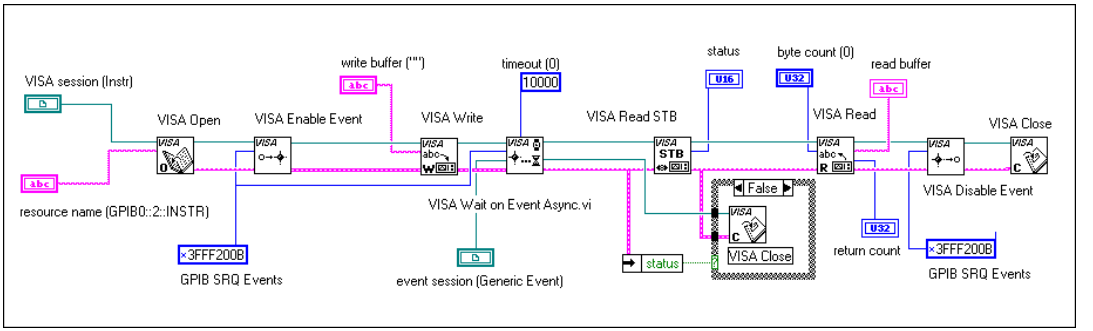
Dieses VI eröffnet eine Session zu einem VXI-Instrument auf Logik-Adresse 2 und liest die Hersteller-ID, den Typcode und den Slot für den VXI-Modul.

Ereignisse

Ein Ereignis ist ein Kommunikationsmittel von VISA zwischen einer Ressource und ihren Anwendungen. Es ist ein Mittel, mit dem die Ressource der Anwendung meldet, daß eine Bedingung eingetreten ist, die seitens der Anwendung eine Handlung erfordert. Beispiele von verschiedenen Ereignissen sind in den folgenden Abschnitten enthalten.

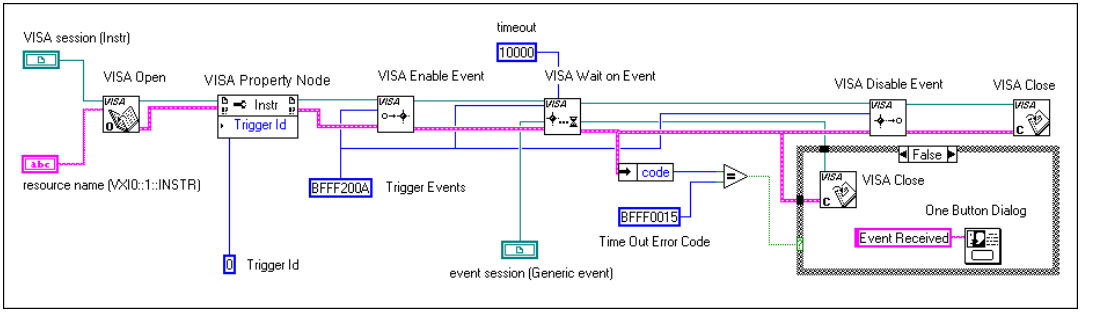
GPIB SRQ-Ereignisse

Das folgende Blockdiagramm zeigt, wie GPIB Service Request (SRQ) Ereignisse mit VISA behandelt werden.



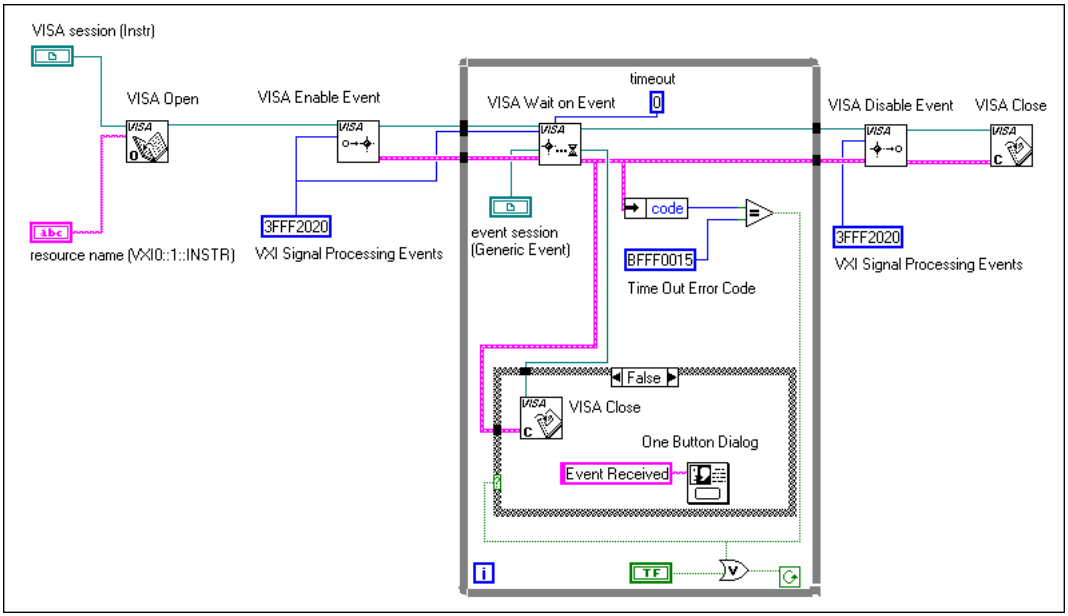
Das VI aktiviert Service Request-Ereignisse und schreibt einen Befehlsstring an das Instrument. Es wird erwartet, daß das Instrument mit einem SRQ antwortet, sobald es den String verarbeitet hat. Das VI Wait on Event Async wird verwendet, um bis zu 10 Sekunden auf Eintritt des SRQ-Ereignisses zu warten. Sobald das SRQ eintritt, wird das Statusbyte des Instruments mit dem VI Read Status Byte gelesen. Das Statusbyte muß gelesen werden, nachdem GPIB SRQ-Ereignisse eintreten, da sonst spätere SRQ-Ereignisse eventuell nicht richtig empfangen werden. Zuletzt wird die Antwort vom Instrument gelesen und angezeigt. Das VI Wait on Event Async unterscheidet sich vom normalen VI Wait on Event dadurch, daß es kontinuierlich Wait on Event mit einem Timeout von Null aufruft, um das Ereignis abzufragen. Hierdurch wird Zeit eingespart, damit andere parallele Programmsegmente ausgeführt werden können, während sie auf das Ereignis warten.

Trigger-Ereignisse



Dieses Diagramm zeigt, wie ein Trigger auf TTL Trigger Line 0 für ein Gerät auf Logik-Adresse 1 erfaßt wird. Sie müssen den Typ von zu erfassenden Trigger-Ereignissen mit einer VISA-Eigenschaft einstellen, bevor die Ereignisse aktiviert werden. Das VI wartet bis zu 10 Sekunden auf den Empfang des Ereignisses. Wird das Ereignis erfolgreich empfangen, so wird das Ereignis im VI abgeschlossen.

Interrupt-Ereignisse

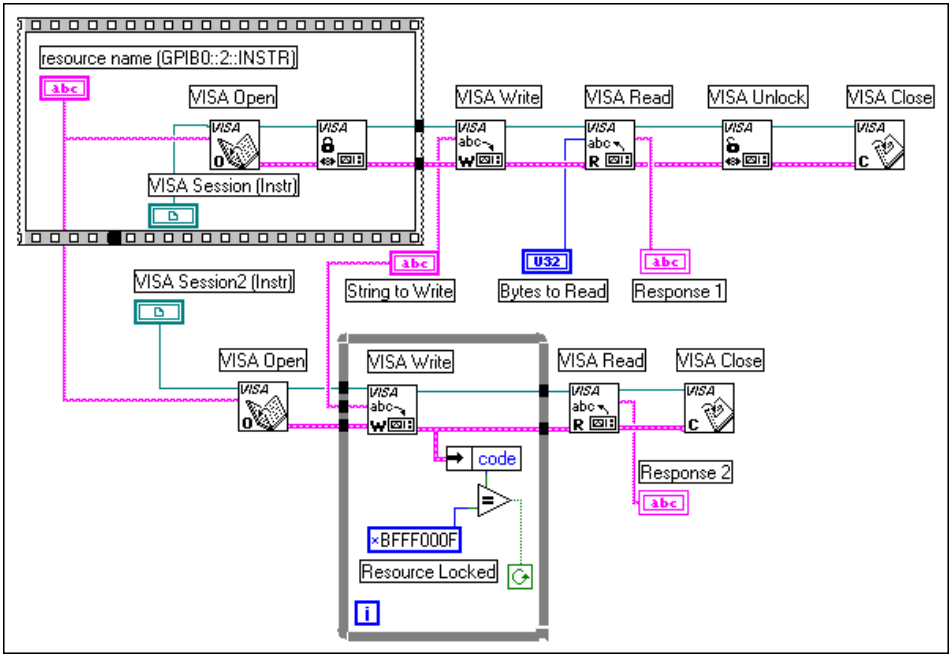


Dieses Diagramm demonstriert den Gebrauch der Fähigkeiten von VISA zur Ereignisbehandlung, um ein von einem VXI-Gerät auf Logik-Adresse 1 bestimmtes VXI-Interrupt zu erfassen. Das VI aktiviert VXI Signalverarbeitungs-Ereignisse und tritt dann in eine Schleife, die wiederholt VISA Wait on Event aufruft. Die Schleife schließt ab, wenn ein Ereignis empfangen wird oder ein Frontpanel-Stoppschalter ausgewählt wird. Das VI Wait on Event hat ein Timeout-Terminal, das auf einen Wert von Null gesetzt wird. In diesem Fall, überprüft das VI einfach, ob Ereignisse empfangen worden sind und gibt dann sofort einen Timeout-Fehler zurück, wenn kein Ereignis in der Ereignis-Warteschlange steht. Wenn ein Ereignis empfangen wird, dann wird die Ereignis-Session geschlossen und eine Meldung des Ereignisses wird erzeugt. Sobald die Ereignisbehandlung fertig ist, werden die Ereignisse deaktiviert.

Fixierung

VISA führt Fixierung ein, um den Zugriff auf Ressourcen zu steuern. In VISA können GPIB- und VXI-Anwendungen mehrere Sessions zur gleichen Ressource gleichzeitig offenhalten und können auf diese Ressource über jene verschiedene Sessions gleichzeitig zugreifen. In einigen Fällen müssen Anwendungen, die auf eine Ressource zugreifen, anderen Sessions den Zugriff auf die Ressource verbieten. Zum Beispiel: Es könnte nötig sein, daß eine Anwendung eine Schreib- und Leseoperation als einen einzigen Schritt ausführt, damit keine anderen Operationen zwischen den Schreib- und Leseoperationen stattfinden. Damit sie so in einem einzelnen Schritt ausgeführt werden, kann die Anwendung die Ressource vor dem Aufruf der Schreiboperation fixieren und es nach der Leseoperation unfixieren.

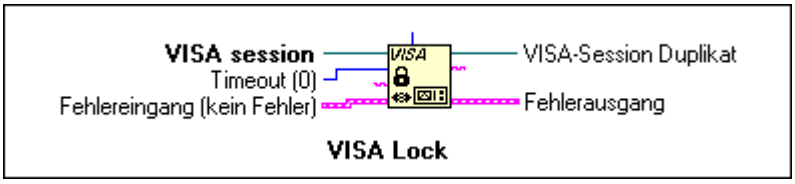
Der VISA-Fixiermechanismus erzwingt die Arbitration von Ressource-Zugriffen auf individueller Basis. Wenn eine Session eine Ressource fixiert, werden durch andere Sessions aufgerufene Operationen bedient oder mit einem Fixierungsfehler zurückgegeben, je nach der Operation und dem Typ der verwendeten Fixierung.



Das VI eröffnet zwei Sessions zur gleichen Ressource, aber es fixiert die erste Session. Die erste Session schreibt dann einen Befehl an die Ressource und liest die Antwort. Sobald die Schreib-/Leseabfolge beendet ist, unfixiert die erste Session die Ressource. Zu diesem Zeitpunkt wird die zweite Session, die versucht, dieselbe Schreib-/Leseoperation auszuführen, keinen Ressource-fixierten-Fehler mehr für die Schreiboperation empfangen und kann erfolgreich abschließen. Fixierung kann in Fällen verwendet werden, in denen mehr als eine Anwendung auf dieselbe Ressource zugreift oder mehrfache Sessions zur selben Ressource in einer einzigen Anwendung eröffnet werden.

Geteilte Fixierung

Es kann Fälle geben, in denen Sie den Zugriff auf eine Ressource fixieren, aber diesen Zugriff selektiv teilen wollen. Die folgende Abbildung zeigt das VI Lock in komplexer Hilfsansicht.



Fixierungtyp ist standardmäßig exklusiv, aber Sie können ihn auf Teilung einstellen. Dann können Sie einen String mit dem **erforderlichen Schlüssel** als das nötige Paßwort verbinden, mit dem andere Anwendungen auf die Ressource zugreifen. Das VI weist aber auch ein solches in **Zugangsschlüssel** zu, falls Sie es nicht anfordern. Sie können dann diesen Schlüssel zum Zugriff auf eine fixierte Ressource benutzen.

Plattformspezifische Fragen

Dieser Abschnitt diskutiert Programmierinformation, die Sie beim Entwickeln von Anwendungen, die den NI-VISA-Treiber benutzen, in Betracht ziehen sollten.

Nach Installierung der Treibersoftware, können Sie anfangen, Ihre VISA-Anwendungssoftware zu entwickeln. Erinnern Sie sich daran, daß der NI-VISA-Treiber auf NI-488.2 und NI-VXI für I/O-Zugriffe auf Treiberebene beruht.

- **Benutzer von Windows 95/NT**—Auf VXI- und MXI-Systemen benutzen Sie T&M Explorer, um den VXI Ressourcenmanager auszuführen, zur Konfigurierung von Ihrer Hardware und zur

Zuweisung von VME- und GPIB-VXI-Adressen. Für GPIB-Systeme verwenden Sie den Geräte-Manager des Systems zur Hardware-Konfigurierung. Um Instrumente über serielle Anschlüsse zu steuern, können Sie T&M Explorer zur Änderung der Standardeinstellungen benutzen, oder Sie können alle nötige Konfigurierung zur Laufzeit durch die Einstellung von VISA-Attributen durchführen.

- **Alle anderen Plattformen**—Auf VXI- und MXI-Systemen müssen Sie immer noch VXIinit und Resman ausführen und VXIedit oder VXItedit zu Konfigurationszwecken benutzen. Für GPIB- und GPIB-VXI-Systemen können Sie in ähnlicher Weise die GPIB Control Panel Anwendung oder IBCONF zur Konfigurierung von Ihrem System verwenden. Um Instrumente über serielle Anschlüsse zu steuern, können Sie alle nötige Konfigurierung zur Laufzeit durch Einstellung von VISA-Attributen durchführen.

Betrachtungen für die Programmierung

Dieser Abschnitt enthält Information, die Sie bei der Entwicklung von Anwendungen, die die NI-VISA I/O-Schnittstellensoftware verwenden, in Betracht ziehen sollen.

Mehrfache Anwendungen unter Verwendung des NI-VISA-Treibers

Unterstützung für mehrfache Anwendungen ist ein wichtiges Merkmal in allen Implementationen des NI-VISA-Treibers. Sie können mehrere Anwendungen, die NI-VISA verwenden, gleichzeitig laufen lassen. Sie können sogar mehrfache Instanzen derselben Anwendung, die den NI-VISA-Treiber verwendet, gleichzeitig laufen lassen, sofern Ihre Anwendung dafür ausgelegt ist. Die NI-VISA-Operationen funktionieren in derselben Weise, ob Sie nur eine Anwendung oder mehrere Anwendungen (oder mehrere Instanzen einer Anwendung) haben, die alle versuchen, den NI-VISA-Treiber zu verwenden.

In Fällen, wenn Sie mehrfache Anwendungen oder Sessions, die die Low-Level VXI-Bus-Zugriffsfunktionen benutzen, haben, müssen Sie aber vorsichtig sein. Die zum Zugriff auf den VXI-Bus verwendeten Speicherbereiche sind eine eingeschränkte Ressource. Sie sollten die Operation VI MapAddress() aufrufen, bevor Sie versuchen, Low-Level VXI-Bus-Zugriff mit viPeek*XX*() oder viPoke*XX*() durchzuführen. Nachdem die Zugriffe abgeschlossen sind, sollten Sie stets die Operation VI UnmapAddress() sofort aufrufen, damit Sie den Speicherbereich für andere Anwendungen freigeben.

Fragen zur Unterstützung von mehrfachen Schnittstellen

Dieser Abschnitt enthält Information darüber, wie Ihre NI-VISA-Software für bestimmte Schnittstellentypen benutzt und/oder konfiguriert wird.

VXI- und GPIB-Plattformen

NI-VISA unterstützt alle bestehende VXI-, GPIB- und serielle Hardware von National Instruments für die Betriebssysteme, auf denen NI-VISA existiert. Für VXI umfaßt das die MXI-1 und MXI-2 Plattformen, das GPIB-VXI und die Linie von VXIpc eingebetteten Computern. Für GPIB umfaßt dies, ist aber nicht eingeschränkt auf, folgendes: die Serien PCI-GPIB, NB-GPIB, GPIB-SPARC, die ganze Linie von AT-GPIB/TNT-Karten und den GPIB-ENET-Kasten, den Sie zur Fernsteuerung von GPIB-Geräten verwenden können. Mit dem GPIB-ENET können Sie sogar VXI-Geräte fernsteuern, wenn Sie einen GPIB-VXI Controller benutzen.

Mehrfacher GPIB-VXI Support

Windows 95/NT-Benutzer können sich an die T&M Explorer-Utility wenden, um mehrfache GPIB-VXI-Controller von National Instruments oder auch einen GPIB-VXI-Controller von anderen Lieferanten ihrem System hinzuzufügen. Windows 3.*x*- und UNIX-Benutzer müssen die VISAconf-Utility verwenden, um die Controller hinzuzufügen.

Unterstützung von seriellen Anschlüssen

NI-VISA unterstützt gegenwärtig jeweils nur eine einzige Session an einem gegebenen seriellen Anschluß. Die Höchstzahl von seriellen Anschlüssen, die NI-VISA gegenwärtig auf irgendeiner Plattform unterstützt, ist 32. Die Standardnumerierung von seriellen Anschlüssen ist systemabhängig.

Plattform	Methode
Windows 3.*x*, Windows 95, Windows NT	ASRL1 – ASRL4 greifen auf COM1 – COM4 zu ASRL10 – ASRL13 greifen auf LPT1 – LPT4 zu
Macintosh 68K, Macintosh PPC	ASRL1 greift auf den Modemanschluß zu ASRL2 greift auf den Druckeranschluß zu
Solaris 1.*x*	ASRL1 – ASRL6 greifen auf /dev/ttya – /dev/ttyf zu
Solaris 2.*x*	ASRL1 – ASRL6 greifen auf /dev/cua/a – /dev/cua/f zu
HP-UX 9, HP-UX 10	ASRL1 und ASRL2 greifen auf serielle Anschlüsse 1 und 2 über /dev/tty00 und /dev/tty01 auf HP-UX9 zu. HP-UX10 benutzt /dev/tty0p0 und /dev/tty1p0. Weitere Anschlüsse werden konsekutiv numeriert ab ASRL3, der /dev/tty02 benutzt.

VME-Unterstützung

Um auf VME-Geräte in Ihrem System Zugriff zu haben, müssen Sie NI-VXI so konfigurieren, daß Sie diese Geräte sehen. Windows 95/NT-Benutzer können NI-VXI unter Verwendung von **Add Device Wizard** im T&M Explorer konfigurieren. Benutzer auf anderen Plattformen müssen den **Non-VXI Device Editor** in VXIedit oder VXItedit verwenden. Für jeden Adreßbereich, in dem Ihr Gerät Speicher besitzt, müssen Sie eine getrennte Pseudo-Geräteintragung mit einer Logik-Adresse zwischen 256 und 511 erstellen. Zum Beispiel: Ein VME-Gerät mit Speicher in den Bereichen A24 und A32 braucht zwei Eintragungen. Sie können ebenfalls bestimmen, welche Interrupt-Ebenen das Gerät benutzt. VXI- und VME-Geräte können Interrupt-Ebenen nicht teilen. Dann können Sie auf das Gerät von NI-VISA aus genauso wie auf ein VXI-Gerät zugreifen, nämlich durch Angabe von dem Adreßbereich und dem Offset von der Basisadresse, wo Sie das Gerät konfiguriert haben. NI-VISA-Unterstützung für VME-Geräte umfaßt die Registerzugriff-Operationen (sowohl High-Level als auch Low-Level) und die Blockversetzungs-Operationen, sowie die Fähigkeit, Interrupts zu empfangen.

Das Debugging für ein VISA-Programm

Dieser Abschnitt enthält Informationen zum Debugging von VISA-Programmen. Naturgemäß gibt es mehr Möglichkeiten, die beim Debugging von VISA-Problemen in Betracht gezogen werden müssen, als wenn mit eigenständigen Treibern gearbeitet wird. VISA macht Aufrufe in serielle APIs von NI-VXI, NI-488 oder dem Betriebssystem. Daher können in VISA vorkommende Probleme mit dem von VISA aufgerufenen Treiber zu tun haben und nicht mit VISA selbst.

Falls anscheinend keine VISA VIs (Gerätetreiber eingeschlossen) in LabVIEW funktionieren, dann ist der erste Schritt das VI VISA Find Ressource. Dieses VI läuft ohne andere VISA-VIs im Blockdiagramm. Wenn dieses VI seltsame Fehler wie nicht-standardmäßige VISA-Fehler erzeugt, so ist es am wahrscheinlichsten, daß die falsche Version von VISA installiert wurde oder daß VISA nicht richtig installiert wurde. Wenn VISA Find Ressource richtig läuft, dann funktioniert LabVIEW richtig mit dem VISA-Treiber. Der nächste Schritt ist, die Abfolge von VIs zu identifizieren, die den Fehler im LabVIEW-Programm verursachen.

Wenn eine einfache Abfolge von Ereignissen den Fehler verursacht, so ist es gut, als nächster Schritt im Debugging zu versuchen, dieselbe Abfolge interaktiv mit der VISAIC-Utility auszuführen (vgl. den folgenden Abschnitt). Es ist im allgemeinen ratsam, die anfängliche Programmentwicklung interaktiv durchzuführen. Wenn die interaktive Utility erfolgreich funktioniert, aber dieselbe Abfolge in LabVIEW nicht funktioniert, so ist dies ein Anzeichen dafür, daß LabVIEW ein Problem bei der Interaktion mit dem VISA-Treiber hat. Wenn dieselbe Abfolge dasselbe Problem interaktiv in VISAIC zeigt, dann ist es möglich, daß ein Problem mit einem der von VISA aufgerufenen Treibern besteht. Sie können mit den interaktiven Utilities für diese Treiber (VIC für NI-VXI und IBIC für NI-488.2) versuchen, die äquivalenten Operationen durchzuführen. Wenn die Probleme auch auf dieser Ebene andauern, ist dies ein Anzeichen dafür, daß ein Problem mit einem Treiber auf niedrigerer Ebene oder mit dessen Installierung vorliegt.

Debugging-Werkzeug für Windows 95/NT

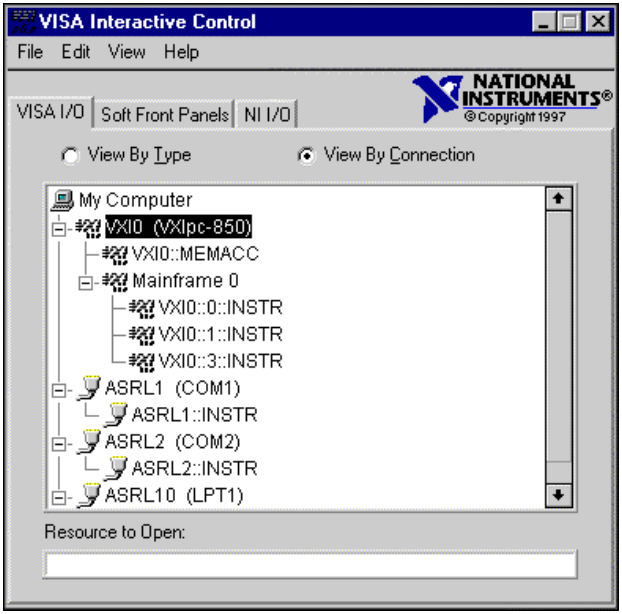
NI Spy protokolliert die Aufrufe, die Ihre Anwendung an National Instruments Test- und Messungs-Treiber (T&M) macht, darunter NI-VISA, NI-VXI und NI-488.2. Benutzer von NI-488.2 werden vielleicht bemerken, daß NI Spy dem GPIB Spy ähnlich ist.

NI Spy hebt Funktionen, die Fehler zurückgeben, hervor, damit sie schnell feststellen können, welche Funktionen während Ihrer Entwicklungsarbeit gescheitert sind. NI Spy kann auch Aufrufe an diese Treiber von Ihrer Anwendung protokollieren, damit Sie sie zu einem gelegenen Zeitpunkt nach Fehlern überprüfen können.

VISAIC

VISA wird mit einer Utility namens VISA Interactive Control (VISAIC) auf allen VISA und LabVIEW unterstützenden Plattformen außer Macintosh mitgeliefert. Diese Utility gibt interaktiv und in einer leicht zu benutzenden graphischen Umgebung Zugang zur ganzen Funktionalität von VISA. Sie ist ein bequemer Anfangspunkt für die Programmentwicklung und zum Lernen von VISA.

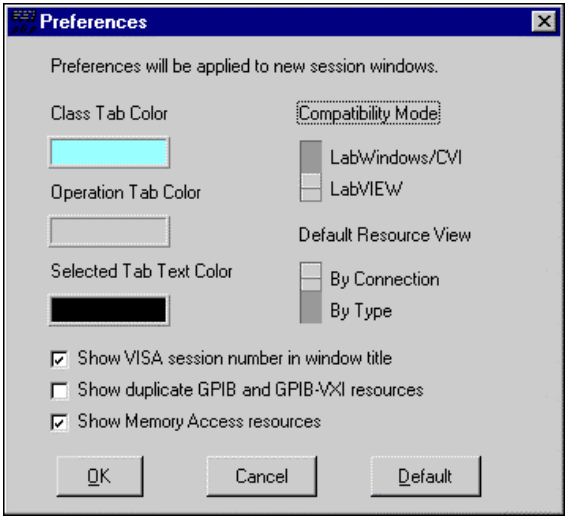
Wenn VISAIC läuft, findet es automatisch alle verfügbaren Ressourcen im System und listet die Instrumenten-Deskriptoren für jede dieser Ressourcen unter dem entsprechenden Ressource-Typ. Das VISAIC-Startfenster ist in der untenstehenden Abbildung gezeigt.



Der Tab Soft Front Panels des Hauptpanels von VISAIC bietet Ihnen die Option, die Soft-Frontpanels von allen im System installierten VXI*plug&play* Instrumenten-Treibern zu starten.

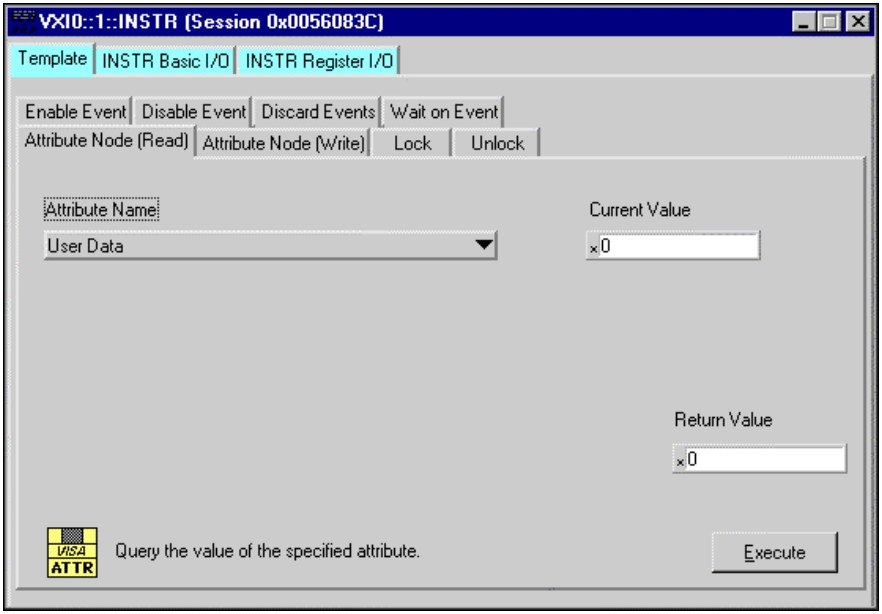
Der Tab NI I/O bietet Ihnen die Option, die NI-VXI interaktive Utility oder die NI-488 interaktive Utility zu starten. Dies verschafft Ihnen bequeme Verknüpfungen zu den interaktiven Utilities für den von VISA aufgerufenen Treiber, falls Sie das Debugging auf dieser Ebene versuchen wollen.

Ein Doppelklick auf einem der im VISAIC-Fenster gezeigten Instrumenten-Deskriptoren eröffnet eine Session zu diesem Instrument. Die Eröffnung von einer Session zum Instrument erzeugt ein Fenster mit einer Reihe von Tabs zur interaktiven Ausführung von VISA-Befehlen. Das genaue Aussehen von diesen Tabs hängt von dem Kompatibilitätsmodus ab, in dem sich VISA befindet. Für Zugriff auf den Kompatibilitätsmodus, sowie auf andere Voreinstellungen von VISAIC, wählen Sie **Bearbeiten» Voreinstellungen…**, um das unten gezeigte Fenster hervorzurufen.



Die Implementationen von VISA unterscheiden sich geringfügig in LabVIEW und LabWindows/CVI. Diese Unterschiede widerspiegeln sich in den Operationen-Tabs, die bei Eröffnung von einer Session zu einer Ressource gezeigt werden. Der Kompatibilitätsmodus ist standardmäßig auf LabWindows/CVI eingestellt, und dies sollten Sie in LabVIEW ändern. Die geänderten Voreinstellungen werden für jede danach eröffnete Session gültig sein.

Wenn eine Session zu einer Ressource interaktiv eröffnet wird, erscheint ein Fenster, das dem unten gezeigten ähnelt.



Drei Haupt-Tabs erscheinen im Fenster. Der anfängliche Tab ist der Vorlage-Tab, der alle Operationen enthält, die mit Ereignissen, Eigenschaften und Fixierungen zu tun haben. Bemerken Sie, daß es für jede dieser Operationen einen anderen Tab unter dem Haupt-Tab gibt. Die anderen Haupt-Tabs sind INSTR Basic I/O und INSTR Register I/O. Der Tab Basic I/O enthält die grundlegenden Operationen für meldungsbasierte Instrumente, während der Tab Register I/O die grundlegenden Operationen für registerbasierte Instrumente enthält. Der Tab Register I/O erscheint nur bei VXI-Instrumenten.

9

Einführung in die LabVIEW GPIB-Funktionen

Dieses Kapitel erklärt, wie der General Purpose Interface Bus (GPIB) funktioniert, sowie den Unterschied zwischen den IEEE 488- und IEEE 488.2-Schnittstellen.

Es gibt zwei GPIB-Normen—IEEE 488 und IEEE 488.2. Hewlett-Packard entwarf den GPIB (ursprünglich den HP-IB genannt), um seine Reihe von programmierbaren Instrumenten zusamenzuschalten und zu steuern. Der GPIB wurde wegen seiner Datenübertragungsraten von bis zu 1 Mbyte/s bald auch für andere Anwendungen, wie z.B. die Kommunikation zwischen Computern und die Steuerung von Peripheralgeräten, verwendet. Er wurde später als IEEE Norm 488-1975 akzeptiert und hat sich seitdem zur ANSI/IEEE Norm 488.2-1987 herausgebildet. Die Vielseitigkeit des Systems hat den Namen General Purpose Interface Bus nahegelegt.

National Instruments brachte den GPIB an die Benutzer von Computern und Geräten von anderen Herstellern als Hewlett-Packard, mit einer Spezialisierung sowohl in hochleistungsfähigen und sehr schnellen Hardwareschnittstellen als auch in umfassender, voll funktionsfähiger Software. Die GPIB-Funktionen für LabVIEW folgen der Norm IEEE 488.2.

Meldungstypen

Der GPIB trägt geräteabhängige Meldungen und Schnittstellenmeldungen.

- Geräteabhängige Meldungen, oft *Daten* oder *Datenmeldungen* genannt, enthalten gerätespezifische Information, wie z.B. Programmierungsbefehle, Meßergebnisse, Maschinenzustand und Datendateien.
- Schnittstellenmeldungen verwalten den Bus selbst. Sie heißen üblicherweise *Befehle* oder *Befehlsmeldungen*. Schnittstellenmeldungen führen Ausgaben wie die Initialisierung des Busses, die Adressierung und De-Adressierung von Geräten und die Einstellung von Gerätemodi für entfernte oder lokale Programmierung durch.

Der Begriff *Befehl* in dem hier gebrauchten Sinn ist nicht zu verwechseln mit einigen Anweisungen für Geräte, die auch Befehle heißen können. Diese gerätespezifischen Anweisungen sind in Wirklichkeit Datenmeldungen.

Die ANSI/IEEE Norm 488.2-1987 erweiterte die frühere IEEE 488.1 Norm um eine genaue Beschreibung davon, wie der Controller den GPIB verwalten soll. Dies umfaßt u.a. die Standardmeldungen, die konforme Geräte verstehen sollen; die Mechanismen zur Meldung von Gerätefehlern und anderen Zustandsinformationen und die verschiedenen Protokolle, die an den Bus angeschlossene konforme Geräte entdecken und konfigurieren.

IEEE 488.2 besitzt die Fähigkeit, jede Busleitung jederzeit zu überwachen, was maßgeblich für die Erfassung von aktiven Geräten (Talker und Listener) auf dem GPIB ist. GPIB-Geräte können Talker, Listener und/oder Controller sein. Ein digitaler Spannungsmesser ist z.B. ein Talker und kann auch Listener sein. Ein Talker sendet Datenmeldungen an einen oder mehrere Listener. Der Controller verwaltet den Informationsfluß auf dem GPIB, indem er Befehle an alle Geräte sendet.

Der GPIB ist ein gewöhnlicher Computerbus, abgesehen davon, daß der Computer seine Schaltungskarten durch einen Rückwandbus zusammenschaltet, während der GPIB eigenständige Geräte hat, die über einen Kabelbus zusammengeschaltet sind.

Die Rolle des GPIB-Controllers ähnelt der einer CPU in einem Computer, aber eine bessere Analogie stellt die Vermittlungszentrale von einem städtischen Telefonsystem dar. Die Vermittlungszentrale (Controller) überwacht das Kommunikationsnetzwerk (GPIB). Sobald die Zentrale (Controller) bemerkt, daß eine Partei (Gerät) einen Anruf machen möchte (eine Datenmeldung senden möchte), verbindet sie den Anrufer (Talker) mit dem Empfänger (Listener).

Der Controller adressiert einen Talker und einen Listener, bevor der Talker seine Meldung an den Listener senden kann. Nachdem der Talker die Meldung übermittelt hat, darf der Controller beide Geräte de-adressieren.

Einige Buskonfigurationen brauchen keinen Controller. Zum Beispiel: Ein Gerät kann immer ein Talker (ein sogenanntes Talk-only-Gerät) oder ein Listen-only-Gerät sein.

Ein Controller ist nötig, wenn Sie den aktiven oder adressierten Talker wechseln müssen. Normalerweise übernimmt ein Computer die Controllerfunktion.

Mit der GPIB-Karte und ihrer Software übernimmt Ihr PC folgende drei Rollen:

- Controller—den PIB verfolgen
- Talker—Daten senden
- Listener—Daten empfangen

Der Controller-In-Charge und der System-Controller

Obwohl mehrfache Controller auf dem GPIB erlaubt sind, darf jeweils nur ein Controller aktiv bzw. Controller-In-Charge (CIC) sein. Sie können die aktive Steuerung vom aktuellen CIC an einen inaktiven Controller übertragen. Nur ein Gerät auf dem Bus—der System-Controller—darf sich selbst zum CIC machen. Die GPIB-Karte ist normalerweise der System-Controller.

Kompatible GPIB-Hardware

Die folgenden GPIB-Hardwareprodukte von National Instruments sind mit LabVIEW kompatibel:

LabVIEW für Windows 95 und Windows 95-Japanisch

- AT-GPIB/TNT, AT-GPIB/TNT (PnP), AT-GPIB/TNT+, PCI-GPIB
- PCMCIA-GPIB, PCMCIA-GPIB+
- GPIB-ENET
- EISA-GPIB
- VXIpc Model 850
- NEC-GPIB/TNT, NEC-GPIB/TNT (PnP)
- GPIB-PCII/IIA
- PC/104-GPIB
- CPCI-GPIB
- GPIB-ENET
- PMC-GPIB

LabVIEW für Windows NT

- AT-GPIB, AT-GPIB/TNT
- PCMCIA-GPIB
- PCI-GPIB
- VXIpc Modell 850
- GPIB-ENET

LabVIEW für Windows 3.1

- AT-GPIB, AT-GPIB/TNT, AT-GPIB/TNT (PnP), AT-GPIB/TNT+, PCI-GPIB
- PCMCIA-GPIB, PCMCIA-GPIB+
- GPIB-ENET
- EISA-GPIB
- VXIpc Modell 850
- NEC-GPIB/TNT (Japanisch), NEC-GPIB/TNT (PnP) (Japanisch), GPIB-PCII/IIA
- GPIB-232CT-A
- GPIB-485CT-A
- GPIB-1284CT
- PCII/IIA
- STD-GPIB
- EXM-GPIB
- MC-GPIB

LabVIEW für Mac OS

- PCI-GPIB
- NB-GPIB/TNT, NB-GPIB-P/TNT
- PCMCIA-GPIB
- LC-GPIB
- GPIB-ENET
- GPIB-232CT-A
- GPIB-SCSI-A
- PC/104-GPIB
- NB-DMA2800 (Nur traditionelle GPIB-Funktionen)

LabVIEW für HP-UX

- GPIB-ENET
- EISA-GPIB
- AT-GPIB/TNT

LabVIEW für Sun

- GPIB-ENET
- GPIB-SCSI-A
- SB-GPIB/TNT

LabVIEW für Concurrent PowerMAX

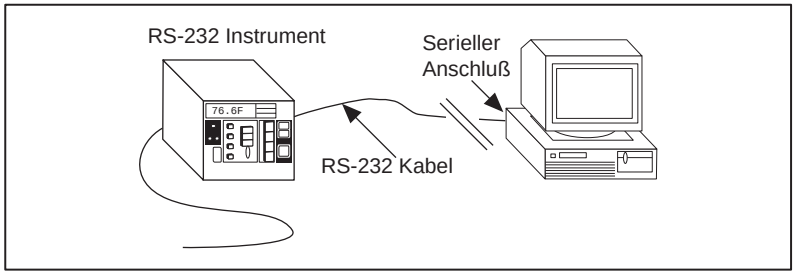
- GPIB-1014
- GPIB-1014D
- GPIB-1014P
- GPIB-1014DP

10

VIs für den seriellen Anschluß

Dieses Kapitel beschreibt die VIs für Operationen mit dem seriellen Anschluß und erklärt die wichtigen Faktoren, die die serielle Kommunikation beeinflussen.

Die serielle Kommunikation ist ein beliebtes Mittel für die Datenübertragung zwischen einem Computer und einem peripheren Gerät, wie z.B. einem programmierbaren Instrument oder sogar einem anderen Computer. Die serielle Kommunikation verwendet einen Sender, der Daten, ein Bit nach dem anderen, über eine einzige Kommunikationsleitung an einen Empfänger sendet. Sie können von dieser Methode Gebrauch machen, wenn die Datenübertragungsraten niedrig sind oder wenn Sie Daten über große Entfernungen übertragen müssen.



Die serielle Kommunikation ist deshalb populär, weil die meisten Computer einen oder zwei serielle Anschlüsse besitzen. Viele GPIB-Instrumente sind gleichfalls mit seriellen Anschlüssen ausgestattet. Eine Einschränkung der seriellen Kommunikation liegt jedoch darin, daß ein serieller Anschluß nur mit jeweils einem Gerät kommunizieren kann.

Einige periphere Geräte brauchen Zeichen zum Abschluß von den an sie gesendeten Datenstrings. Gebräuchliche Abschlußzeichen sind ein Wagenrücklauf, ein Zeilenvorschub oder ein Semikolon. Wenden Sie sich an das Gerät-Handbuch zur Feststellung, ob ein Abschlußzeichen erforderlich ist.

Für Beispiele der Verwendung von VIs für den seriellen Anschluß siehe `examples\instr\smplserl.llb`.

Handshake-Typen

Ein häufiges Problem in der seriellen Kommunikation ist die Sicherstellung, daß Sender und Empfänger beide bei der Datenübertragung mithalten können. Der Treiber für den seriellen Anschluß kann einkommende/abgehende Informationen in einem Puffer speichern, aber jener Puffer ist von begrenzter Größe. Wenn er voll wird, ignoriert der Computer neue Informationen, bis genug Daten aus dem Puffer gelesen werden, um Platz für neue Information zu schaffen.

Handshaking hilft zu verhindern, daß dieser Puffer überläuft. Mit Handshaking melden der Sender und der Empfänger einander, wenn die Puffer voll werden. Der Sender kann dann die Sendung von neuen Informationen einstellen, bis das andere Ende der seriellen Kommunikation für neue Daten bereit ist.

Sie können in LabVIEW zwei Arten von Handshaking durchführen -Software-Handshaking und Hardware-Handshaking. Sie schalten diese Formen von Handshaking durch den Gebrauch vom VI Serial Port Init an oder aus. Standardmäßig verwenden die VIs kein Handshaking.

Software-Handshaking—XON/XOFF

XON/XOFF ist ein Protokoll für das Software-Handshaking, das Sie verwenden können, um Überlauf der Puffer vom seriellen Anschluß zu vermeiden. Sobald der Empfangspuffer fast voll ist, sendet der Empfänger XOFF (<Strg-S> [Dezimal 19]), um das andere Gerät anzuweisen, die Sendung von Daten einzustellen. Sobald der Empfangspuffer ausreichend leer ist, sendet der Empfänger XON (<Strg-Q> [Dezimal 17]) als Information, daß die Sendung wieder anfangen kann. Wenn Sie XON/XOFF aktivieren, interpretieren die Geräte <Strg-Q> und <Strg-S> immer als XON- und XOFF-Zeichen, nie als Daten. Wenn sie XON/XOFF deaktivieren, können Sie <Strg-Q> und <Strg-S> als Daten senden. Verwenden Sie XON/XOFF nicht mit binären Datenübertragungen, da <Strg-Q> oder <Strg-S> möglicherweise in den Daten eingebettet ist und die Geräte sie als XON bzw. XOFF statt als Daten interpretieren.

Fehlercodes

Sie können den Parameter **Fehlercode** mit einem der VIs für Fehlerbehandlung verbinden. Diese VIs können den Fehler beschreiben und Ihnen Optionen über das weitere Verfahren bieten, falls ein Fehler auftritt.

Einige Fehlercodes, die von den VIs für den seriellen Anschluß zurückgegeben werden, sind plattformspezifisch. Wenden Sie sich bitte an Ihre Systemdokumentation für eine Liste der Fehlercodes.

Anschlußnummer

Windows 95-NT und 3.*x*

Wenn Sie die VIs für den seriellen Anschluß unter Windows 95/NT und Windows 3.*x verwenden*, kann der Parameter **Anschlußnummer** folgende Werte annehmen:

0:	COM1	5:	COM6	10:	LPT1
1:	COM2	6:	COM7	11:	LPT2
2:	COM3	7:	COM8	12:	LPT3
3:	COM4	8:	COM9	13:	LPT4
4:	COM5				

Wenn Sie die VIs für den seriellen Anschluß unter Windows 95 oder Windows NT verwenden, ist der Parameter **Anschlußnummer** 0 für COM1, 1 für COM2 und so weiter.

Macintosh

Auf dem Macintosh ist Anschluß 0 das Modem und gebraucht die Treiber `.ain` und `.aout`. Anschluß 1 ist der Drucker und setzt die Treiber `.bin` und `.bout` ein. Um weitere Anschlüsse auf einem Macintosh zu erhalten, müssen Sie andere Karten mit den sie begleitenden Treibern installieren.

UNIX

Auf einer Sun SPARCstation unter Solaris 1 und auf Concurrent PowerMAX ist der Parameter **Anschlußnummer** in den VIs für den seriellen Anschluß 0 für `/dev/ttya`, 1 für `/dev/ttyb` usw. Unter Solaris 2 bezieht sich Anschluß 0 auf `/dev/cua/a`, 1 auf `/dev/cua/b` usw. Unter HP-UX bezieht sich Anschlußnummer 0 auf `/dev/tty00`, 1 auf `/dev/tty01` usw.

Auf Concurrent PowerMAX bezieht sich Anschluß 0 auf `/dev/console`, Anschluß 1 bezieht sich auf `/dev/tty1`, Anschluß 2 bezieht sich auf `/dev/tty2` usw.

Weil die seriellen Anschlußkarten von anderen Lieferanten beliebige Gerätenamen haben können, hat LabVIEW eine einfache Schnittstelle entwickelt, die die Numerierung von Anschlüssen vereinfacht. In LabVIEW für Sun, HP-UX und Concurrent PowerMAX besteht eine Konfigurationsoption, die LabVIEW instruiert, wie die seriellen Anschlüsse zu adressieren sind. LabVIEW unterstützt jede Karte, die standardmäßige UNIX-Geräte verwendet. Einige Hersteller schlagen für ihre Karte den Gebrauch von `cua`- statt `tty`-Geräteknoten vor. LabVIEW kann beide Knotentypen adressieren.

Die Datei `.labviewrc` enthält die LabVIEW-Konfigurationsoptionen. Um zu bestimmen, welche Geräte von den VIs für den seriellen Anschluß verwendet werden, passen Sie die Konfigurationsoption `labview.serialDevices` auf die Liste der von Ihnen zu gebrauchenden Geräten an.

Der Standard ist beispielsweise:

```
labview.serialDevices:/dev/ttya:/dev/ttyb:/dev/
ttyc:...:/dev/ttyz.
```

☞ **Hinweis** ***Dies erfordert, daß jede Installation einer seriellen Karte von dritter Seite eine Methode einschließt, mit der eine standardmäßige*** `dev` ***Datei (Knoten) erzeugt wird, und daß der Benutzer den Namen von jener Datei weiß.***

Teil III

Analyse

Dieser Teil des Handbuchs enthält grundsätzliche Informationen zur Analyse von Daten nach der Datenerfassung, zur Signalverarbeitung und Signalerzeugung, zu linearer Algebra, zur Kurvenanpassung, zu Wahrscheinlichkeitsberechnungen und zur Statistik.

Teil III, *Analyse*, enthält die folgenden Kapitel.

- Kapitel 11, *Einführung in die Analyse in LabVIEW*, bietet eine Einführung in die Konzepte, die allen Analyseanwendungen zugrundeliegen, einschließlich unterstützter Funktionalitäten, Notations- und Benennungskonventionen und Abtastsignalmethoden.
- Kapitel 12, *Signalerzeugung*, erläutert, wie Signale unter Verwendung normalisierter Frequenzen erzeugt werden und wie ein simulierter Funktionsgenerator angefertigt wird.
- Kapitel 13, *Digitale Signalverarbeitung*, beschreibt die Grundsätze von Fast Fourier Transform (FFT/Schnelle Fourier-Transformation) und Discrete Fourier Transform (DFT/Diskrete Fourier-Transformation) und beschreibt, wie FFT und DFT in der Spektrumanalyse verwendet werden.
- Kapitel 14, *Fensterglättung*, erläutert, wie durch Verwendung von Fenstern Leckeffekte verhindert werden und die Analyse erfaßter Signale verbessert wird.
- Kapitel 15, *Spektralanalyse und -messung*, zeigt, wie Sie das Amplituden- und Phasenspektrum bestimmen, einen Spektrumanalysator entwickeln und die harmonische Gesamtverzerrung THD (Total Harmonic Distortion) Ihrer Signale bestimmen.
- Kapitel 16, *Filterung*, erläutert, wie Sie mit IIR (Infinite Impulse Response)-, FIR (Finite Impulse Response)- und nichtlinearen Filtern unerwünschte Frequenzen aus Signalen herausfiltern können.

- Kapitel 17, *Kurvenanpassung*, zeigt, wie Informationen aus einem Datensatz extrahiert werden, um eine funktionale Beschreibung zu erhalten.
- Kapitel 18, *Lineare Algebra*, erläutert, wie Matrixberechnungen und -analysen durchgeführt werden.
- Kapitel 19, *Wahrscheinlichkeit und Statistik*, erläutert einige grundsätzliche Konzepte der Wahrscheinlichkeitsrechnung und Statistik und zeigt, wie diese Konzepte zur Lösung realer Probleme verwendet werden können.

11

Einführung in die Analyse in LabVIEW

Digitalsignale sind in unserer Umwelt allgegenwärtig. Telefonunternehmen benutzen Digitalsignale zur Darstellung der menschlichen Stimme. Radio-, Fernseh- und Hi-Fi-Systeme gehen allmählich in den digitalen Bereich über, aufgrund der Überlegenheit hinsichtlich der Wiedergabetreue, der Geräuschunterdrückung und der Signalverarbeitungsflexiblität. Daten werden von Satelliten zu Bodenstationen in digitaler Form gesendet. Die Bilder der NASA von fernen Planeten und dem Weltraum werden oft digital verarbeitet, um Geräusche zu entfernen und nützliche Information zu extrahieren. Wirtschaftsdaten, Zensusergebnisse und Börsennotierungen sind alle in digitaler Form erhältlich. Wegen der vielen Vorteile der digitalen Signalverarbeitung werden Analogsignale in digitale Form umgewandelt, bevor sie mit einem Computer verarbeitet werden.

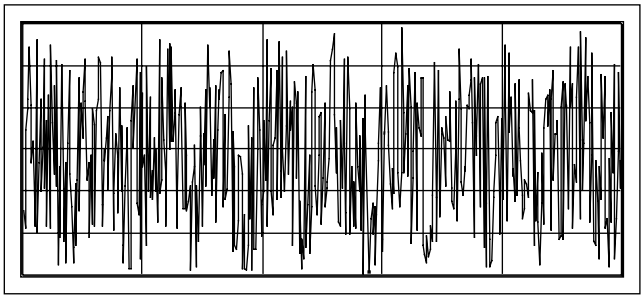
Dieses Kapitel vermittelt einen Hintergrund in den Grundlagen der digitalen Signalverarbeitung und eine Einführung in die LabVIEW Analysebibliothek, die aus Hunderten von VIs für Signalverarbeitung und -analyse besteht.

Die Bedeutung der Datenanalyse

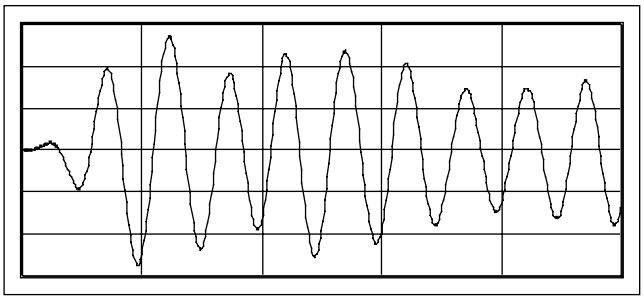
Moderne numerische und digitale Fließpunkt-Signalverarbeiter hoher Geschwindigkeit sind für Echtzeit- und Analysesysteme ständig wichtiger geworden. Einige der möglichen Anwendungen umfassen biomedische Datenverarbeitung, Sprachsynthese und -erkennung sowie digitale Ton- und Bildverarbeitung.

Die Bedeutung der Integration von Analysebibliotheken in technische Arbeitsstationen liegt darin, daß die *Roh*daten, wie in der folgenden Abbildung gezeigt, nicht immer nützliche Information vermittelt. Oft muß man das Signal umwandeln, Störgeräusche entfernen, durch fehlerhafte

Geräte korrumpierte Daten korrigieren oder Umwelteinflüsse wie Temperatur und Luftfeuchtigkeit ausgleichen.



Durch die Analyse und Verarbeitung der digitalen Daten können Sie die nützlichen Informationen von dem Geräusch trennen und in einer Form darstellen, die verständlicher als die Rohdaten ist. Die folgende Abbildung zeigt die verarbeiteten Daten.



Der Blockdiagramm-Programmieransatz von LabVIEW und der umfangreiche Satz von Analyse-VIs in LabVIEW vereinfachen die Entwicklung von Analyseanwendungen.

Die Analyse-VIs von LabVIEW bieten Ihnen die neuesten Techniken der Datenanalyse durch den Gebrauch von VIs, die Sie miteinander verbinden können. Anstatt sich um die Implementierung von Details der Analyseroutinen zu kümmern, wie Sie es in konventionellen Programmiersprachen tun, können Sie sich auf die Lösung von Ihren Datenanalyseproblemen konzentrieren.

Full Development System

Die grundlegende Bibliothek von Analyse-VIs ist eine Untermenge der fortgeschrittenen Bibliothek von Analyse-VIs. Die grundlegende Analysebibliothek enthält VIs für die statistische Analyse, Linearalgebra, und die numerische Analyse. Die fortgeschrittene Analysebibliothek enthält weitere VIs in den obigen Gebieten, sowie VIs für die Signalerzeugung, Algorithmen für den Zeit- und Frequenzbereich, die Fensterroutine, digitale Filter, Auswertungen und Regressionen.

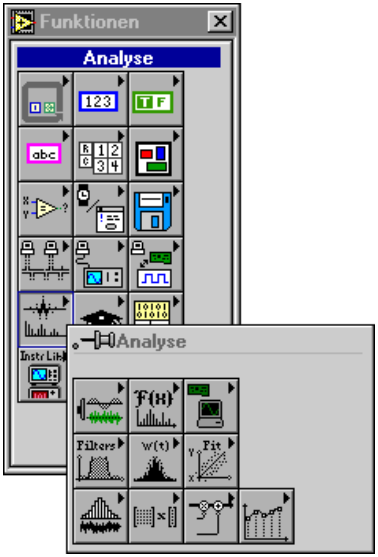
Wenn die VIs in der grundlegenden Analysebibliothek Ihren Anforderungen nicht entsprechen, dann können Sie die LabVIEW Advanced Analysebibliotheken dem LabVIEW Base Package hinzufügen. Sobald sie aufrüsten, haben sie alle in dem Full Development System enthaltenen Analysewerkzeuge zur Verfügung.

Übersicht über die Analyse-VIs

Wenn das Analogsignal durch den A/D Wandler (ADC) in digitale Form umgewandelt wurde und in Ihrem Computer als Digitalsignal (eine Menge von Abtastwerten) vorliegt, werden Sie normalerweise diese Abtastwerte irgendwie verarbeiten wollen. Zweck der Verarbeitung könnte die Feststellung der Charakteristiken des Systems, von dem die Abtastwerte genommen wurden, die Messung von bestimmten Eigenschaften des Signals oder die Umwandlung der Abtastwerte in eine für den menschlichen Verstand geeignete Form sein.

Die Analysebibliothek enthält VIs zur Durchführung von umfangreicher numerischer Analyse, zur Signalerzeugung und -verarbeitung, zur Kurvenanpassung, zur Messung sowie für andere Analysefunktionen. Die Analysebibliothek, im vollen LabVIEW Entwicklungssystem enthalten, ist eine Schlüsselkomponente im Aufbau von einem virtuellen Instrumentationssystem. Sie enthält nicht nur die in vielen Mathematikpaketen vorhandene analytische Funktionalität, sondern weist auch viele einzigartige Signalverarbeitungs- und Messungsfunktionen auf, die spezifisch für die Instrumentationsbranche ausgelegt sind.

Die Analyse-VIs von LabVIEW sind in der Subpalette **Analyse** der Palette **Funktionen** in LabVIEW oder BridgeVIEW erhältlich.



Es gibt 10 Analyse-VI-Bibliotheken. Die Hauptkategorien sind:

Signalerzeugung: VIs, die digitale Muster und Wellenformen erzeugen.

Digitale Signalverarbeitung: VIs, die Umwandlungen im Frequenzbereich, Analyse im Frequenzbereich, Analyse im Zeitbereich und andere Transformationen wie die Hartley- und Hilbert-Transformationen durchführen.

Messungsfunktionen: VIs, die messungsorientierte Funktionen wie einseitige Spectren, skalierte Fensterung und Spitzenleistungs- und Frequenzabschätzungen durchführen.

Digitale Filter: VIs, die IIR-, FIR- und nichtlineare digitale Filterfunktionen durchführen.

Fensterung-Funktionen: VIs, die Daten-Fensterung durchführen.

Wahrscheinlichkeits- und Statistikfunktionen: VIs, die deskriptive Statistikfunktionen, wie das Identifizieren des Mittelwerts oder der Standardabweichung von einem Datensatz durchführen, sowie auch inferenzielle Statistikfunktionen für die Wahrscheinlichkeit und die Analyse der Varianz (ANOVA).

Kurvenanpassungsfunktionen: VIs, die Kurvenanpassungen und Interpolationen durchführen.

Linearalgebra-Funktionen: VIs, die algebraische Funktionen für reale und komplexe Vektoren und Matrizen durchführen.

Array-Operationen: VIs, die übliche ein- und zweidimensionale numerische Array-Operationen, wie lineare Auswertung und Skalierung, durchführen.

Weitere numerische Methoden: VIs, die numerische Methoden zur Durchführung von Wurzelfindung, numerischer Integration und Spitzenerfassung einsetzen.

In diesen Kapiteln werden Sie lernen, wie VIs aus der Analysebibliothek zum Aufbau eines Funktionserzeugers und eines einfachen, aber praktischen Spektrumsanalysators eingesetzt werden können. Sie werden auch die Benutzung von Digitalfiltern, den Zweck von Fensterung und die Vorteile der verschiedenen Fenstertypen, die Durchführung von einfachen Kurvenanpassungsaufgaben und vieles mehr lernen. Die Aktivitäten in diesen Kapiteln erfordern das volle LabVIEW/BridgeVIEW Entwicklungssystem. Für fortgeschrittenere Benutzer ist ein ausführlicher Satz von Beispielen, die den Gebrauch von den Analyse-VIs demonstrieren, in dem Ordner `labview\examples\analysis` zu finden.

Über die Analysebibliothek hinaus bietet National Instruments viele zusätzliche Analyseprodukte, die LabVIEW zu einem der leistungfähigsten Analyse-Softwarepakete auf dem Markt machen. Diese Zusätze umfassen das *Joint Time-Frequency Analysis Toolkit*, das den preisgekrönten Gabor Spectrogramm-Algorithmus von National Instruments enthält, mit dem man Zeit-Frequenzeigenschaften analysieren kann, die durch konventionelle Fourieranalyse nicht ohne weiteres zu erfassen sind; das *G Math Toolkit*, das erweiterte mathematische Funktionalität bietet, wie z.B. einen Formeluntersucher, Routinen für die Optimierung und Lösung von differentiellen Gleichungen, viele Sorten von 2D - und 3D-Kurven; das *Digital Filter Design Toolkit*; und viele andere.

Schreib- und Benennungskonventionen

Um Ihnen zu helfen, den Typ von Parameter und Operationen zu identifizieren, benutzt dieser Abschnitt des Handbuchs die folgenden Schreib- und Benennungskonventionen, soweit nicht anders in einer VI-Beschreibung angegeben. Obwohl es einige Skalarfunktionen und -operationen gibt, verarbeiten die meisten Analyse-VIs große Datenblöcke in der Form von eindimensionalen Arrays (oder Vektoren) und zweidimensionale Arrays (oder Matrizen).

Normale Kleinbuchstaben stellen Skalargrößen oder Konstanten dar. Zum Beispiel:

$$a,$$
$$\pi,$$
$$b = 1{,}234.$$

Großbuchstaben repräsentieren Arrays. Zum Beispiel:

$$X,$$
$$A,$$
$$Y = a\,X + b.$$

Im allgemeinen bezeichnen *X* und *Y* 1D-Arrays, und *A*, *B* und *C* stellen Matrizen dar.

Arrayindizes in LabVIEW gehen von Null aus. Der Index des ersten Elements im Array, ohne Rücksicht auf seine Dimension, ist Null. Die folgende Zahlensequenz stellt ein 1D-Array dar, das *n* Elemente enthält.

$$X = \{x_0, x_1, x_2, ..., x_{n-1}\}$$

Die folgende Skalargröße repräsentiert das i^{te} Element der Sequenz *X*.

$$x_i, \quad 0 \le i < n$$

Das erste Element in der Sequenz ist x_0 und das letzte Element in der Sequenz ist x_{n-1}, also insgesamt *n* Elemente.

Die folgende Zahlensequenz stellt ein 2D-Array dar, das *n* Reihen und *m* Spalten enthält.

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & \cdots & a_{0m-1} \\ a_{10} & a_{11} & a_{12} & \cdots & a_{1m-1} \\ a_{20} & a_{21} & a_{22} & \cdots & a_{2m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-10} & a_{n-11} & a_{n-2} & \cdots & a_{n-1m-1} \end{bmatrix}$$

Die Gesamtzahl der Elemente im 2D-Array ist das Produkt von *n* und *m*. Der erste Index entspricht der Reihennummer und der zweite Index entspricht der Spaltennummer. Die folgende Skalargröße stellt das Element in der i^{ten} Reihe und der j^{ten} Spalte dar.

$$a_{ij},\ 0 \le i < n \text{ und } 0 \le j < m$$

Das erste Element in *A* ist $a_{0\,0}$ und das letzte Element ist $a_{n-1\,m-1}$.

Soweit nicht anders angegeben, verwendet das vorliegende Handbuch die folgenden vereinfachten Schreibweisen für Array-Operationen.

Das Setzen der Elemente eines Arrays gleich einer Skalarkonstante ist durch

$$X = a$$

dargestellt, was der Sequenz

$$X = \{a, a, a, \ldots, a\}$$

entspricht und statt

$$x_i = a$$

für

$$i = 0, 1, 2, \ldots, n-1$$

benutzt wird.

Multiplizieren der Elemente eines Arrays mit einer Skalarkonstante wird durch

$$Y = a\,X$$

dargestellt, was der Sequenz

$$Y = \{a\,x_0, a\,x_1, a\,x_2, \ldots, a\,x_{n-1}\}$$

entspricht und statt

$$y_i = a\,x_i$$

für

$$i = 0, 1, 2, \ldots, n-1$$

verwendet wird.

Multiplizieren eines 2D-Arrays mit einer Skalarkonstante wird durch

$$B = k\,A$$

dargestellt, was der Sequenz

$$B = \begin{bmatrix} ka_{00} & ka_{01} & ka_{02} & \ldots & ka_{0m-1} \\ ka_{10} & ka_{11} & ka_{12} & \ldots & ka_{1m-1} \\ ka_{20} & ka_{21} & ka_{22} & \ldots & ka_{2m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ ka_{n-10} & ka_{n-11} & ka_{n-12} & \ldots & ka_{n-1m-1} \end{bmatrix}$$

entspricht und statt

$$b_{ij} = k\,a_{ij}$$

für

$$i = 0, 1, 2, \ldots, n-1$$

und

$$j = 0, 1, 2, \ldots, m-1$$

benutzt wird. Ein Array mit keinen Elementen ist ein Leerarray und wird so dargestellt

$$\text{Leer} = \text{NULL} = \emptyset = \{\ \}.$$

Im allgemeinen führen Operationen auf Leerarrays zu leeren Ausgangsarrays oder zu undefinierten Ergebnissen.

Datenabtastung

Abtastsignale

Um digitale Signalverarbeitungstechniken anzuwenden, muß man zuerst ein Analogsignal in seine digitale Repräsentation umwandeln. In der Praxis wird dies durch einen analog/digitalen (A/D) Wandler implementiert. Betrachten wir ein Analogsignal $x(t)$, das alle Δt Sekunden abgetastet wird. Das Zeitintervall Δt heißt das *Abtastintervall* oder die *Abtastperiode*. Sein Reziprokwert, $1/\Delta t$, heißt die *Abtastfrequenz*, in Einheiten von Abtastwerte/Sekunde. Jeder der diskreten Werte von x(t) zu $t = 0, \Delta t, 2\Delta t, 3\Delta t$ usw. heißt ein *Abtastwert*. Also, $x(0), x(\Delta t), x(2\Delta t)$, ..., sind alle Abtastwerte. Das Signal $x(t)$ kann daher als eine diskrete Menge von Abtastwerten repräsentiert werden:

$$\{x(0), x(\Delta t), x(2\Delta t), x(3\Delta t), \ldots, x(k\Delta t), \ldots\}.$$

Die unten dargestellte Abbildung 11-1 zeigt ein Analogsignal und dessen entsprechende abgetastete Version. Das Abtastintervall ist Δt. Beachten Sie, daß die Abtastwerte zu diskreten Zeitpunkten definiert sind.

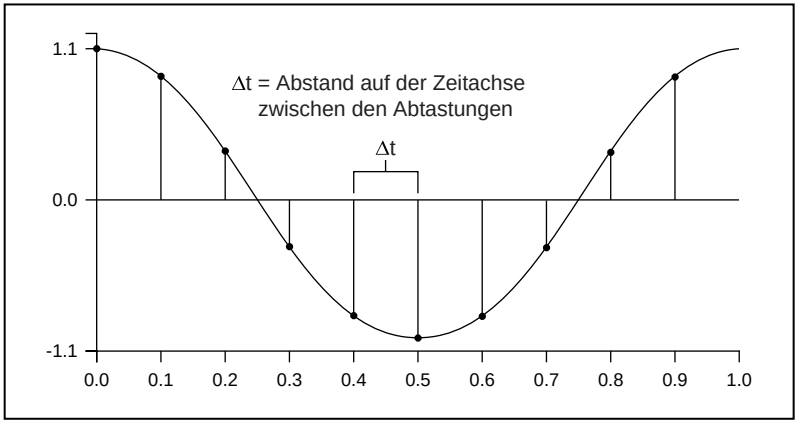


Abbildung 11-1. Analogsignal und die entsprechende abgetastete Version

Die folgende Schreibweise repräsentiert die einzelnen Abtastwerte:

$$x[i] = x(i\Delta t)$$

für

$$i = 0, 1, 2, \ldots$$

Wenn N Abtastwerte vom Signal $x(t)$ gewonnen werden, dann läßt sich $x(t)$ durch die Sequenz

$$X = \{x[0], x[1], x[2], x[3], \ldots, x[N-1]\}$$

darstellen. Dies heißt die *digitale Repräsentation* oder die *abgetastete Version* von $x(t)$. Beachten Sie, daß die Sequenz $X = \{x[i]\}$ auf der Integer-Variable i indexiert ist und keine Information über die Abtastrate enthält. Also, lediglich durch die Kenntnis der in X enthaltenen Abtastwerte können Sie nicht wissen, was die Abtastrate ist.

Berücksichtigungen bei der Abtastung

A/D-Wandler (ADC) sind ein integraler Bestandteil der DAQ-Karten von National Instruments. Einer der wichtigsten Parameter von einem analogen Eingabesystem ist die Rate, mit der die DAQ-Karte ein einkommendes Signal abtastet. Die Abtastrate bestimmt, wie oft eine analog-digitale (A/D) Wandlung stattfindet. Eine schnelle Abtastrate erfaßt mehr Punkte in einer gegebenen Zeit und kann deshalb oft eine bessere Repräsentation des Originalsignals bilden. Zu langsames Abtasten kann zu einer schlechten Repräsentation des Analogsignals führen. Die Abbildung 11-2 zeigt ein adäquat abgetastetes Signal, so wie auch die Auswirkungen von unzureichender Abtastung. Die Auswirkung von unzureichender Abtastung ist, daß das Signal so aussieht, als ob es eine andere als seine tatsächliche Frequenz hätte. Diese Mißrepräsentation von einem Signal heißt ein *Alias*.

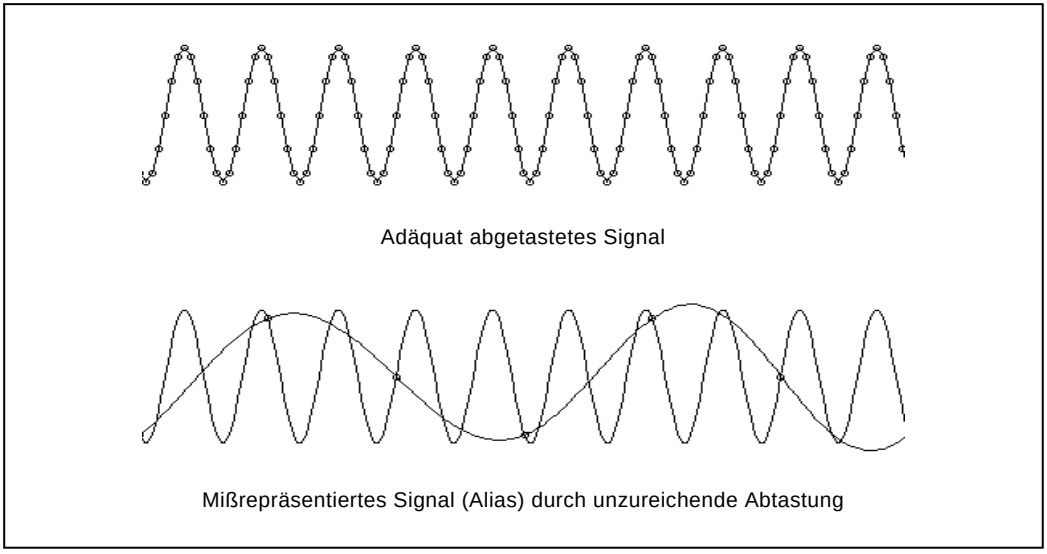


Abbildung 11-2. Aliasing-Wirkung von unrichtiger Abtastrate

Gemäß dem *Satz von Nyquist* muß man, um Aliasing zu vermeiden, mit einer Rate abtasten, die zweimal größer als die höchstfrequente Komponente des zu erfassenden Signals ist. Für eine gegebene Abtastrate ist die höchste Frequenz, die genau, also ohne Aliasing, wiedergegeben werden kann, als die Nyquist-Frequenz bekannt. Die Nyquist-Frequenz ist die halbe Abtastfrequenz. Signale mit Frequenzkomponenten über der Nyquist-Frequenz werden mit einem Alias zwischen DC und der Nyquist-Frequenz erscheinen. Die Aliasfrequenz ist der Betrag der

Differenz zwischen der Frequenz des Eingangsignals und dem nächsten Integer-Vielfachen der Abtastrate. Die Abbildungen 11-3 und 11-4 veranschaulichen dieses Phänomen. Nehmen wir beispielsweise an, daß *f*s,die Abtastfrequenz, 100 Hz beträgt. Nehmen wir weiterhin an, daß das Eingangssignal die folgenden Frequenzen enthält—25 Hz, 70 Hz, 160 Hz und 510 Hz. Diese Frequenzen sind in Abbildung 11-3 dargestellt.

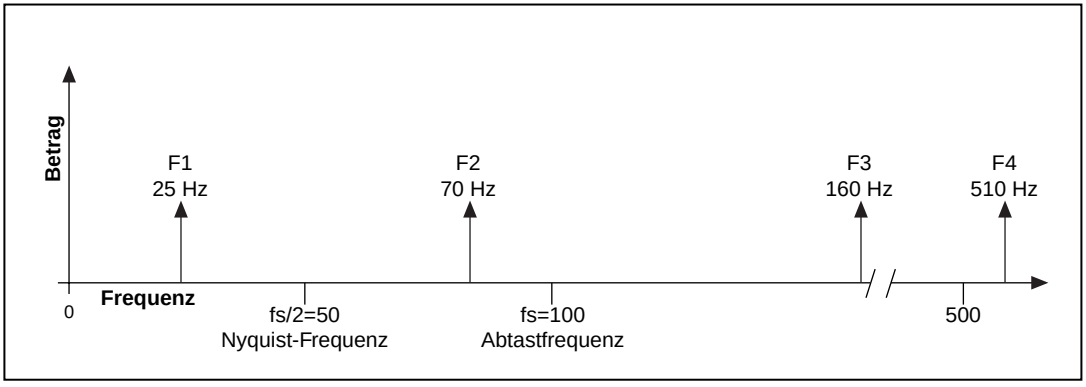


Abbildung 11-3. Eigentliche Signalfrequenz-Komponenten

In der Abbildung 11-4 sehen wir, daß Frequenzen unterhalb der Nyquist-Frequenz (*f*s/2=50 Hz) richtig abgetastet werden. Frequenzen oberhalb der Nyquist-Frequenz erscheinen als Aliases. Zum Beispie: F1 (25 Hz) erscheint auf der richtigen Frequenz, aber F2 (70 Hz), F3 (160 Hz) und F4 (510 Hz) haben Aliases auf 30 Hz, 40 Hz und 10 Hz. Zur Kalkulation der Aliasfrequenz wird die folgende Gleichung verwendet:

Aliasfreq. = ABS (Nächstes Integer-Vielfache
der Abtastfreq. – Eingangsfreq.)

wobei ABS “den Betrag” bedeutet. Zum Beispiel:

Alias F2 = |100 – 70| = 30 Hz
Alias F3 = |(2)100 – 160| = 40 Hz
Alias F4 = |(5)100 – 510| = 10 Hz

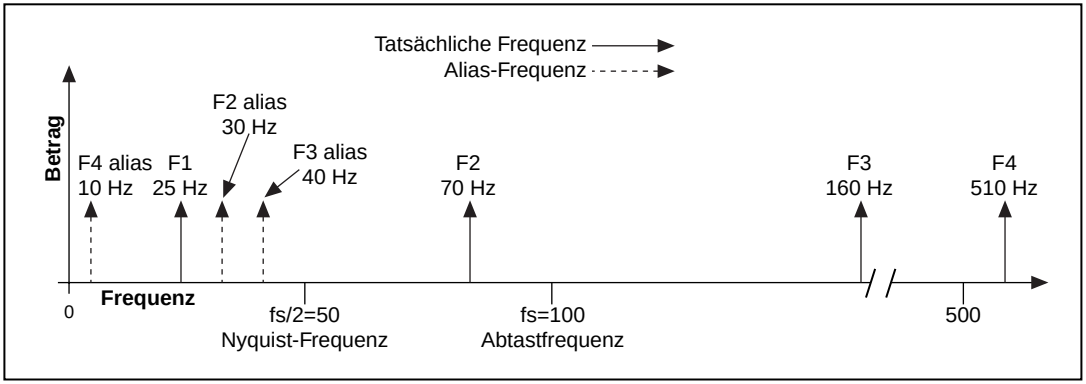


Abbildung 11-4. Signalfrequenz-Komponenten und Aliases

Eine häufig gestellte Frage lautet, “Wie schnell sollte ich abtasten?”. Ihr erster Gedanke ist vielleicht, daß Sie auf der höchsten Rate abtasten, die auf Ihrer DAQ-Karte verfügbar ist. Wenn Sie aber sehr schnell über längere Zeiträume hinweg abtasten, haben Sie vielleicht nicht genug Arbeitsspeicher oder Festplattenspeicherplatz vorhanden, um die Daten zu fassen. Die Abbildung 11-5 zeigt die Auswirkungen von verschiedenen Abtastraten. Im Fall a wird die Sinuswelle mit Frequenz f auf der gleichen Frequenz fs (Abtastwerte/s) = f (Zyklen/s) abgetastet, oder auf 1 Abtastwert pro Zyklus. Die rekonstruierte Wellenform erscheint als ein Alias, nämlich mit der Frequenz DC. Wenn man die Abtastrate auf 7 Abtastwerte/4 Zyklen erhöht, wie im Fall b, nimmt die Wellenform an Frequenz zu, erscheint aber als Alias mit einer Frequenz, die kleiner ist als das Originalsignal (3 Zyklen statt 4). Die Abtastrate im Fall b ist fs = 7/4 f. Wenn man die Abtastrate auf fs = 2f erhöht, hat die digitalisierte Wellenform die richtige Frequenz (dieselbe Anzahl von Zyklen) und läßt sich als die ursprüngliche Welle rekonstruieren, wie im Fall c gezeigt. Für die Verarbeitung im Zeitbereich kann es wichtig sein, die Abtastrate so zu erhöhen, daß die Abtastwerte das ursprüngliche Signal genauer repräsentieren. Durch ein Erhöhung der Abtastrate auf weit über f, sagen

wir fs = 10f oder 10 Abtastwerte/Zyklus, kann man die Wellenform genau wiedergeben, wie im Fall d gezeigt.

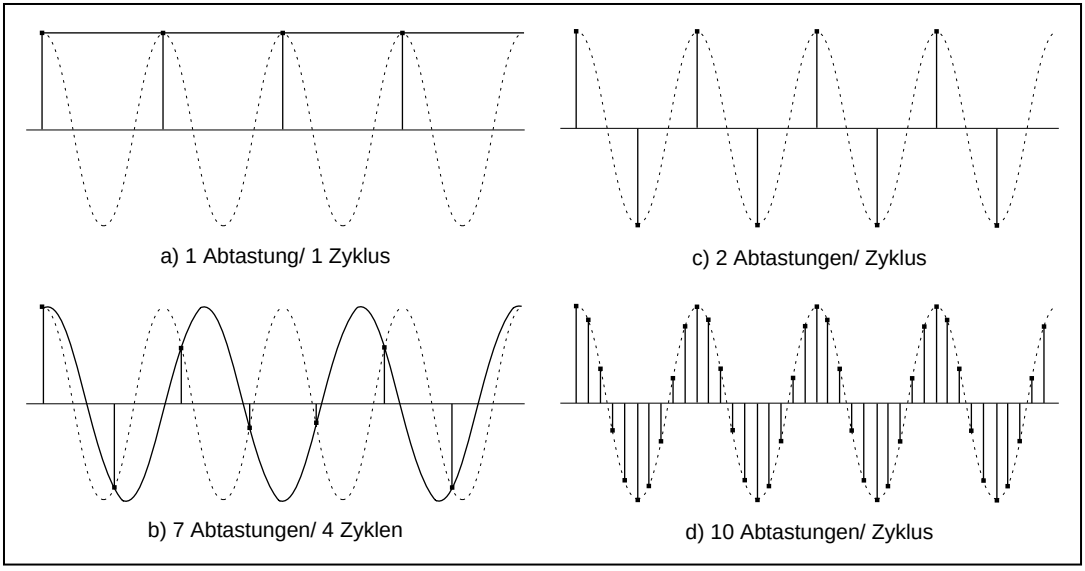


Abbildung 11-5. Auswirkungen der Abtastung auf verschiedenen Raten

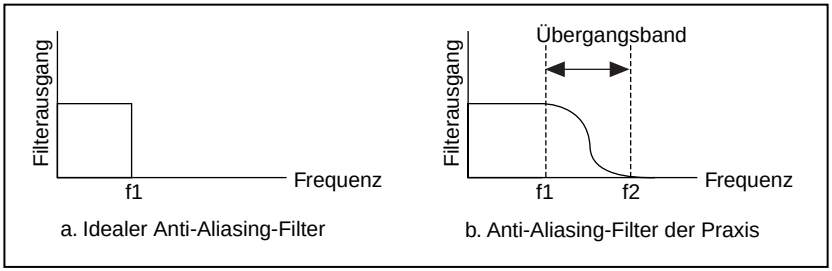
Warum braucht man Anti-Aliasing-Filter?

Wir haben gesehen, daß die Abtastrate mindestens das Zweifache der Maximalfrequenz im abgetasteten Signal sein sollte. In anderen Worten, die Maximalfrequenz des Eingangsignals sollte weniger als oder gleich der Hälfte der Abtastrate sein. Wie aber kann man sicherstellen, daß dies in der Praxis definitiv der Fall ist? Selbst wenn man sicher ist, daß das gemessene Signal in seiner Frequenz eine Obergrenze hat, könnte der Empfang von Streusignalen (wie z.B. die Netzfrequenz oder Signale von örtlichen Rundfunksendern) Frequenzen über die Nyquist-Frequenz enthalten. Diese Frequenzen können dann durch Aliasing in den gewünschten Frequenzbereich fallen und so zu fehlerhaften Ergebnissen führen.

Um ganz sicher zu sein, daß der Frequenzinhalt des Eingangsignals begrenzt ist, wird ein Tiefpaßfilter (ein Filter, der niedrige Frequenzen durchläßt, aber hohe Frequenzen dämpft) dem Abtaster and dem ADC vorgeschaltet. Dieser Filter heißt ein *Anti-Aliasing*-Filter, weil er durch Dämpfung der höheren Frequenzen (über der Nyquist-Frequenz) verhindert, daß die Aliasing verursachenden Komponenten abgetastet werden. Da wir uns in dieser Stufe (nämlich vor dem Abtaster und dem

ADC) noch immer in der Analogwelt befinden, ist der Anti-Aliasing-Filter auch ein Analogfilter.

Ein idealer Anti-Aliasing-Filter ist wie in der Abbildung (a) unten dargestellt.



Er läßt alle gewünschten Eingangsfrequenzen (unter *f1*) durch und sperrt alle unerwünschten Frequenzen (über *f1*). Ein solcher Filter ist aber nicht physikalisch zu verwirklichen. In der Praxis sehen Filter so aus, wie in der Abbildung (b) oben gezeigt. Sie lassen alle Frequenzen < *f1* durch und sperren alle Frequenzen > *f2*. Der Bereich zwischen *f1* und *f2* heißt das *Übergangsband*, in dem eine allmähliche Dämpfung der Eingangsfrequenzen enthalten ist. Obwohl man nur Signale mit Frequenzen < *f1* durchlassen möchte, könnten diese Signale im Übergangsband noch immer Aliasing verursachen. Deshalb sollte die Abtastfrequenz mehr als zweimal größer als die höchste Frequenz im Übergangsband sein. Diese Abtastfrequenz ist im Ergebnis mehr als das Zweifache der maximalen Eingangsfrequenz (*f1*). Das ist einer der Gründe, warum die Abtastrate über zweimal größer als die maximale Eingangsfrequenz ist. Wir werden in einem späteren Abschnitt sehen, wie das Übergangsband des Filters vom zu konstruierenden Filtertyp abhängt.

Warum Dezibel benutzen?

Bei manchen Instrumenten werden Sie die Option sehen, die Amplitude in einer Linear- oder Dezibelskala (dB) anzuzeigen. Die Linearskala zeigt die Amplituden so wie sie sind, während die Dezibelskala eine Transformation der Linearskala in eine logarithmische Skala darstellt. Wie werden sehen, warum diese Transformation benötigt wird.

Nehmen wir an, Sie wollen ein Signal mit sehr großen sowie auch sehr kleinen Amplituden anzeigen. Es sei ferner angenommen, Sie haben eine 10 cm-hohe Anzeige und Sie wollen die ganze Höhe der Anzeige für die höchste Amplitude ausnützen. Wenn also die größte Amplitude im Signal 100 V ist, entspricht eine Höhe von 1 cm an der Anzeige 10 V. Ist die

kleinste Amplitude im Signal 0.1 V, so entspricht dies einer Höhe von nur 0.1 mm. An der Anzeige ist das kaum zu sehen.

Um alle Amplituden, von den kleinsten bis zu den größten, zu sehen, müssen Sie die Amplitudenskala ändern. Alexander Graham Bell erfand eine Einheit, das Bell, das logarithmisch ist und daher große Amplituden komprimiert und kleine Amplituden erweitert. Das Bell war aber eine viel zu große Einheit, weshalb üblicherweise das Dezibel (1/10 Bell) gebraucht wird. Das Dezibel (dB) ist definiert als

$$\text{ein dB} = 10 \log_{10} (\text{Leistungsverhältnis}) = 20 \log_{10} (\text{Spannungsverhältnis})$$

Die folgende Tabelle zeigt die Beziehung zwischen dem Dezibel und den Leistungs- und Spannungsverhältnissen.

dB	Leistungs-verhältnis	Spannungs-verhältnis
+40	10000	100
+20	100	10
+6	4	2
+3	2	1,4
0	1	1
–3	1/2	1/1,4
–6	1/4	1/2
–20	1/100	1/10
–40	1/10000	1/100

Sie sehen also, daß die dB-Skala zum Komprimieren eines breiten Bereichs von Amplituden in eine kleine Menge von Zahlen nützlich ist. Die Dezibelskala wird häufig in der Schall- und Schwingungsmessung sowie zur Anzeige von Information aus dem Frequenzbereich eingesetzt.

12

Signalerzeugung

Dieses Kapitel erklärt, wie Signale unter Verwendung normalisierter Frequenz erzeugt werden und wie ein simulierter Funktionsgenerator gebaut wird. Für Beispiele darüber, wie die Signalerzeugungs-VIs verwendet werden, sehen Sie sich bitte die in `examples\analysis\sigxmpl.llb`. befindlichen Beispiele an.

Sie werden lernen, wie die VIs in der Analysebibliothek zur Erzeugung von vielen verschiedenen Signalarten eingesetzt werden. Einige Anwendungen für die Signalerzeugung sind:

- Das Simulieren von Signalen, um Ihren Algorithmus zu testen, wenn wirkliche Signale nicht zur Verfügung stehen (z.B. wenn eine DAQ-Karte zur Erfassung von wirklichen Signalen fehlt oder der Zugang zu wirklichen Signalen sonstwie unmöglich ist).
- Die Erzeugung von Signalen zum Anlegen an einen D/A-Wandler.

Normalisierte Frequenz

In der Analogwelt wird eine Signalfrequenz in Hz oder Zyklen pro Sekunde gemessen. Das digitale System benutzt aber oft eine Frequenz, nämlich das Verhältnis zwischen der Analogfrequenz und der Abtastfrequenz:

Digitalfrequenz = Analogfrequenz / Abtastfrequenz

Diese Digitalfrequenz heißt die *normalisierte* Frequenz. Seine Einheiten sind Zyklen/Abtastwert.

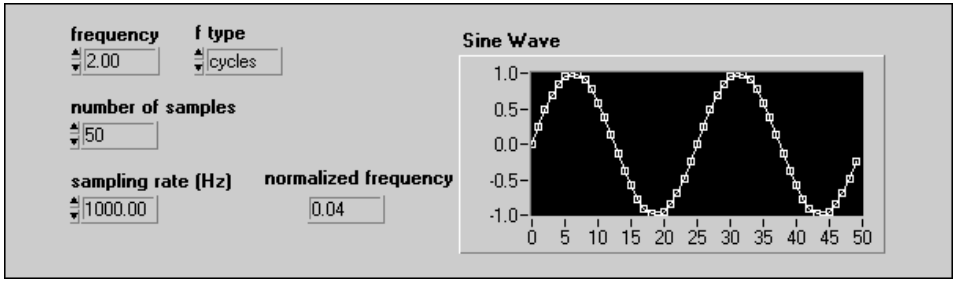
Einige der Signalerzeugungs-VIs verwenden ein Eingangsfrequenz-Bedienelement, *f*, von dem angenommen wird, daß es *normalisierte Frequenz*-Einheiten von *Zyklen pro Abtastwert* benutzt. Diese Frequenz bewegt sich zwischen 0,0 und 1,0, was einem realen Frequenzbereich von 0 bis zur Abtastfrequenz *fs* entspricht. Diese Frequenz schlingt sich auch um 1,0, so daß eine normalisierte Frequenz von 1,1 eine von 0,1 äquivalent ist. Wenn ein Signal beispielsweise mit der Nyquistrate (fs/2) abgetastet wird, bedeutet das, daß es zweimal pro Zyklus abgetastet wird (d.h., mit zwei Abtastwerten/Zyklus). Dies entspricht einer normalisierten Frequenz von 1/2 Zyklus/Abtastwert = 0,5 Zyklus/Abtastwert. Der Reziprokwert der

normalisierten Frequenz, *1/f*, ergibt die Zahl der Abtastungen vom Signal in einem Zyklus.

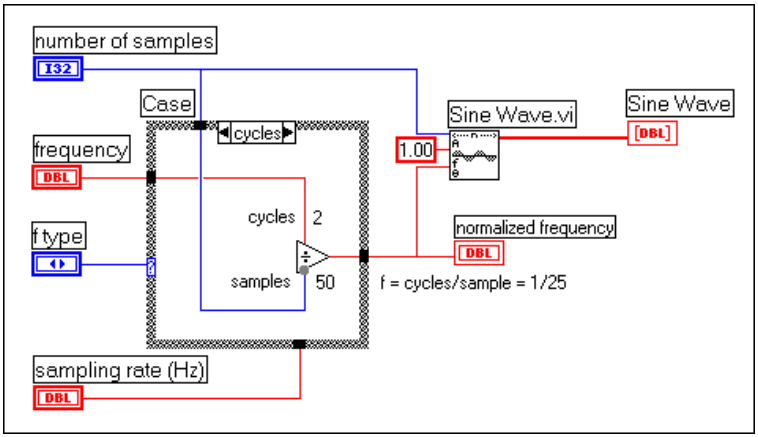
Wenn Sie ein VI benutzen, das die normalisierte Frequenz als Eingabe braucht, müssen Sie Ihre Frequenzeinheiten in die normalisierte Einheiten Zyklen/Abtastwert umwandeln. Sie müssen diese normalisierte Einheiten mit den folgenden VIs benutzen:

- Sinus-Schwingung
- Rechteck-Schwingung
- Sägezahn-Schwingung
- Dreieck-Schwingung
- Willkürliche Schwingung
- Chirp-Muster

Wenn Sie gewohnt sind, mit Frequenzeinheiten von Zyklen zu arbeiten, können Sie Zyklen in Zyklen/Abtastwert umwandeln, indem Sie die Zyklen durch die Anzahl der erzeugten Abtastwerte dividieren. Die folgende Abbildung zeigt das VI **Sinus-Schwingung** das zur Erzeugung von zwei Zyklen von einer Sinus-Schwingung eingesetzt wird.



Die folgende Abbildung zeigt das Blockdiagramm für die Umwandlung von Zyklen in Zyklen/Abtastwert.

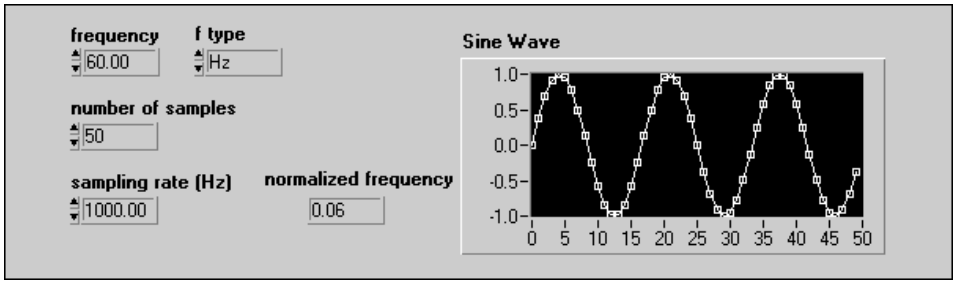


Sie brauchen nur die Frequenz (in Zyklen) durch die Anzahl von Abtastwerten zu teilen. Im obigen Beispiel wird die Frequenz von 2 Zyklen durch 50 Abtastwerte geteilt, was eine normalisierte Frequenz von f = 1/25 Zyklen/Abtastwert ergibt. Dies bedeutet, daß 25 (Reziprokwert von f) Abtastwerte nötig sind, um einen Zyklus der Sinus-Schwingung zu erzeugen.

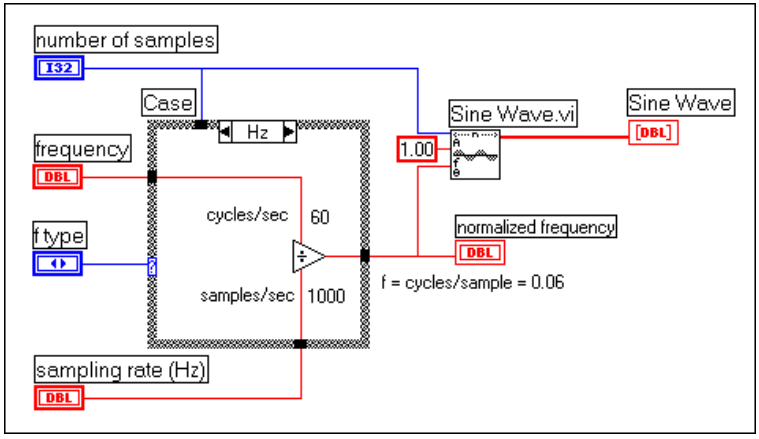
Es kann aber auch sein, daß Sie Frequenzeinheiten von Hz (Zyklen/s) benutzen müssen. Wenn Sie von Hz (oder Zyklen/s) in Zyklen/Abtastwert umwandeln wollen, teilen Sie Ihre Frequenz in Zyklen/s durch die Abtastrate, angegeben in Abtastwerte/s.

$$\frac{\text{Zyklen/s}}{\text{Abtastwerte/s}} = \frac{\text{Zyklen}}{\text{Abstastwert}}$$

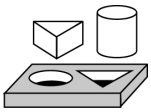
Die folgende Abbildung zeigt das VI **Sinus-Schwingung**, das zur Erzeugung von einem Sinussignal auf 60 Hz eingesetzt wird.



Die folgende Abbildung zeigt ein Blockdiagramm, das zur Erzeugung von einem Hertzschen Sinussignal verwendet wird. Man teilt die Frequenz von 60 Hz durch die Abtastrate von 1000 Hz, um die *normalisierte* Frequenz von *f* = 0,06 Zyklen/Abtastwert zu erhalten. Zur Erzeugung von einem Zyklus der Sinus-Schwingung sind daher fast 17 (1/0,06) Abtastwerte nötig.



Die Signalerzeugungs-VIs erschaffen viele häufige Signale, die für die Netzwerkanalyse und -simulation erforderlich sind. Sie können die Signalerzeugungs-VIs im Zusammenhang mit der Hardware von National Instruments zur Erzeugung von Analog-Ausgangssignale verwenden.

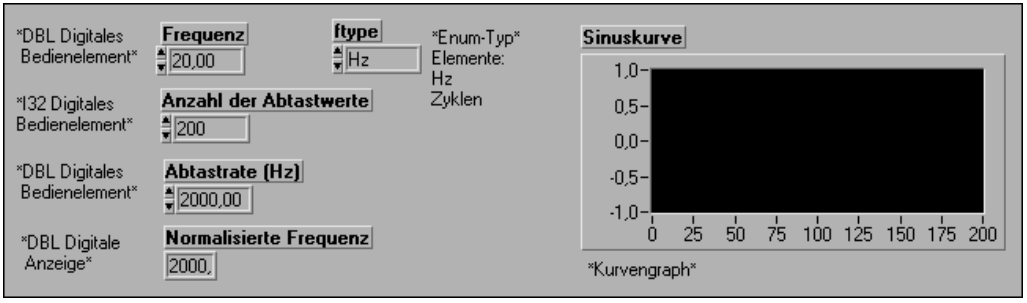


Übung 12-1. Mehr über die normalisierte Frequenz lernen

Ihr Übungsziel ist es, mehr über die normalisierte Frequenz zu lernen, indem Sie die Frequenz, die Abtastrate und die Anzahl der Abtastwerte einstellen und die Auswirkung davon auf die Sinus-Schwingung beobachten.

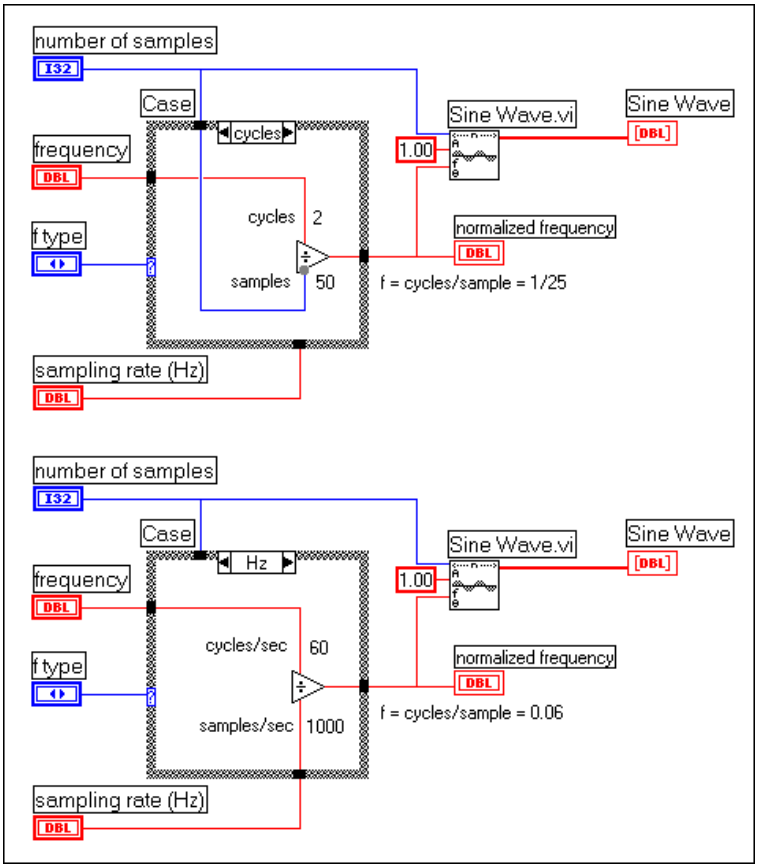
Frontpanel

1. Öffnen Sie ein neues Frontpanel, und erstellen Sie die Objekte wie in der folgenden Abbildung dargestellt.



Blockdiagramm

2. Erstellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm.

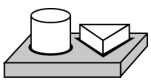


VI Sinus-Schwingung (Palette **Analyse»Signalerzeugung**).

3. Speichern Sie das VI als `Normalized Frequency.vi` im Verzeichnis `LabVIEW\Activity`.
4. Wählen Sie eine Frequenz von 2 Zyklen, **Frequenz** = 2 und **ftype** = Zyklen) und **Anzahl der Abtastwerte** = 100. Führen Sie das VI aus. Beachten Sie, daß die Kurve 2 Zyklen zeigen wird.
5. Erhöhen Sie die **Anzahl der Abtastwerte** auf 150, 200 und 250. Wieviele Zyklen sehen Sie?

6. Halten Sie jetzt die **Anzahl der Abtastwerte** = 100. Erhöhen Sie die Anzahl von Zyklen auf 3, 4 und 5. Wieviele Zyklen sehen Sie?

   Wenn Sie demnach die Frequenz in Zyklen auswählen, werden genausoviele Zyklen der Eingangs-Kurvenform auf der Kurve angezeigt. Beachten Sie, daß die Abtastrate in diesem Fall irrelevant ist.
7. Ändern Sie **ftype** auf Hz und **Abtastrate (Hz)** auf 1,000.
8. Indem Sie die **Anzahl der Abtastwerte** auf 100 fixiert halten, ändern Sie die **Frequenz** auf 10, 20, 30 und 40. Wieviele Zyklen der Kurvenform sehen Sie in jedem Fall auf der Kurve? Erklären Sie Ihre Beobachtungen.
9. Wiederholen Sie den obigen Schritt, indem Sie die **Frequenz** auf 10 fixiert halten, und ändern Sie die **Anzahl der Abtastwerte** auf 100, 200, 300 und 400. Wieviele Zyklen der Kurvenform sehen Sie in jedem Fall auf der Kurve? Erklären Sie Ihre Beobachtungen.
10. Halten Sie die **Frequenz** auf 20 und die **Anzahl der Abtastwerte** auf 200 fixiert. Ändern Sie die **Abtastrate (Hz)** auf 500, 1,000 und 2,000. Achten Sie darauf, daß Sie die Ergebnisse verstehen.



Ende der Übung 12-1.

Schwingungs- und Muster-VIs

Sie werden bemerken, daß die Namen von den meisten Signalerzeugungs-VIs die Wörter *Schwingung* oder *Muster* enthalten. Es gibt einen wesentlichen Unterschied in der Funktionsweise der zwei verschiedenen Arten von VIs. Dieser betrifft die Tatsache, ob das VI die Phase des von ihm erzeugten Signals bei jedem Aufruf festhalten kann.

Phase-Bedienelement

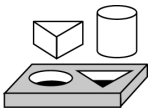
Die *Schwingungs-VI*s besitzen ein *Phaseneingang*-Bedienelement, wo Sie die Anfangsphase (in Grad) des ersten Abtastwerts von der erzeugten Kurvenform spezifizieren können. Sie verfügen ebenso über ein *Phasenausgang*-Anzeigeelement, das spezifiziert, was die Phase des nächsten Abtastwerts der erzeugten Kurvenform sein soll. Außerdem bestimmt ein *Phase zurücksetzen*-Bedienelement darüber, ob die Phase des ersten, beim Aufruf des *Schwingungs-VI*s erzeugten Abtastwerts diejenige sein soll, die am *Phaseneingang*-Bedienelement spezifiziert ist, oder diejenige, die bei der letzten Ausführung des VIs am *Phasenausgang*-

Bedienelement vorlag. Ein Wert von WAHR für *Phase zurücksetzen* setzt die Anfangsphase auf *Phaseneingang*, während ein Wert von FALSCH sie auf den Wert von *Phasenausgang* bei der letzten Ausführung des VIs setzt.

Die *Schwingungs-VI*s sind alle ablaufinvariant (können die Phase intern festhalten) und akzeptieren die Frequenz in normalisierte Einheiten (Zyklen/Abtastwert). Das einzige *Muster*-VI, das gegenwärtig normalisierte Einheiten gebraucht, ist das VI **Chirp-Muster**. Das Setzen des Booleschen Werts *Phase zurücksetzen* auf FALSCH erlaubt kontinuierliche Simulation von Abtastung.

☞ **Hinweis** ***Schwingungs-VIs sind ablaufinvariant und akzeptieren die Frequenzeingabe in normalisierte Einheiten.***

In der nächsten Übung werden Sie eine Sinus-Schwingung unter Verwendung sowohl von dem VI **Sinus-Schwingung** und dem VI **Sinus-Muster** generieren. Sie werden sehen, wie mit dem VI **Sinus-Schwingung** Sie die Anfangsphase besser als in dem VI **Sinus-Muster** steuern können.

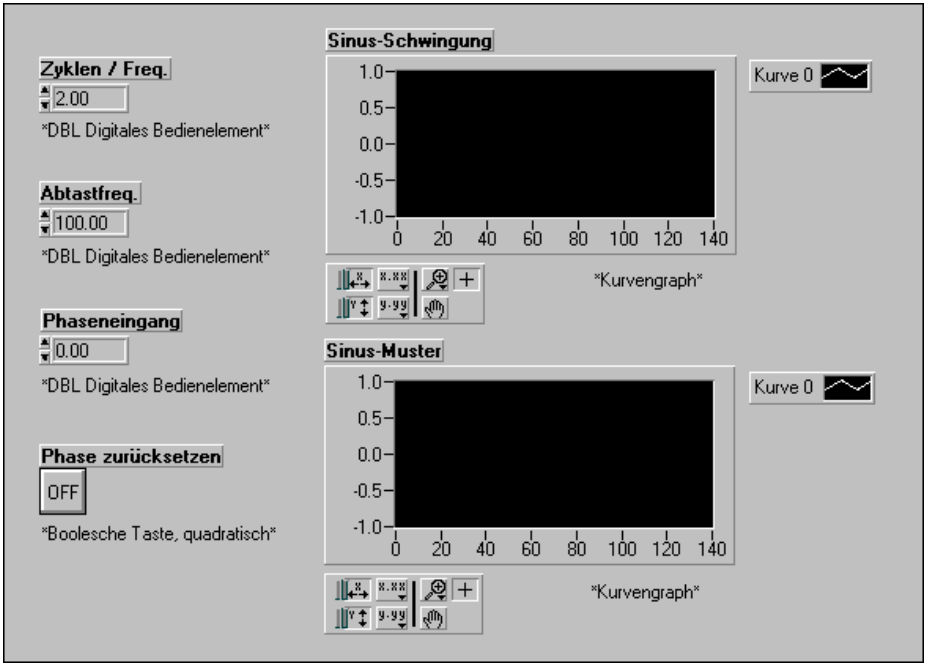


Übung 12-2. Mehr über die normalisierte Frequenz lernen

Ihr Übungsziel ist es, eine sinusoidale Kurvenform unter Verwendung sowohl vom VI Sinus Schwingung und vom VI Sinus-Muster zu erzeugen und die Unterscheide dazwischen zu verstehen.

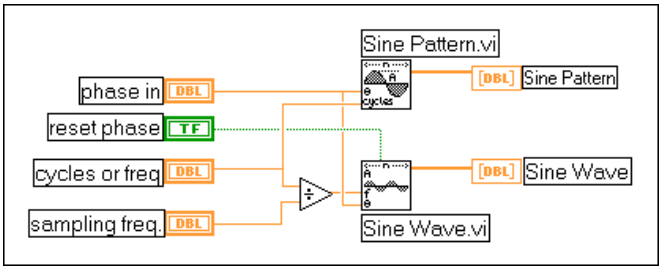
Frontpanel

1. Öffnen Sie ein neues Frontpanel, und erstellen Sie die in der folgenden Abbildung dargestellten Objekte.



Blockdiagramm

2. Erstellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm.



VI Sinus-Muster (Palette **Analyse»Signalerzeugung**).

VI Sinus-Schwingung (Palette **Analyse»Signalerzeugung**).

3. Speichern Sie das VI als `Wave and Pattern.vi` im Verzeichnis `LabVIEW\Activity`
4. Stellen Sie die Bedienelemente auf die folgenden Werte ein:

   Zyklen/Freq.: 2,00

   Abtastfreq.: 100

   Phaseneingang: 0,00

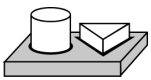
   Phase zurücksetzen: OFF

   Führen Sie das VI mehrmals aus.

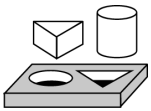
   Beobachten Sie, daß die **Kurve** der **Sinus-Schwingung** sich bei jeder Ausführung des VIs ändert. Da **Phase zurücksetzen** auf OFF eingestellt ist, ändert sich die Phase der Sinus-Schwingung mit jedem Aufruf des VIs, indem es jedesmal dem Wert von **Phasenausgang** beim letzten Aufruf gleicht. Die Kurve für das Sinus-Muster bleibt aber immer gleich, wobei sie 2 Zyklen der sinusoidalen Kurvenform zeigt. Die Anfangsphase der Kurve für das Sinus-Muster gleicht dem im Bedienelement **Phaseneingang** eingestellten Wert.

Hinweis ***"Phaseneingang" und "Phasenausgang" werden in Grad spezifiziert.***

5. Ändern Sie **Phaseneingang** auf 90, und führen Sie das VI mehrmals aus. Genau wie vorher ändert sich die Kurve für Sinus-Schwingung bei jeder Ausführung des VIs. Die Kurve für Sinus-Muster ändert sich nicht, aber die Anfangsphase des sinusoidalen Musters ist 90 Grad—genau wie im Bedienelement **Phaseneingang** spezifiziert.
6. Mit **Phaseneingang** noch immer auf 90 stellen Sie **Phase zurücksetzen** auf ON ein, und führen Sie das VI mehrmals aus. Die sinusoidalen Kurvenformen in den Kurven für Sinus-Schwingung und Sinus-Muster fangen beide bei 90 Grad an, ändern sich aber nicht bei nachfolgenden Aufrufen des VIs.
7. Indem Sie **Phase zurücksetzen** auf ON halten, führen Sie das VI mehrmals für jeden er folgenden Werte von **Phaseneingang** aus: 45, 180, 270 und 360. Beachten Sie die Anfangsphase der erzeugten Kurvenform bei jeder Ausführung des VIs.



Ende der Übung 12-2.



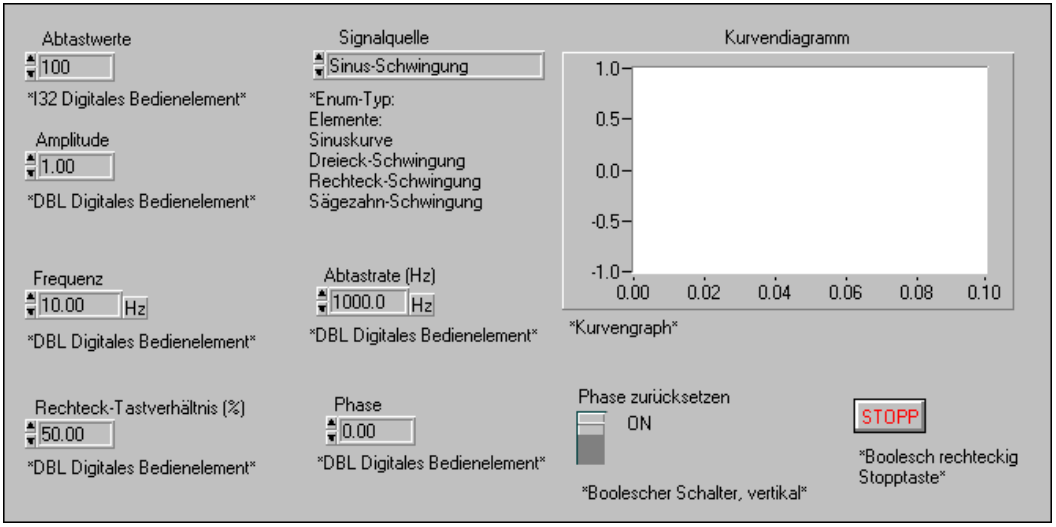
Übung 12-3. Einen Funktionsgenerator erstellen

Ihr Übungsziel ist es, einen einfachen Funktionsgenerator zu erstellen, der die folgenden Kurvenformen erzeugen kann.

- ***Sinus-Schwingung***
- ***Rechteck-Schwingung***
- ***Dreieck-Schwingung***
- ***Sägezahn-Schwingung***

Frontpanel

1. Öffnen Sie ein neues Frontpanel, und erstellen Sie die in der folgenden Abbildung dargestellten Objekte.



Das Bedienelement **Signalquelle** wählt die zu erzeugende Kurvenform aus.

Das Bedienelement **Rechteck-Tastverhältnis** wird nur zur Einstellung des Tastverhältnisses für die Rechteck-Schwingung verwendet.

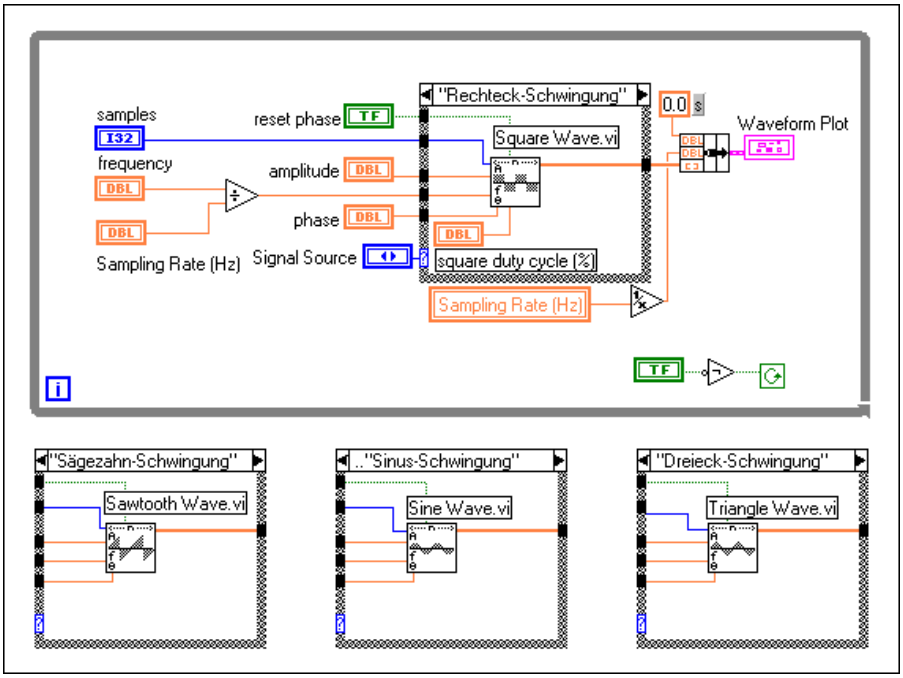
Das Bedienelement **Abtastwerte** bestimmt die Anzahl von Abtastwerten in der Kurve.

Beachten Sie, daß diese alle *Schwingungs-VI*s sind. Deshalb muß die Frequenzeingabe eine normalisierte Frequenz sein. Deshalb teilen Sie

Frequenz durch die **Rate** und das Ergebnis ist die normalisierte Frequenz, die mit dem *f*-Eingang der VIs verbunden ist.

Blockdiagramm

2. Erstellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm.



VI Sinus-Schwingung (Palette **Analyse»Signalerzeugung)** erzeugt eine Sinus-Schwingung der normalisierten Frequenz *f*.

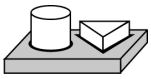
VI Dreieck-Schwingung (Palette **Analyse»Signalerzeugung)** erzeugt eine Dreieck-Schwingung der normalisierten Frequenz *f*.

VI Rechteck-Schwingung (Palette **Analyse»Signalerzeugung)** erzeugt eine Rechteck-Schwingung der normalisierten Frequenz *f* mit dem spezifizierten Abtastverhältnis.

VI Sägezahn-Schwingung (Palette **Analyse»Signalerzeugung)** erzeugt eine Sägezahn-Schwingung der normalisierten Frequenz *f*.

3. Speichern Sie das VI als `Function Generator.vi` im Verzeichnis `LabVIEW\Activity`.

4. Wählen Sie eine **Abtastrate** von 1000 Hz, **Amplitude** = 1, **Abtastwert** = 100, **Frequenz** = 10, **Phase zurücksetzen** = ON und **Signalquelle** = Sinus-Schwingung. Da **Abtastrate** = 1000 und **Frequenz** = 10 Hz, entsprechen 100 Abtastwerte einem Zyklus.
5. Führen Sie das VI aus, und beobachten Sie die sich ergebende Kurve.
6. Ändern Sie **Abtastwerte** auf 200, 300 und 400. Wieviele Zyklen der Kurvenform sehen Sie? Erklären Sie warum.
7. Mit **Abtastwerte** auf 100 eingestellt, ändern Sie **Phase zurücksetzen** auf OFF. Bemerken Sie einen Unterschied in der Kurve?
8. Ändern Sie **Frequenz** auf 10,01 Hz. Was geschieht? Warum?
9. Ändern Sie **Phase zurücksetzen** auf ON. Was geschieht diesmal? Erklären Sie warum.
10. Wiederholen Sie Schritte 4 bis 9 für verschiedene Kurvenformen, die im Bedienelement **Signalquelle** ausgewählt werden.



Ende der Übung 12-3.

13

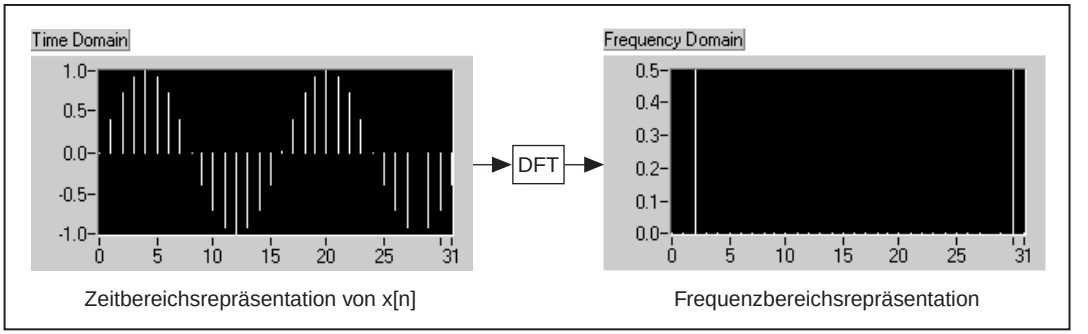
Digitale Signalverarbeitung

Dieses Kapitel beschreibt die Grundlagen der Schnellen Fourier-Transformation (FFT) und der Diskreten Fourier-Transformation (DFT) sowie ihre Verwendung in der Spektralanalyse. Für Beispiele, wie die digitale Signalverarbeitung-VI eingesetzt werden, sehen Sie sich bitte die Beispiele im Verzeichnis `examples\analysis\dspxmpl.llb` an.

Die Schnelle Fourier-Transformation (FFT)

Die Abtastwerte eines von einer DAQ-Karte erhaltenen Signals stellen die Repräsentation des Signals im *Zeitbereich* dar. Diese Repräsentation gibt die Amplituden des Signals zu den verschiedenen Z*eit*-Momenten, in der es abgetastet wurde, wieder. In vielen Fällen möchte man aber lieber den Frequenzinhalt eines Signals als die Amplituden der einzelnen Abtastwerte wissen. Die Repräsentation eines Signals durch seine einzelne Frequenzkomponente wird die Repräsentation des Signals im *Frequenzbereich* genannt. Die Repräsentation im Frequenzbereich könnte mehr Einsicht bieten über das Signal und das System, aus dem es erzeugt wurde.

Der Algorithmus, der zur Umwandlung von Abtastwerten der Daten aus dem Zeitbereich in den Frequenzbereich dient, ist als die *diskrete Fourier-Transformation* oder DFT bekannt. Die DFT stellt die Beziehung zwischen den Abtastwerten eines Signals im Zeitbereich und ihrer Repräsentation im Frequenzbereich dar. Die DFT findet vielfache Anwendung in Gebieten wie der Spektralanalyse, der angewandten Mechanik, der Akustik, der medizinischen Bildaufbereitung, der numerischen Analyse, der Instrumentation und der Nachrichtentechnik.



Nehmen wir an, Sie haben *N* Abtastwerte eines Signals von einer DAQ-Karte erhalten. Wenn Sie die DFT auf *N* Abtastwerte von dieser Repräsentation des Signals im Zeitbereich anwenden, hat das Resultat ebenfalls die Länge *N* Abtastwerte, aber die darin enthaltene Information gehört der Repräsentation im Frequenzbereich an. Die Beziehung zwischen den *N* Abtastwerten im Zeitbereich und den *N* Abtastwerten im Frequenzbereich wird unten erklärt.

Wird das Signal mit einer Abtastrate von f_s Hz abgetastet, so ist das Zeitintervall zwischen den Abtastwerten (das Abtastintervall also) Δt, wobei

$$\Delta t = \frac{1}{f_s}.$$

Die Abtastsignale werden durch x[i], $0 \le i \le N-1$ bezeichnet (d.h., insgesamt *N* Abtastwerte liegen vor). Wenn die diskrete Fourier-Transformation, durch

$$X_k = \sum_{i=0}^{N-1} x_i e^{-j2\pi ik/N} \qquad (13\text{-}1)$$

definiert, für

$$k = 0, 1, 2, \ldots, N-1$$

auf diese *N* Abtastwerte angewandt wird, so ist die Ausgabe (*X*[k], 0 ð k ð N–1) die Repräsentation von x[i] im Frequenzbereich. Bemerken Sie, daß der Zeitbereich *x* und der Frequenzbereich *X* beide insgesamt *N*

Abtastwerte aufweisen. Analog zum *Zeitabstand* von Δt zwischen den Abtastwerten von x im Zeitbereich, gibt es einen Frequenzabstand von

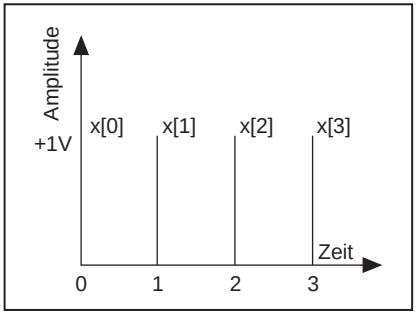
$$\Delta f = \frac{f_s}{N} = \frac{1}{N\Delta t}$$

zwischen den Komponenten von X im Frequenzbereich. Δf wird auch die *Frequenzauflösung* genannt. Zur Erhöhung der Frequenzauflösung (kleineres Δf) muß man entweder die Anzahl der Abtastwerte N (bei konstantem f_s) vergrößern oder die Abtastfrequenz f_s (bei konstantem N) verkleinern.

Im folgenden Beispiel werden Sie mittels Gleichung 13-1 die DFT für ein D.C-Signal berechnen.

DFT Kalkulationsbeispiel

Im nächsten Abschnitt werden Sie die genauen Frequenzen sehen, denen die N Abtastwerte der DFT entsprechen. Es sei für die vorliegende Diskussion angenommen, daß $X[0]$ DC oder dem Mittelwert des Signals entspricht. Um das Resultat der Kalkulation der DFT für eine Kurvenform unter Verwendung von der Gleichung 13-1 zu sehen, betrachten Sie ein DC-Signal mit einer konstanten Amplitude von +1 V. Vier Abtastwerte werden von diesem Signal genommen, wie in der untenstehenden Abbildung dargestellt.



Jeder Abtastwert hat den Wert +1, was die Zeitsequenz

$$x[0] = x[1] = x[3] = x[4] = 1$$

ergibt. Durch den Gebrauch von Gleichung 13-1 zur Berechnung der DFT von dieser Sequenz und unter Verwendung der Eulerschen Identität,

$$\exp(-i\theta) = \cos(\theta) - j\sin(\theta),$$

erhält man:

$$X[0] = \sum_{i=0}^{N-1} x_i e^{-j2\pi i 0/N} = x[0] + x[1] + x[2] + x[3] = 4$$

$$X[1] = x[0] + x[1]\left(\cos\left(\frac{\pi}{2}\right) - j\sin\left(\frac{\pi}{2}\right)\right) + x[2](\cos(\pi) - j\sin(\pi)) + x[3]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) = (1 - j - 1 + j) = 0$$

$$X[2] = x[0] + x[1](\cos(\pi) - j\sin(\pi)) + x[2](\cos(2\pi) - j\sin(2\pi)) + x[3](\cos(3\pi) - j\sin(3\pi)) = (1 - 1 + 1 - 1) = 0$$

$$X[3] = x[0] + x[1]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) + x[2](\cos(3\pi) - j\sin(3\pi)) + x[3]\left(\cos\left(\frac{9\pi}{2}\right) - j\sin\left(\frac{9\pi}{2}\right)\right) = (1 - j - 1 - j) = 0$$

Daher sind alle Werte außer der DC-Komponente, *X*[0], Null, was zu erwarten war. Der kalkulierte Wert von *X*[0] hängt jedoch von dem Wert von *N* (der Zahl der Abtastwerte) ab. Weil wir *N* = 4 hatten, so ist X[0] = 4. Wäre *N* = 10, dann hätten wir *X*[0] = 10 kalkuliert. Diese Abhängigkeit von *X*[ ] von *N* kommt auch für die anderen Frequenzkomponenten vor. Daher teilt man gewöhnlich die DFT-Ausgabe durch *N*, um so den richtigen Betrag der Frequenzkomponente zu erhalten.

Betrags- und Phaseninformation

Sie haben gesehen, daß die *N* Abtastwerte des Eingangsignals *N* Abtastwerte der DFT ergeben. Das heißt, die Zahl der Abtastwerte in den Zeit- und den Frequenzrepräsentationen ist gleich. Aus Gleichung 13-1 sieht man, daß *X*[k] immer komplex ist, ganz gleich ob das Eingangssignal *x*[i] real oder komplex ist (obwohl der Imaginärteil Null sein kann). Weil die DFT also komplex ist, enthält sie zwei Informationsteile—die Amplitude und die Phase. Es stellt sich heraus, daß für reale Signale (*x*[i]

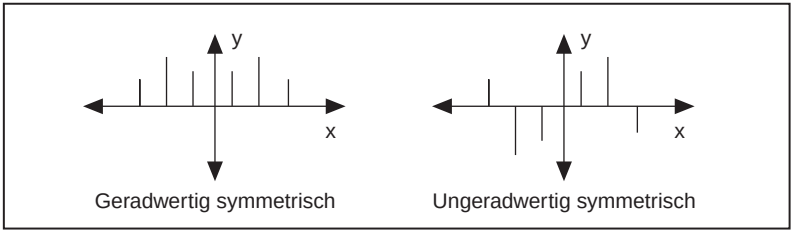
real), wie sie vom Ausgang von einem Kanal einer DAQ-Karte erhalten werden, die DFT symmetrisch ist, und zwar mit den folgenden Eigenschaften:

$$| X[k] | = | X[N-k] |$$

und

$$\text{Phase} (X[k]) = - \text{Phase}(X[N-k])$$

Die Begriffe zur Beschreibung von dieser Symmetrie sind, daß der Betrag von X[k] *geradwertig symmetrisch* ist und die Phase(X[k]) *ungeradwertig symmetrisch* ist. Ein geradwertig symmetrisches Signal ist eines, das um die y-Achse symmetrisch ist, während ein ungeradwertig symmetrisches Signal um den Ursprung symmetrisch ist. Dies ist in der folgenden Abbildung veranschaulicht.



Im Endeffekt bedeutet diese Symmetrie, daß es eine Wiederholung der in den *N* Abtastwerten der DFT enthaltenen Information gibt. Aufgrund dieser Informationswiederholung müssen nur die Hälfte der Abtastwerte von der DFT tatsächlich kalkuliert oder angezeigt werden, da die andere Hälfte durch diese Wiederholung zu erhalten ist.

Hinweis ***Wenn das Eingangssignal komplex ist, wird die DFT nicht symmetrisch sein, und man kann diesen Trick nicht benutzen.***

Frequenzabstand zwischen DFT/FFT Abstastwerten

Ist das Abtastintervall Δt Sekunden, und wird der erste Datenabtastwert ($k = 0$) um 0 Sekunden genommen, dann findet die k^{te} ($k > 0$, k Integer) Datenabtastwert um $k\Delta t$ Sekunden statt. In ähnlicher Weise findet, wenn die Frequenzauflösung Δf *Hz*

($\Delta f = \frac{f_s}{N}$) ist, der k^{te} Abtastwert der DFT zu einer Frequenz von $k\Delta f$ Hz statt. (Eigentlich gilt dies, wie Sie bald sehen werden, nur für bis zu der ersten Hälfte der Frequenzkomponenten. Die andere Hälfte stellen negative Frequenzkomponenten dar.) Je nachdem, ob die Anzahl der Abtastwerte, N, gerade oder ungerade ist, hat man eine andere Interpretation der Frequenz, die dem k^{ten} Abtastwert der DFT entspricht.

Nehmen wir z.B. an, daß N gerade ist und es sei $p = \frac{N}{2}$. Die folgende Tabelle zeigt die Frequenz, der jedes Formatelement der komplexen Ausgabesequenz X entspricht.

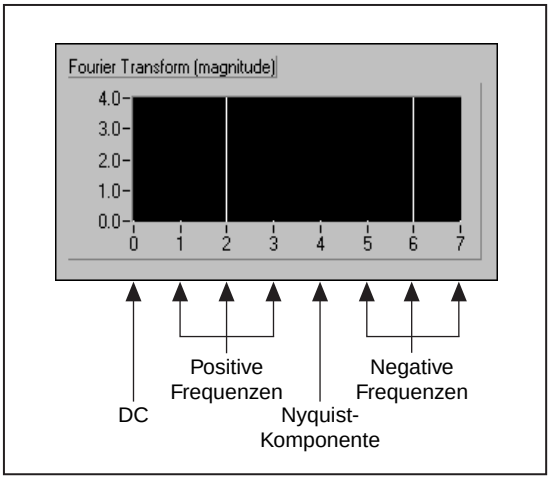
Bemerken Sie, daß das p^{te} Element, X[p], der Nyquist-Frequenz entspricht. Die negativen Eintragungen in der zweiten Spalte nach der Nyquist-Frequenz stellen *negative* Frequenzen dar.

Wenn z.B. $N = 8$, $p = N/2 = 4$, dann

X[0]	DC
X[1]	Δf
X[2]	$2\Delta f$
X[3]	$3\Delta f$
X[4]	$4\Delta f$ (Nyquist-Frequenz)
X[5]	$-3\Delta f$
X[6]	$-2\Delta f$
X[7]	$-\Delta f$

Hier haben X[1] und X[7] den gleichen Betrag, X[2] und X[6} haben den gleichen Betrag und X[3] und X[5] haben den gleichen Betrag. Der Unterschied ist, daß X[1], X[2] und X[3] positiven Frequenzkomponenten entsprechen, während X[5], X[6] und X[7] negativen Frequenzkomponenten entsprechen. Bemerken Sie, daß X[4] auf der Nyquist-Frequenz liegt.

Die folgende Abbildung stellt diese komplexe Sequenz für *N* = 8 dar.



Eine derartige Repräsentation, in der man sowohl die positiven als auch die negativen Frequenzen sieht, heißt eine *zweiseitige* Transformation.

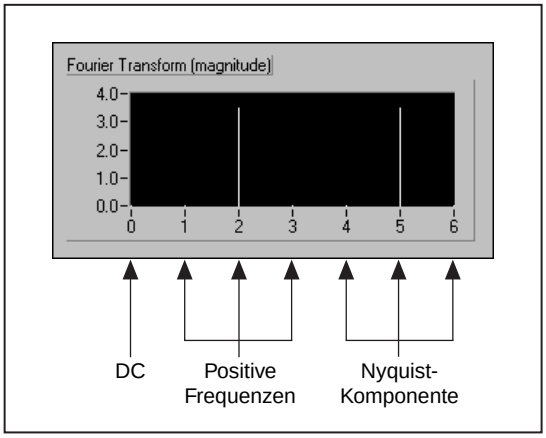
Bemerken Sie, daß es keine Komponente auf der Nyquist-Frequenz gibt, wenn *N* ungerade ist.

Wenn z.B. N = 7, p = (N–1)/2 = (7–1)/2 = 3 und es gilt

X[0]	DC
X[1]	Δf
X[2]	$2\Delta f$
X[3]	$3\Delta f$
X[4]	$-3\Delta f$
X[5]	$-2\Delta f$
X[6]	$-\Delta f$

Nun haben X[1] und X[6] den gleichen Betrag, X[2] und X[5] haben den gleichen Betrag und X[3] und X[4] haben den gleichen Betrag. X[1], X[2] und X[3] entsprechen jedoch positiven Frequenzen, während X[4], X[5] und X[6] negativen Frequenzen entsprechen. Weil *N* ungerade ist, gibt es keine Komponente auf der Nyquist-Frequenz.

Die folgende Abbildung stellt die vorstehende Tabelle für $N = 7$ dar.



Auch diese ist eine zweiseitige Transformation, weil sowohl positive als auch negative Frequenzen vorliegen.

Schnelle Fourier-Transformationen

Die Direktimplementierung von einer DFT auf N Datenabtastwerten erfordert ungefähr N^2 Komplex-Operationen und ist daher ein zeitintensives Verfahren. Wenn jedoch die Sequenzlänge eine Potenz von 2 ist,

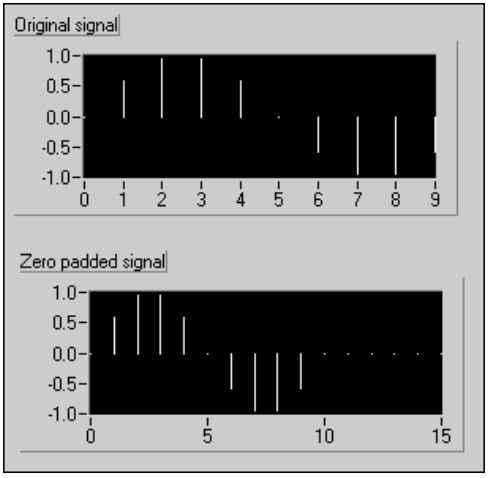
$$N = 2^m \quad \text{für } m = 1, 2, 3,\ldots,$$

kann man die Berechnung der DFT mit ungefähr $N \log_2(N)$ Operationen verwirklichen. Das macht die Berechnung der DFT wesentlich schneller und die Literatur der Digitalen Signalverarbeitung (DSP) bezeichnet diese Algorithmen als schnelle Fourier-Transformationen (FFT). Die FFT ist nichts anderes als ein schneller Algorithmus zur Berechnung der DFT, wenn die Anzahl der Abtastwerte (N) eine Potenz von 2 ist.

Zu den Vorteilen der FFT gehören Schnelligkeit und Speichereffizienz, da das VI die FFT “vor Ort” berechnen kann; d.h. keine zusätzlichen Speicherpuffer sind zur Berechnung der Ausgabewerte nötig. Die Länge der Eingabesequenz muß jedoch eine Potenz von 2 sein. Die DFT kann zwar eine Sequenz von beliebiger Länge verarbeiten, aber die DFT ist langsamer als die FFT und erfordert mehr Speicherplatz, weil sie zusätzliche Puffer zur Speicherung der Zwischenergebnisse während der Verarbeitung zuweisen muß.

Nullpolsterung

Eine Technik, die dazu verwendet wird, um die Eingabesequenz einer Potenz von 2 gleichzusetzen, ist das Hinzufügen von Nullen am Ende der Sequenz, damit die Gesamtzahl der nächsthöheren Potenz von 2 gleicht. Wenn man z.B. 10 Abtastwerte von einem Signal hat, kann man sechs Nullen hinzufügen, damit die Gesamtzahl von Abtastwerten 16 (= 2^4—eine Potenz von 2) gleicht. Dies ist unten dargestellt:



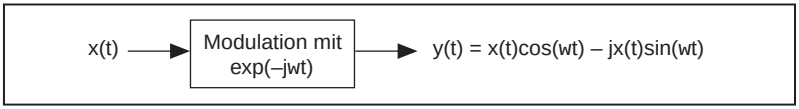
Die Hinzufügung von Nullen am Ende der Kurvenform im Zeitbereich berührt das Signalspektrum nicht. Neben der Gleichsetzung der Gesamtzahl von Abtastwerten an eine Potenz von zwei, damit durch Verwendung der FFT eine schnellere Berechnung ermöglicht wird, hilft diese Nullpolsterung auch dadurch, daß sie die Frequenzauflösung erhöht (man vergegenwärtige sich, daß Δf = fs/N), indem die Zahl der Abtastwerte *N* erhöht wird.

FFT-VIs in der Analysebibliothek

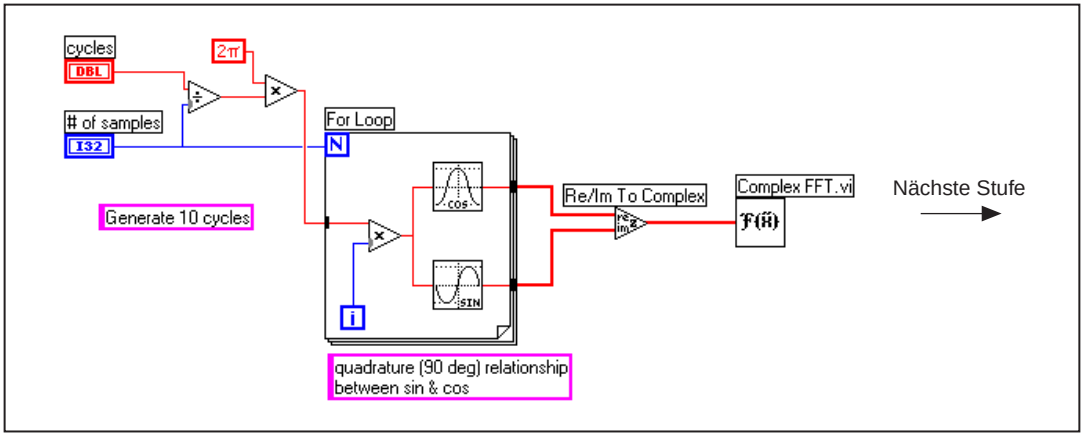
Die Analysebibliothek enthält zwei VIs, die die FFT von einem Signal berechnen. Das sind die VIs **Reale FFT** und **Komplexe FFT**.

Der Unterschied zwischen den zwei VIs ist, daß das VI **Reale FFT** die FFT von einem realwertigen Signal berechnet, während das VI **Komplexe FFT** die FFT von einem komplexwertigen Signal berechnet. Man sollte aber darauf achten, daß die Ausgaben von beiden VIs komplex sind.

Die meisten Signale in der realen Welt sind realwertig, und Sie können daher das VI **Reale FFT** für die meisten Anwendungen einsetzen. Sie könnten natürlich auch das VI **Komplexe FFT** benutzen, indem Sie den Imaginärteil des Signals Null gleichsetzen. Ein Beispiel von einer Anwendung, in der man das VI **Komplexe FFT** verwenden könnte, ist ein Signal, das sowohl aus einem Real- als auch aus einem Imaginärteil besteht. Ein derartiges Signal kommt häufig im Gebiet der Nachrichtentechnik vor, in der man die Kurvenform durch eine komplexe Potenz moduliert. Der Vorgang der Modulation durch eine komplexe Potenz ergibt ein komplexes Signal, wie unten dargestellt:



Das folgende Blockdiagramm zeigt in vereinfachter Weise wie Sie 10 Zyklen von einem komplexen Signal erzeugen können:

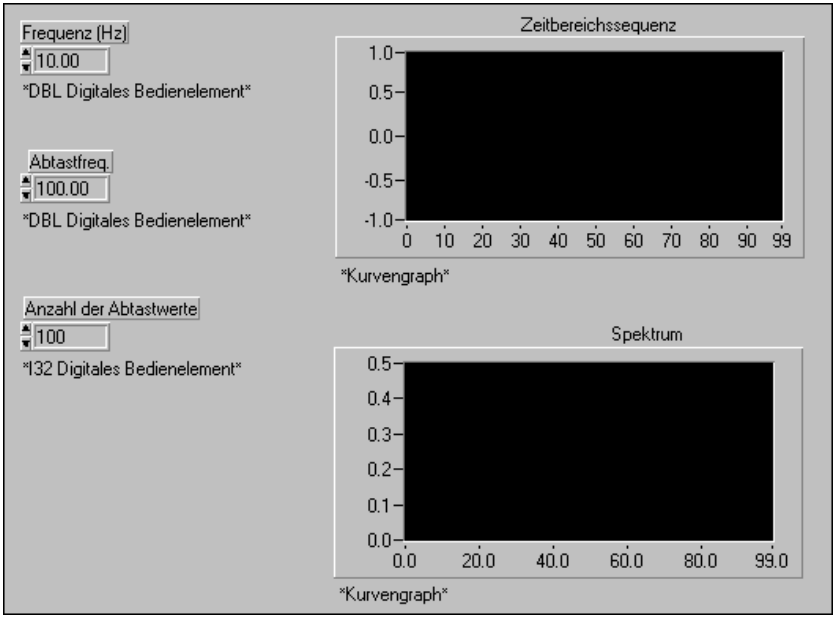


Übung 13-1. Das VI Reale FFT verwenden

In dieser Übung ist es Ihr Ziel, die zweiseitige und die einseitige Fourier-Transformation von einem Signal unter Verwendung des VIs Reale FFT anzuzeigen und die Wirkung von Aliasing im Frequenzspektrum zu beobachten.

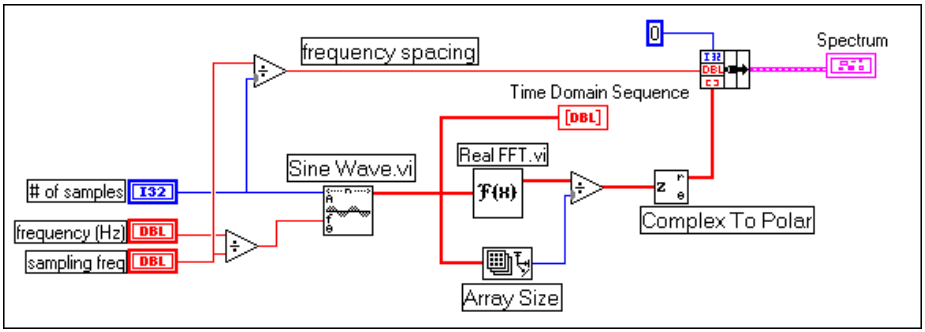
Frontpanel

1. Erstellen Sie das in der folgenden Abbildung dargestellte Frontpanel.



Blockdiagramm

2. Erstellen Sie das in der folgenden Abbildung dargestellte Blockdiagramm.



Die Funktion **Array-Größe** (Palette **Funktionen»Array**) skaliert die Ausgabe der FFT durch die **Anzahl der Abtastwerte,** um so die richtige Amplitude der Frequenzkomponenten zu erhalten.

Die Funktion **Sinus-Schwingung** (Palette **Funktionen»Analyse» Signalerzeugung**) erzeugt eine sinusoidale Kurvenform im Zeitbereich.

Die Funktion **Reale FFT** (Palette **Funktionen»Analyse»Digitale Signalverarbeitung**) berechnet die FFT der eingegebenen Datenabtastwerte.

Die Funktion **Komplex in Polar** (Palette **Funktionen»Numerisch» Komplex**) trennt die komplexe Ausgabe der FFT in ihre Real- und Imaginärteile (Betrag und Phase). Die Phaseninformation ist in Einheiten von Radianten. Hier zeigen Sie nur den Betrag der FFT an.

Der Frequenzabstand Δf ergibt sich durch Teilung der **Abtastfreq.** durch die **Anzahl der Abtastwerte**.

3. Speichern Sie dieses VI als `FFT_2sided.vi` im Verzeichnis `LabVIEW\Activity`.
4. Wählen Sie **Frequenz (Hz)** = 10, **Abtastfreq.** = 100 und **Anzahl der Abtastwerte** = 100 aus. Führen Sie das VI aus.

Beachten Sie die Kurven der Zeit-Kurvenform und das Frequenzspektrum. Da **Abtastfreq.** = **Anzahl der Abtastwerte** = 100 ist, tasten Sie im Endeffekt für 1 Sekunde. Die Zahl der Zyklen von der Sinus-Schwingung, die Sie in der Zeit-Kurvenform sehen, gleicht der **Frequenz (Hz),** die Sie auswählen. In diesem Fall werden Sie 10 Zyklen sehen. (Wenn Sie die **Frequenz (Hz)** auf 5 ändern, werden Sie 5 Zyklen sehen.)

Zweiseitige FFT

5. Untersuchen Sie das Frequenzspektrum (die Fourier-Transformation). Sie werden zwei Spitzen bemerken—eine auf 10 Hz und die andere auf 90 Hz. Die Spitze auf 90 Hz ist eigentlich die negative Frequenz von 10 Hz. Die Kurve, die Sie sehen, heißt die zweiseitige *FFT*, weil sie sowohl die positiven als auch die negativen Frequenzen anzeigt.
6. Führen Sie das VI mit **Frequenz (Hz)** = 10 und dann mit **Frequenz (Hz)** = 20 aus. Notieren Sie sich für jeden Fall die Verschiebung in beiden Spitzen des Spektrums.

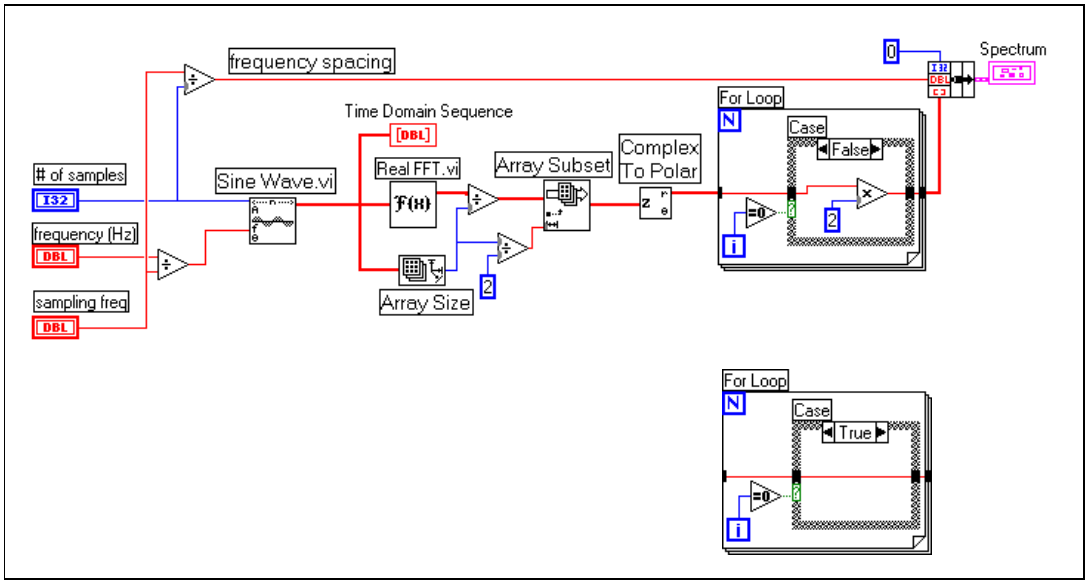
Hinweis ***Beobachten Sie auch die Kurven im Zeitbereich für Frequenz (Hz) = 10 und 20. Welche bietet eine bessere Repräsentation der Sinus-Schwingung? Warum?***

7. Da fs = 100 Hz ist, können Sie nur Signale mit einer Frequenz < 50 Hz (Nyquist-Frequenz = fs/2) genau abtasten. Ändern Sie die **Frequenz (Hz)** auf 48 Hz. Sie sollten die Spitzen auf ± 48 Hz auf der Spektrumkurve sehen.
8. Ändern Sie nun die **Frequenz (Hz)** auf 52 Hz. Gibt es einen Unterschied zwischen dem Ergebnis von Schritt 5 und dem, was Sie nun auf den Kurven sehen? Da 52 > Nyquist ist, ist die Frequenz 52 durch Aliasing nunmehr |100 – 52| = 48 Hz.
9. Ändern Sie die **Frequenz (Hz)** auf 30 Hz und 70 Hz, und führen Sie das VI aus. Gibt es einen Unterschied zwischen den zwei Fällen? Erklären Sie warum.

Einseitige FFT

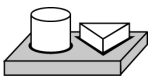
10. Modifizieren Sie das Blockdiagramm des VIs wie unten dargestellt. Sie haben gesehen, daß die FFT Informationswiederholung aufwies, weil sie Informationen sowohl über die positiven als auch über die negativen Frequenzen enthielt. Diese Modifizierung zeigt nun nur die Hälfte der FFT-Punkte (nur die positiven Frequenzkomponenten). Diese Repräsentation heißt die *einseitige FFT.* Die einseitige FFT zeigt nur die positiven Frequenzkomponenten. Bemerken Sie, daß Sie die

positiven Frequenzkomponenten mit zwei multiplizieren müssen, um die richtige Amplitude zu erhalten. Die DC-Komponente bleibt aber unberührt.



Die Funktion **Gleich 0?** (Palette **Funktionen»Vergleich**) überprüft, ob der Arrayindex gleich Null ist. Wenn ja, entspricht er der D.C.- Komponente und sollte nicht mit zwei multipliziert werden.

11. Führen Sie das VI mit den folgenden Werten aus: **Frequenz (Hz)** = 30, **Abtastfreq.** = 100, **Anzahl der Abtastwerte** = 100.
12. Speichern Sie das VI als `FFT_1sided.vi` im Verzeichnis `LabVIEW\Activity`.
13. Ändern Sie den Wert von **Frequenz (Hz)** auf 70, und führen Sie das VI aus. Bemerken Sie einen Unterschied gegenüber dem Ergebnis von Schritt 9?



Ende der Übung 13-1.

Das Leistungsspektrum

Sie haben gesehen, daß die DFT (oder die FFT) von einem realen Signal eine komplexe Zahl ist, die also einen Real- und einen Imaginärteil aufweist. Die *Leistung* in jeder durch die DFT/FFT repräsentierte Frequenzkomponente läßt sich durch Quadrieren des Betrags von dieser Frequenzkomponente errechnen. Die Leistung in der k^{ten} Frequenzkomponente (das k^{te} Element der DFT/FFT) ist daher durch $|X[k]|^2$ gegeben. Die Kurve, die die Leistung in jeder der Frequenzkomponenten zeigt, heißt das *Leistungsspektrum*. Da die DFT/FFT von einem realen Signal symmetrisch ist, gleicht die Leistung auf einer positiven Frequenz von $k\Delta f$ der Leistung auf der entsprechenden negativen Frequenz von $-k\Delta f$ (DC- und Nyquist-Komponenten nicht eingeschlossen). Die Gesamtleistung in den DC- und Nyquist-Komponenten ist $|X[0]|^2$ bzw. $\left|X\left[\frac{N}{2}\right]\right|^2$.

Verlust von Phaseninformation

Da die Leistung durch Quadrieren von dem Betrag der DFT/FFT erhalten wird, ist das Leistungsspektrum immer realwertig. Der Nachteil davon ist, daß die Phaseninformation verloren geht. Wenn Sie das Leistungsspektrum erfassen wollen, müssen Sie die DFT/FFT verwenden, wodurch Sie einen komplexen Ausgabewert erhalten.

Das Leistungsspektrum können Sie in Anwendungen verwenden, in denen die Phaseninformation nicht nötig ist (um beispielsweise die harmonische Leistung in einem Signal zu berechnen). Sie können eine sinusförmige Eingabe an ein nichtlineares System anlegen und die Leistung in den Harmonischen am Systemsausgang beobachten.

Frequenzabstand zwischen Abtastwerten

Sie können das VI **Leistungsspektrum** in der Subpalette **Analyse» Digitale Signalverarbeitung** zur Berechnung des Leistungsspektrums der Datenabtastwerte im Zeitbereich einsetzen. Genau wie bei der DFT/FFT, ist die Zahl der Abtastwerte vom Ausgang des VIs **Leistungsspektrum** mit der Zahl der an dem Dateneingang angelegten Abtastwerte identisch. Der Frequenzabstand zwischen den Datenabtastwerten am Ausgang ist außerdem Δf = fs/N.

Zusammenfassung

Die Repräsentation von einem Signal im Zeitbereich (Abtastwerte) kann in die Repräsentation im Frequenzbereich mittels eines als diskrete Fourier-Transformation (DFT) bekannten Algorithmusses umgewandelt werden. Für eine schnelle Berechnung der DFT wird ein als schnelle Fourier-Transformation (FFT) bekannter Algorithmus verwendet. Diesen Algorithmus können Sie benutzen, wenn die Anzahl der Signalabtastwerte eine Potenz von zwei ist.

Die Ausgabe der konventionellen DFT/FFT ist zweiseitig, weil sie Information über die positiven und die negativen Frequenzen enthält. Diese Ausgabe kann in eine einseitige DFT/FFT durch den Gebrauch von nur die Hälfte der DFT/FFT-Ausgabepunkte umgewandelt werden. Der Frequenzabstand zwischen den Abtastwerten der DFT/FFT ist $\Delta f = fs/N$.

Das Leistungsspektrum läßt sich von der DFT/FFT durch Quadrieren des Betrags von den einzelnen Frequenzkomponenten berechnen. Das VI **Leistungsspektrum** in der fortgeschrittenen Analysebibliothek besorgt dies automatisch für Sie. Die Einheiten der Ausgabe vom VI Leistungsspektrum sind in $Vrms^2$. Das Leistungsspektrum liefert aber keine Phaseninformation.

Die DFT, die FFT und das Leistungsspektrum sind zur Messung des Frequenzinhalts von stationären oder transienten Signalen nützlich. Die FFT liefert den durchschnittlichen Frequenzinhalt des Signals über die gesamte Dauer der Signalerfassung. Deshalbe benutzen Sie die FFT hauptsächlich zur Analyse von stationären Signalen (wenn sich das Signal über die Zeit der Signalerfassung nicht signifikant im Frequenzinhalt ändert) oder wenn Sie nur die gemittelte Energie auf jeder Frequenzlinie wissen wollen. Eine große Klasse von Meßproblemen fällt in diese Kategorie. Zur Messung von Frequenzinformationen, die sich während der Erfassung ändern, sollten Sie das Zeit-Frequenzanalyse-Toolkit (JTFA) oder das Kleinwellen- und Filterbankdesigner-Toolkit (WFBD) zu Hilfe ziehen.

14

Fensterglättung

Dieses Kapitel erklärt wie der Gebrauch von Fenstern die Spektralleckage vermeidet und die Analyse der erfaßten Signale verbessert. Für Beispiele der Verwendung von den Analysefenster-VIs, sehen Sie sich bitte die Beispiele an, die sich in `examples\analysis\windxmpl.llb` befinden.

Einführung in die Glättungsfenster

In praktischen Anwendungen der Signalabtastung kann man nur eine endliche Aufzeichnung von dem Signal bekommen, selbst wenn man den Abtastungssatz und die Abtastungsbedingungen sorgfältig beachtet. Für das diskrete Zeitsystem ist es leider der Fall, daß die endliche Abtastungsaufzeichnung zu einer abgestrichenen Kurvenform führt, die eine andere Spektraleigenschaft gegenüber dem ursprünglichen kontinuierlichen Zeitsignal hat. Diese Diskontinuitäten erzeugen Spektralleckage, die ein Spektrum in diskreter Zeit ergeben, das eine verschmierte Version des ursprünglichen Spektrums in kontinuierlicher Zeit darstellt.

Ein einfaches Mittel zur Verbesserung der Spektraleigenschaften von einem abgetasteten Signal ist die Anwendung von Glättungsfenstern. Bei der Durchführung von Fourier- oder Spektralanalyse auf Daten von endlicher Länge kann man Fenster zur Minimierung der Übergangsrändern an den abgestrichenen Kurvenformen einsetzen und dabei die Spektralleckage reduzieren. So eingesetzt, gleichen Glättungsfenster vordefinierten, schmalbändigen Tiefpaßfiltern.

Über die Spektralleckage und Glättungsfenster

Wenn Sie die DFT-FFT zur Auffindung des Frequenzinhalts von einem Signal verwenden, wird implizit angenommen, daß die vorliegenden Daten eine einzige Periode von einer periodisch wiederkehrenden Kurvenform sind. Dies ist in der Abbildung 14-1 dargestellt. Die erste dargestellte Periode ist die abgetastete Kurve. Die dieser Periode entsprechende Kurvenform wird dann über die Zeit wiederholt, um die periodische Kurvenform zu ergeben.

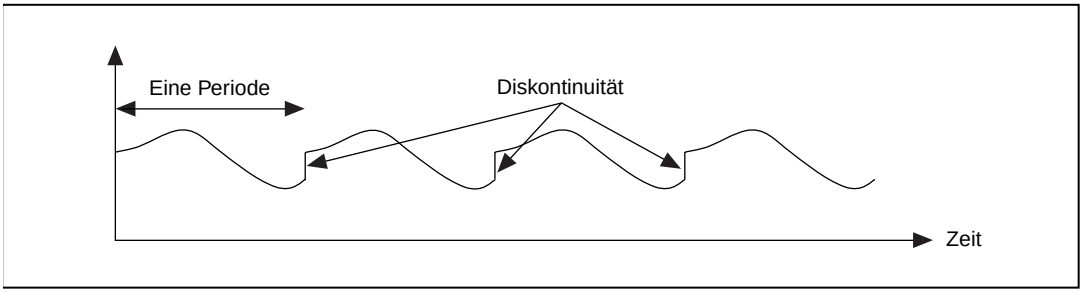


Abbildung 14-1. Von der abgetasteten Periode kreierte Kurvenform

Wie in der vorigen Abbildung ersichtlich, werden aufgrund der angenommenen Periodizität Diskontinuitäten zwischen sukzessiven Perioden vorkommen. Dies geschieht, wenn man eine nicht-integrale Anzahl von Zyklen abtastet. Diese *künstlichen* Diskontinuitäten kommen als sehr hohe Frequenzen im Signalspektrum vor, d.h., Frequenzen, die im ursprünglichen Signal nicht vorhanden waren. Diese Frequenzen können viel höher als die Nyquist-Frequenz sein, und wie Sie bereits gesehen haben, werden sie dann durch Aliasing irgendwo zwischen 0 und fs/2 erscheinen. Das durch die Verwendung von DFT/FFT erhaltene Spektrum wird daher nie das eigentliche Spektrum des ursprünglichen Signals sein, sondern es wird immer eine verschmierte Version sein. Es sieht so aus, als ob die Energie von einer Frequenz auf alle anderen Frequenzen *ausgeleckt wäre*. Dieses Phänomen wird deshalb als *Spektralleckage* bezeichnet.

Die Abbildung 14-2 zeigt eine Sinus-Schwingung und die entsprechende Fourier-Transformation. Die im Zeitbereich abgetastete Kurvenform ist im Graphen 1 zu sehen. Da die Fourier-Transformation eine Periodizität voraussetzt, wird diese Kurvenform in der Zeit wiederholt, und die periodische Zeitkurvenform der Sinus-Schwingung im Graphen 1 ist im Graphen 2 dargestellt. Die entsprechende Spektraldarstellung ist im Graphen 3 gezeigt. Da die zeitliche Aufzeichnung im Graphen 2 periodisch und ohne Diskontinuitäten ist, besteht sein Spektrum aus einer einzigen Linie, die die Frequenz der Sinus-Schwingung aufweist. Die Kurvenform im Graphen 2 hat deshalb keine Diskontinuitäten, weil Sie eine integraleZahl (im vorliegenden Fall 1) von Zyklen der zeitlichen Kurvenform abgetastet haben.

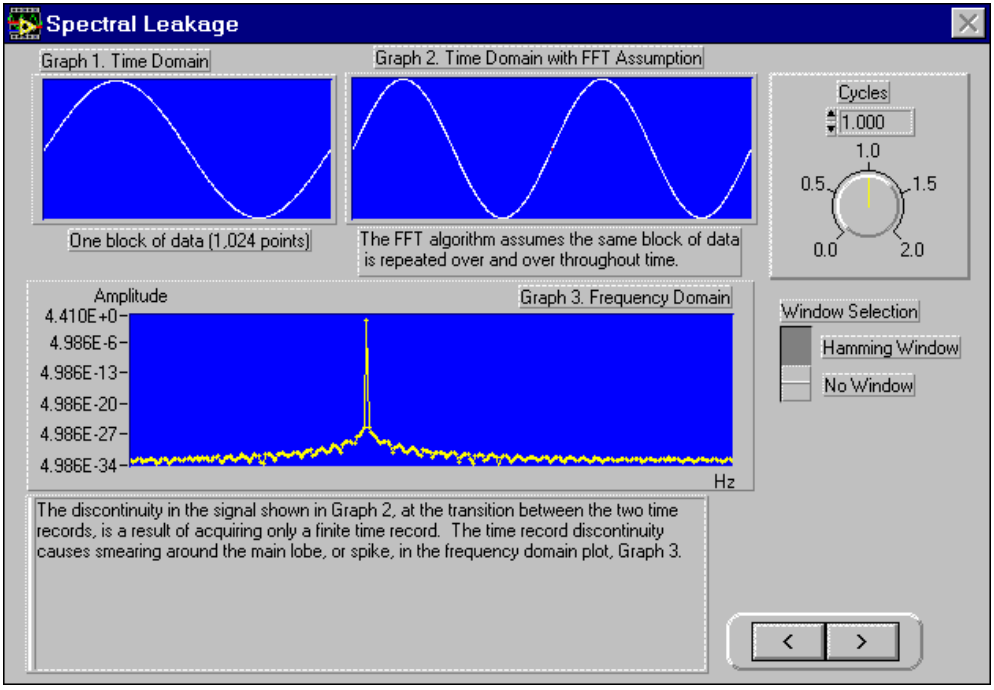


Abbildung 14-2. Sinus-Schwingung und die entsprechende Fourier-Transformation

In der Abbildung 14-3 sehen Sie die spektrale Darstellung, wenn man eine nicht-integrale Zahl von Perioden der zeitlichen Kurvenform (nämlich 1,25) abtastet. Der Graph 1 besteht nunmehr aus 1,25 Zyklen der Sinus-Schwingung. Wenn dies periodisch wiederholt wird, besteht die resultierende Kurvenform, wie im Graphen 2 dargestellt, aus Diskontinuitäten. Das entsprechende Spektrum ist im Graphen 3 dargestellt. Bemerken Sie, wie die Energie jetzt über einen weiten Frequenzbereich ausgebreitet ist. Diese Schmierung der Energie ist die*Spektralleckage*. Die Energie ist aus einer der FFT-Linien herausgeleckt und hat sich auf alle anderen Linien verschmiert.

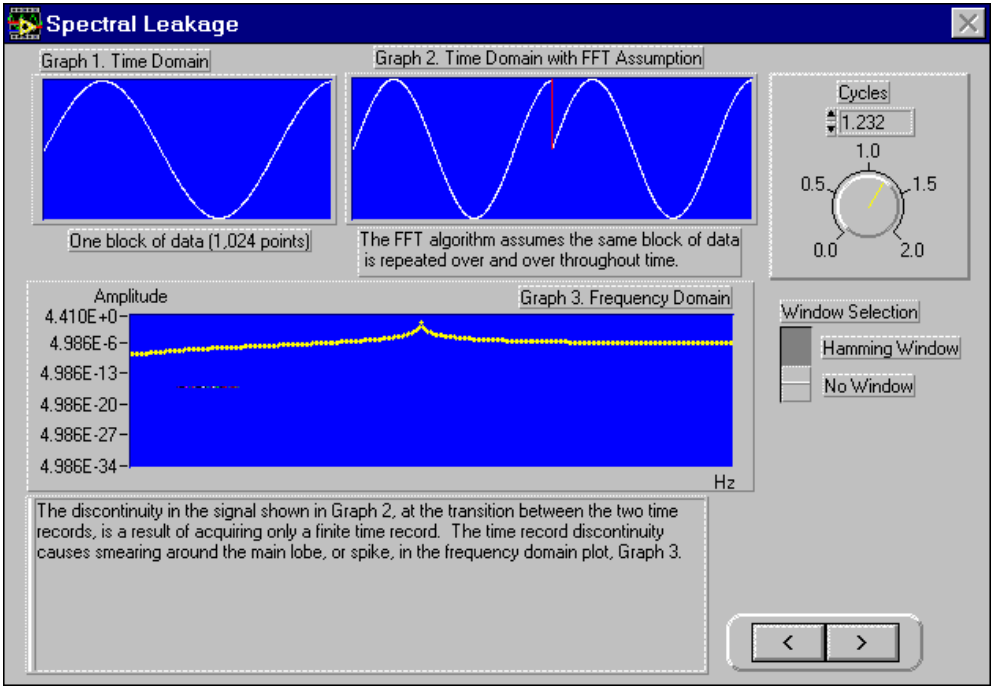


Abbildung 14-3. Spektrale Darstellung bei Abtastung über eine nicht-integrale Zahl von Abtastwerten

Leckage existiert aufgrund der endlichen zeitlichen Aufzeichnung des Eingangssignals. Eine Lösung zur Überwindung der Leckage ist es, eine unendliche Zeitaufzeichnung vorzunehmen, von –Unendlichkeit bis +Unendlichkeit. Dann würde die FFT eine einzige Linie auf der richtigen Frequenz errechnen. Eine unendliche Zeit zu warten ist aber in der Praxis unmöglich. Weil man daher auf eine zeitlich endliche Aufzeichnung begrenzt ist, wird eine andere Technik, als Fensterung bekannt, zur Reduzierung der Spektralleckage eingesetzt.

Die Größe der Spektralleckage hängt von der Amplitude der Diskontinuität ab. Je größer die Diskontinuität, desto mehr Leckage und umgekehrt. Sie können Fensterung benutzen, um die Amplitude der Diskontinuitäten an den Rändern von jeder Periode zu verringern. Dies besteht darin, daß die Zeitaufzeichnung mit einem Fenster endlicher Länge multipliziert wird, wobei die Amplitude des Fensters glatt und allmählich gegen Null an den Rändern variiert. Dies zeigt sich in der Abbildung 14-4, wo das ursprüngliche Zeitsignal unter Verwendung von einem *Hanning*-Fenster geglättet wird. Bemerken Sie, daß die zeitliche Kurvenform des geglätteten Signals allmählich auf Null an den Rändern abschrägt. Bei der Durchführung von Fourier- oder Spektralanalyse an Daten endlicher Länge, kann man also Fenster zur Minimierung der Übergangsränder der abgetasteten Kurvenform einsetzen. Eine Glättungsfenster-Funktion, die auf die Daten vor der Umwandlung in den Frequenzbereich angewandt wird, minimiert die Spektralleckage.

Beachten Sie, daß die Annahme der Periodizität zu keinen Diskontinuitäten führt, wenn die zeitliche Aufzeichnung eine integrale Anzahl von Zyklen enthält, wie in der Abbildung 14-2 dargestellt, und es kommt daher keine Spektralleckage vor. Das Problem stellt sich nur, wenn man eine nicht-integrale Anzahl von Zyklen hat.

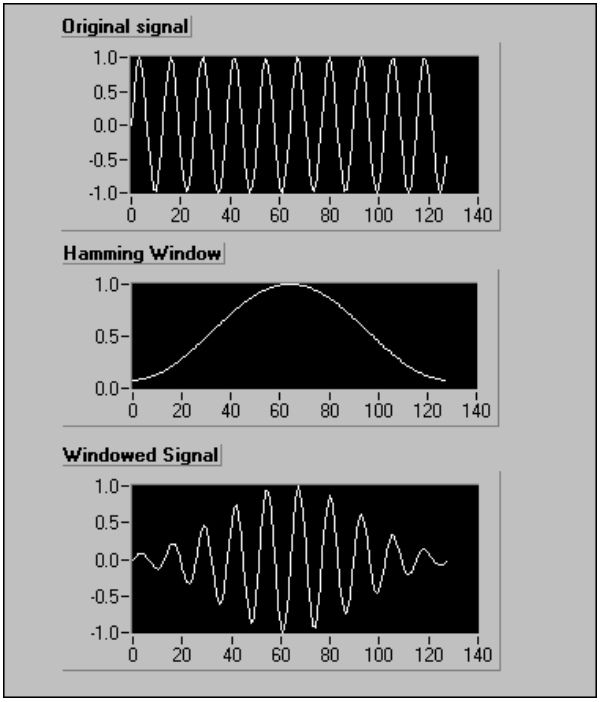


Abbildung 14-4. Zeitsignal, geglättet unter Verwendung eines Hamming-Fensters

Fensterung-Anwendungen

Es gibt mehrere Gründe, warum man Fensterung gebraucht. Einige davon sind:

- Um die Dauer einer Beobachtung zu definieren.
- Verminderung der Spektralleckage.
- Trennung eines Signals von kleiner Amplitude von einem Signal größerer Amplitude, wobei die beiden sehr ähnliche Frequenzen haben.

Charakteristiken der verschiedenen Arten von Fensterung-Funktionen

Ein Fenster einem Signal im Zeitbereich aufzulegen (Fensterung) ist das Äquivalent zu einer Multiplikation des Signals mit der Fensterung-Funktion. Da die Multiplikation im Zeitbereich mit einer Konvolution im Frequenzbereich äquivalent ist, so ist das Spektrum des Signals mit Windowing eine Konvolution von dem Spektrum des ursprünglichen Signals mit dem Spektrum vom Fenster. Neben der Veränderung der Form des Signals im Zeitbereich, beeinflußt die Fensterung ebenfalls das Spektrum, das man sieht.

Viele verschiedene Sorten von Fenstern stehen in der Analysebibliothek in LabVIEW zur Verfügung. Je nach Ihrer Anwendung sind einige nützlicher als andere. Einige von diesen Fenstern sind:

Rechteckig (kein Fenster)

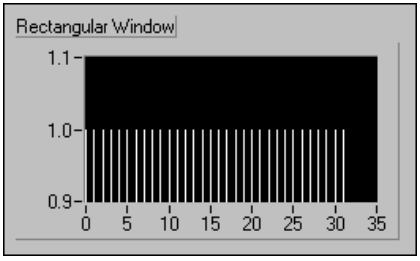
Das rechteckige Fenster hat einen Wert von eins über sein ganzes Zeitintervall. Mathematisch läßt es sich so schreiben:

$$w[n] = 1,0$$

für

$$n = 0, 1, 2.....N-1,$$

wobei *N* die Länge des Fensters ist. Das Auflegen von einem rechteckigen Fenster ist das Äquivalent von dem Gebrauch von keinem Fenster. Das kommt daher, daß die rechteckige Funktion das Signal einfach innerhalb von einem endlichen Zeitintervall abschneidet. Das rechteckige Fenster hat die höchste Spektralleckage. Das rechteckige Fenster für $N = 32$ ist in der folgenden Abbildung dargestellt:



Das rechteckige Fenster ist nützlich zur Analyse von Transienten von einer Dauer, die kleiner als die Dauer des Fensters ist. Es wird auch in der *Ordnungs-Abtastung* verwendet, in der die effektive Abtastrate mit der Geschwindigkeit der Welle in rotierenden Maschinen proportional ist. In dieser Anwendung erfaßt sie den Hauptmodus der Schwingung in der Maschine und ihre Harmonischen.

Hanning

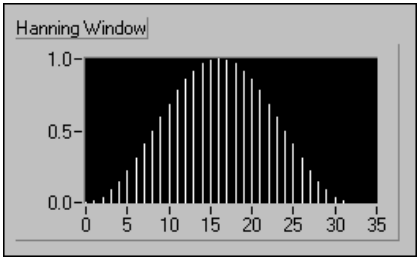
Das Fenster hat eine Form, die einem Halbzyklus von einer Kosinuskurve ähnlich ist. Seine Bestimmungsgleichung lautet:

$$w(n) = 0{,}5 - 0{,}5\cos(2\pi n/N)$$

für

$$n = 0, 1, 2, ...N-1$$

Ein Hanning-Fenster mit N = 32 ist unten dargestellt:



Das Hanning-Fenster ist nützlich für die Analyse von Transienten, die länger als die zeitliche Dauer des Fensters sind, und auch für allgemeine Anwendungen.

Hamming

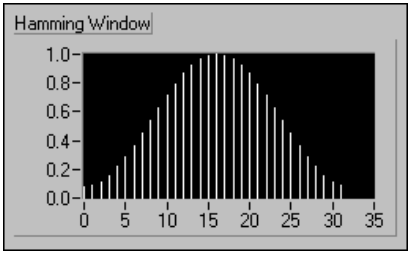
Dieses Fenster ist eine modifizierte Version des Hanning-Fensters. Seine Form ist ebenfalls der einer Kosinuskurve ähnlich. Es läßt sich als

$$w(n) = 0{,}54 - 0{,}46\cos(2\pi n/N)$$

definieren für

$$n = 0, 1, 2, ...N-1$$

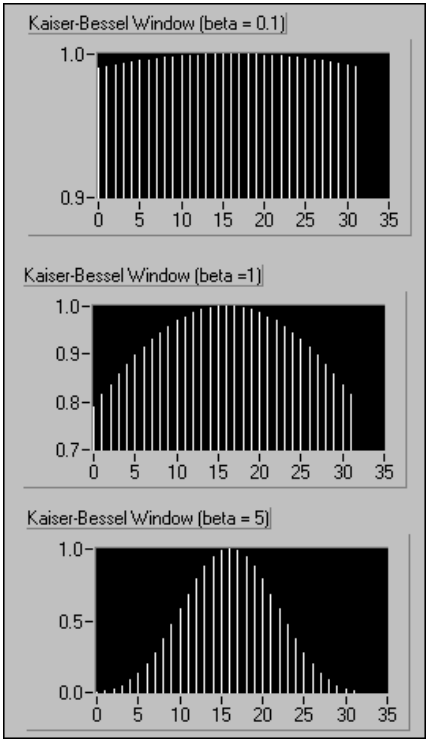
Ein Hamming-Fenster mit N = 32 ist unten dargestellt:



Sie sehen, daß das Hanning- und das Hamming-Fenster eine gewisse Ähnlichkeit aufweisen. Bemerken Sie aber, daß im Zeitbereich das Hamming-Fenster an den Rändern Null nicht so nahekommt, wie dies beim Hanning-Fenster der Fall ist.

Kaiser-Bessel

Dieses Fenster ist ein “flexibles” Fenster, dessen Form der Benutzer durch Einstellen des Parameters *beta* modifizieren kann. Sie können also, je nach Ihrer Anwendung, die Form des Fensters verändern, um den Grad an Spektralleckage zu kontrollieren. Kaiser-Bessel Fenster für verschiedene Werte von *beta* sind unten dargestellt:



Bemerken Sie, daß die Form für kleine beta-Werte beinahe die eines rechteckigen Fensters ist. In der Tat, man bekommt für beta = 0,0 ein rechteckiges Fenster. Mit zunehmender beta schrägt sich das Fenster zunehmend seitlich ab.

Das Fenster eignet sich gut zur Erfassung von zwei Signalen mit fast gleicher Frequenz, aber ganz unterschiedlichen Amplituden.

Dreieck

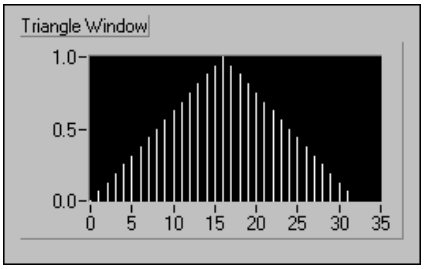
Die Form dieses Fensters ist die eines Dreiecks. Sie ist durch

$$w[n] = 1 - | (2n-N) / N |$$

gegeben für

$$n = 0, 1, 2, ..., n-1$$

Ein Dreieck-Fenster für N = 32 ist unten dargestellt:



Flattop

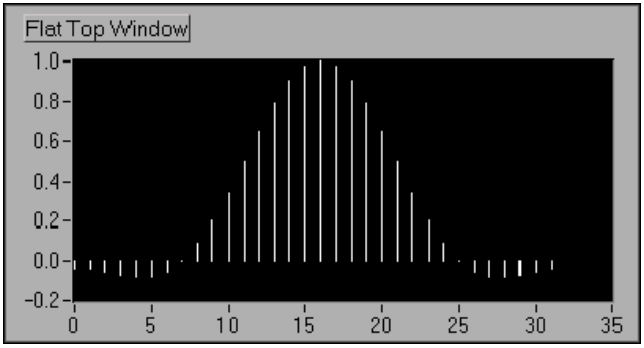
Dieses Fenster hat die beste Amplitudengenauigkeit von allen Fensterung-Funktionen. Die erhöhte Amplitudengenauigkeit (± 0,02 dB für Signale zwischen exakt integralen Zyklen) geht zu Lasten der Frequenzselektivität. Das Flattop-Fenster ist am nützlichsten für die genaue Messung der Amplitude von einzelnen Frequenzkomponenten, wo wenig spektrale Energie im benachbarten Signal vorhanden ist. Das Flattop-Fenster läßt sich als

$$w(n) = a0 - a1*\cos(2\pi n/N) + a2*\cos(4\pi n/N)]$$

definieren mit

$$a0 = 0{,}2810638602$$
$$a1 = 0{,}5208971735$$
$$a2 = 0{,}1980389663$$

Ein Flattop-Fenster ist unten dargestellt:



Exponential

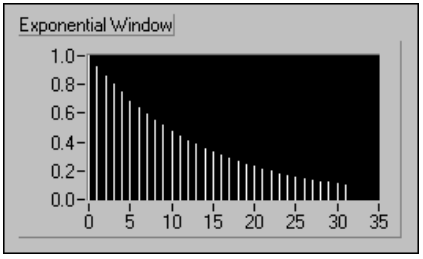
Die Form dieses Fensters entspricht der einer abklingenden Exponentialfunktion. Mathematisch läßt sie sich als

$$w[n] = e^{\left(\frac{n\ln(f)}{N-1}\right)} = f^{\left(\frac{n}{N-1}\right)}$$

schreiben für

$$n = 0, 1, 2.....N - 1,$$

wobei *f* der Endwert ist. Der Anfangswert des Fensters ist eins, und er klingt allmählich gegen Null ab. Der Endwert der Exponentialfunktion läßt sich zwischen 0 und 1 einstellen. Das Exponential-Fenster für N = 32 mit dem Endwert als 0,1 angegeben ist unten dargestellt:



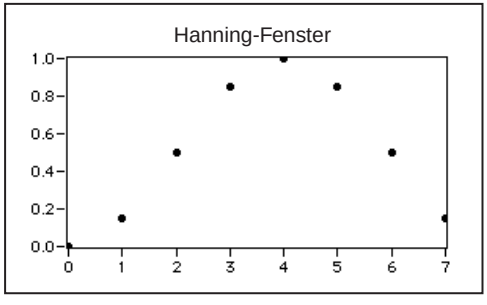
Dieses Fenster ist nützlich zur Analyse von Transienten (Signale, die nur während einer kurzen Zeitdauer existieren), deren Dauer länger als die Fensterlänge ist. Dieses Fenster kann Signalen aufgelegt werden, die exponential abklingen, wie z.B. das Ansprechen von Strukturen mit geringer Dämpfung, die durch einen Schlag (z.B. von einem Hammer) erregt werden.

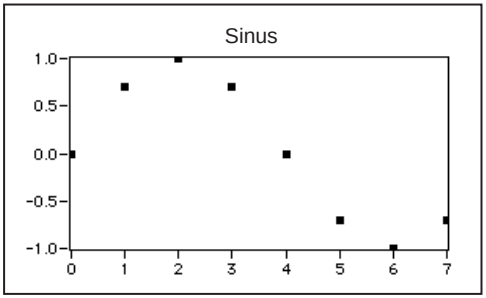
Fenster für die Spektralanalyse gegenüber Fenstern zur Koeffizientenauslegung

Die Fenster-VIs, die in der Analysebibliothek von LabVIEW implementiert werden, wurden für Anwendungen in der Spektralanalyse ausgelegt. In diesen Anwendungen wird dem Eingangssignal ein Fenster aufgelegt, indem es durch eines der Fenster-VIs geleitet wird. Das geglättete Signal wird dann an ein VI auf der Grundlage von DFT zur Anzeige und Analyse im Frequenzbereich weitergeleitet.

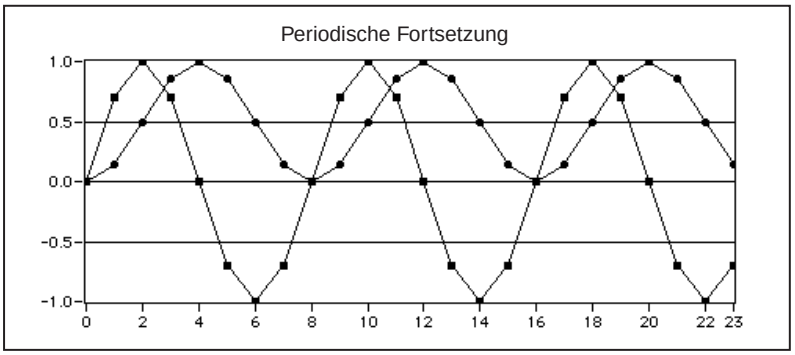
Die für die Spektralanalyse ausgelegten Fensterung-Funktionen müssen *DFT-even* sein, nach dem Begriff, der von Fredric J. Harris in seinem Artikel definiert wurde: *On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform* (*Proceedings of the IEEE*, Volume 66, No.1, Januar 1978). Eine Fensterung-Funktion ist DFT-even, falls sein inneres Produkt mit integralen Zyklen von Sinus-Sequenzen identisch Null ist. Eine andere Art, eine DFT-even-Sequenz aufzufassen, ist, daß seine DFT keine Imaginärkomponente hat.

Die folgenden Abbildungen illustrieren das Hanning-Fenster und einen Zyklus von einem Sinus-Muster für eine Probengröße von 8. Die Abbildungen zeigen, daß das DFT-even Hanning-Fenster um seinen Mittelpunkt nicht symmetrisch und sein letzter Punkt nicht gleich seinem ersten Punkt ist, und entspricht somit einem kompletten Zyklus von einem Sinus-Muster.





Die DFT hält letztlich Eingabesequenzen für periodisch—das zu analysierende Signal ist also eine Verkettung von dem Eingabesignal. Die folgende Abbildung zeigt drei solcher Zyklen der vorigen Sequenz, um die glatte periodische Fortsetzung von dem DFT-even Fenster und dem einzykligen Sinus-Muster zu demonstrieren.



Eine andere Art von Fensteranwendung ist die der Auslegung von FIR-Filtern, vgl. den Abschnitt *Gefensterte FIR-Filter* im Kapitel 16, *Filterung*. Diese Anwendung erfordert Fenster, die um ihren Mittelpunkt symmetrisch sind.

Die folgenden Gleichungen der Hanning Fensterung-Funktion illustrieren den Unterschied zwischen der DFT-even Fensterung-Funktion (Spektralanalyse) und die symmetrische Fensterung-Funktion (Koeffizientenauslegung).

Hanning Fensterung-Funktion für Spektralanalyse:

$$w[i] = 0,5\left(1 - \cos\left(\frac{2\pi i}{N}\right)\right)$$

für

$$i=0,1, 2, ..., N-1$$

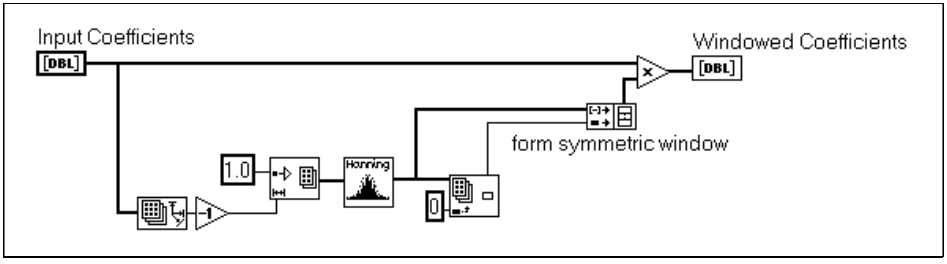
Hanning Fensterung-Funktion für symmetrische Koeffizientenauslegung:

$$w[i] = 0,5\left(1 - \cos\left(\frac{2\pi i}{N-1}\right)\right)$$

für

$$i=0, 1, 2, ..., N-1$$

Die obigen zwei Gleichungen zeigen, daß man die symmetrischen Fensterung-Funktionen durch leichte Modifizierung des Einsatzes von den DFT-even Fensterung-Funktionen erhalten kann. Die folgende Abbildung zeigt ein Blockdiagramm, das das VI Hanning-Fenster zur Verwirklichung von symmetrischer Fensterung von Filterkoeffizienten verwendet.



Vergleiche Anhang A, *Analyse-Referenzen*, für weitere Auskunft über Glättungsfenster.

Welche Art Fenster soll ich benutzen?

Da Sie nun einige der vielen verschiedenen Fenstertypen, die Ihnen zur Verfügung stehen, gesehen haben, fragen Sie sich vielleicht: ‘Welche Art Fenster soll ich benutzen?’ Die Antwort hängt davon ab, welche Art von Signal Sie haben und was Sie suchen. Das richtige Fenster zu wählen erfordert einige Vorkenntnis des zu analysierenden Signals. Die folgende Tabelle zeigt summarisch die verschiedenen Signalarten und die geeigneten Fenster, die man mit ihnen verwenden kann.

Signaltyp	Fenster
Transienten, deren Dauer kürzer als die Fensterdauer ist	Rechteckig
Transienten, deren Dauer länger als die Fensterdauer ist	Exponential, Hanning
Allgemeine Anwendungen	Hanning
Ordnungsabtastung	Rechteckig
Systemanalyse (Frequenzgangmessungen)	Hanning (bei zufälliger Erregung), Rechteckig (bei pseudo-zufälliger Erregung)
Trennung von zwei Tönen mit sehr nahen Frequenzen, aber sehr verschiedenen Amplituden	Kaiser-Bessel
Trennung von zwei Tönen mit sehr nahen Frequenzen, aber mit fast gleichen Amplituden	Rechteckig
Genaue Amplitudenmessungen auf einzelnen Tönen	Flat Top

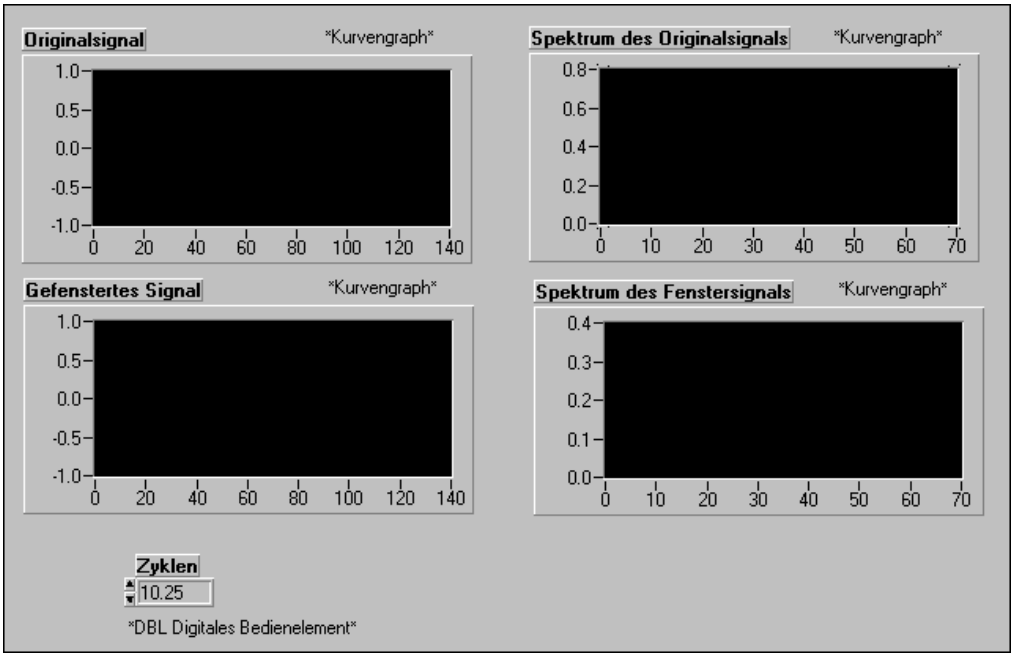
In vielen Fällen haben Sie nicht genug Vorkenntnis über das Signal und müssen daher mit verschiedenen Fenstern experimentieren, um das beste zu finden.

Übung 14-1. Ein Signal mit und ohne Fensterung vergleichen

Ihr Übungsziel ist es, den Unterschied (sowohl im Zeit- als auch im Frequenzbereich) zwischen einem Signal mit und einem ohne Fensterung zu beobachten.

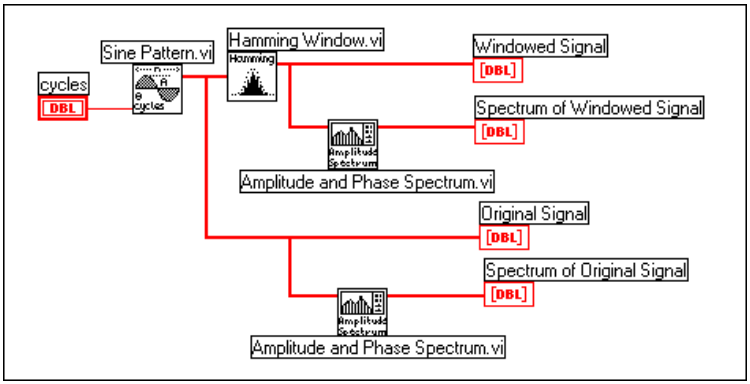
Frontpanel

1. Öffnen Sie ein neues Frontpanel, und erstellen Sie die Objekte, wie in der folgenden Abbildung gezeigt.



Blockdiagramm

2. Erstellen Sie das in der folgenden Abbildung gezeigte Blockdiagramm.



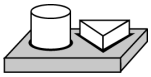
Das VI **Sinus-Muster** (**Palette Funktionen»Analyse»Signalerzeugung**) erzeugt eine Sinus-Schwingung mit der im Bedienelement **Zyklen** spezifizierten Anzahl von Zyklen.

Der zeitlichen Kurvenform der Sinus-Schwingung wird ein Fenster unter Verwendung des VIs **Hamming-Fenster** (Palette **Funktionen»Analyse»Fenste**r) aufgelegt, und zeitliche Kurvenformen mit und ohne Fensterung werden an den zwei linken Kurven am Frontpanel angezeigt.

Das VI **Amplituden- und Phasenspektrum** (Palette **Funktionen»Analyse»Messung**) erhält das Amplitudenspektrum der zeitlichen Kurvenformen mit und ohne Fensterung. Diese Kurvenformen werden an den zwei Kurven an der rechten Seite des Frontpanels angezeigt.

3. Speichern Sie das VI als `Windowed & Unwindowed Signal.vi` im Verzeichnis `LabVIEW\Activity`.
4. Stellen Sie **Zyklen** auf 10 (eine integrale Zahl) ein, und führen Sie das VI aus. Beachten Sie, daß das Spektrum des Signals mit Fensterung dicker (breiter) ist als das Spektrum des Signals ohne Fensterung. Aber beide Spektra sind in der Nähe von 10 an der x-Achse konzentriert.

5. Ändern Sie **Zyklen** auf 10,25 (eine nichtintegrale Zahl), und führen Sie das VI aus. Beachten Sie, daß das Spektrum des Signals ohne Fensterung jetzt weiter als vorher ausgespreizt ist. Dies kommt daher, daß Sie nun eine nichtintegrale Zahl von Zyklen haben, und wenn Sie die Kurvenform wiederholen, um sie periodisch zu machen, bekommen Sie Diskontinuitäten. Das Spektrum des Signals mit Fensterung ist noch konzentriert, aber das des Signals ohne Fensterung hat sich über den ganzen Frequenzbereich geschmiert. (Dies ist die Spektralleckage.)
6. Ändern Sie **Zyklen** auf 10,5, und beobachten Sie die Kurven im Frequenzbereich. Spektralleckage des ursprünglichen Signals ist leicht ersichtlich.



Ende der Übung 14-1.

15

Spektralanalyse und -messung

Dieses Kapitel zeigt, wie man das Amplituden- und das Phasenspektrum bestimmt, einen Spektrumanalysator entwickelt und den Klirrfaktor (THD) von Signalen bestimmt. Für Beispiele der Verwendung von den Messung-VIs, sehen Sie sich bitte die Beispiele an, die sich in `examples\analysis\measure\measxmpl.llb` befinden.

Einführung in die Messung-VIs

Einige Messung-VIs führen oft benutzte Umwandlungen vom Zeitbereich in den Frequenzbereich durch, so wie z.B. Bestimmungen von Amplituden- und Phasenspektrum, Signalleistungsspektrum, Netzwerktransferfunktion usw. Andere Messung-VIs wirken mit VIs zusammen, die Funktionen wie die skalierte Zeitbereichfensterung oder die Leistungs- und Frequenzabschätzung durchführen.

Sie können die Messung-VIs für die folgenden Anwendungen einsetzen:

- Anwendung in der Spektralanalyse:
  - Amplituden- und Phasenspektrum
  - Leistungsspektrum
  - Skaliertes Zeitbereichfenster
  - Leistungs- und Frequenzabschätzung
  - Spektrumanalyse und Messungen des Klirrfaktors (THD)
- Anwendungen in der Netzwerk- und Zweikanal-Analyse:
  - Impulsantwortfunktion
  - Netzwerkfunktionen (darunter die Kohärenz)
  - Kreuzleistungsspektrum

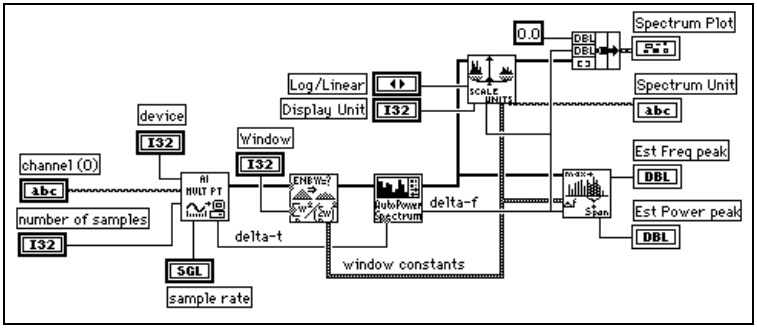
DFT, FFT und das Leistungsspektrum sind zur Messung des Frequenzinhalts von stationären oder transienten Signalen nützlich. Die FFT liefert den gemittelten Frequenzinhalt des Signals über die ganze Dauer der Signalerfassung. Man benutzt die FFT daher hauptsächlich für die Analyse von stationären Signalen (wobei sich das Signal im Frequenzinhalt nicht signifikant über die Zeitdauer der Erfassung ändert),

oder wenn man nur die Durchschnittsenergie in jeder Frequenzlinie wissen möchte. Eine große Klasse von Meßproblemen fällt in diese Kategorie hinein. Zur Messung von Frequenzinformation, die sich während der Erfassung ändert, sollten Sie Zeit-Frequenzanalyse-VIs, wie z.B. das Gabor Spektrogramm, verwenden.

Die Messung-VIs sind auf den Signalverarbeitung-VIs aufgebaut worden und haben die folgenden Eigenschaften, die das Verhalten von konventionellen Frequenzanalyse-Instrumenten auf dem Labortisch modellieren.

- Sie gehen von realen Signaleingaben aus dem Zeitbereich aus.
- Ausgaben sind in Betrag und Phase, skaliert, gegebenenfalls in geeigneten Einheiten und daher sofort in Graphen einsetzbar.
- Einseitige Spektra von DC bis zur $\frac{\text{Abtastfrequenz}}{2}$.
- Umwandlung von Abtastperiode in Frequenzintervall für graphische Darstellung mit geeigneten Einheiten auf der X-Achse (in Hz).
- Berichtigungen für die eingesetzten Fenster werden in geeigneten Fällen vorgenommen.
- Fenster sind derart skaliert, daß jedes Fenster das gleiche Spitzenergebnis für die Spektrumamplitude innerhalb seiner Genauigkeitsbeschränkungen ausgibt.
- Leistungs- oder Amplitudenspektra werden in verschiedenen Einheitsformaten gezeigt, darunter Dezibel und Einheiten der Spektraldichte, wie V^2/Hz, $V/\sqrt{Hz}$ usw.

Sie können die Messung-VIs im allgemeinen direkt mit dem Ausgang der Datenerfassungs-VIs und mit Graphen über den Achsen-Cluster verbinden, wie das folgende Diagramm eines Spektrumanalysators zeigt.



Die Messungsbeispiele umfassen die folgenden:

- Beispiel Amplitudenspektrum
- Beispiel Simulierte Dynamische Signalanalyse
- Beispiel Klirrfaktor (THD)

Mit Hardware von National Instruments können Sie die folgenden Beispiele benutzen.

- Einfacher Spektrumanalysator und Spektrumanalysator—Beide funktionieren mit beliebiger Analog-Eingabehardware. (Verwenden Sie dynamische Signalerfassungs-Hardware für Messungen guter Qualität.)
- Dynamischer Signalanalysator und Netzwerkanalysator—Beide funktionieren mit dynamischer Signalerfassungs-Hardware (DSA).

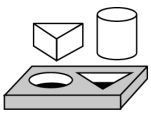
Sie werden lernen

- Informationen über die Messung-VIs und wie sie zur Durchführung von verschiedenen Signalverarbeitung-Operationen eingesetzt werden können.
- Wie man das Frequenzspektrum (Amplitude & Phase) von einem Signal im Zeitbereich mit den geeigneten Einheiten berechnet.
- Wie man den Frequenzgang von einem System mit den geeigneten Einheiten berechnen kann, indem die Erregungs- und Antwortsignale des Systems verarbeitet werden.
- Wie die Kohärenz-Funktion berechnet wird und wie sie zum Verständnis der Frequenzgang-Messungen benutzt wird.
- Wie der Klirrfaktor in einem Signal bestimmt wird.

Spektrumanalyse

Das Amplituden- und das Phasenspektrum eines Signals berechnen

In vielen Anwendungen bietet die Kenntnis des Frequenzinhalts von einem Signal Einsicht in das System, das dieses Signal erzeugte. Sie können die gewonnene Information verwenden zur Analyse des Frequenzinhalts von Tönen, zur Kalibrierung von Instrumenten, zur Abschätzung der Geräusche und Schwingungen, die durch Maschinenteilen erzeugt werden usw. Die folgende Übung demonstriert, wie das VI Amplituden- und Phasenspektrum zur Messung der Amplitude und Phase von einem Signal verwendet wird.

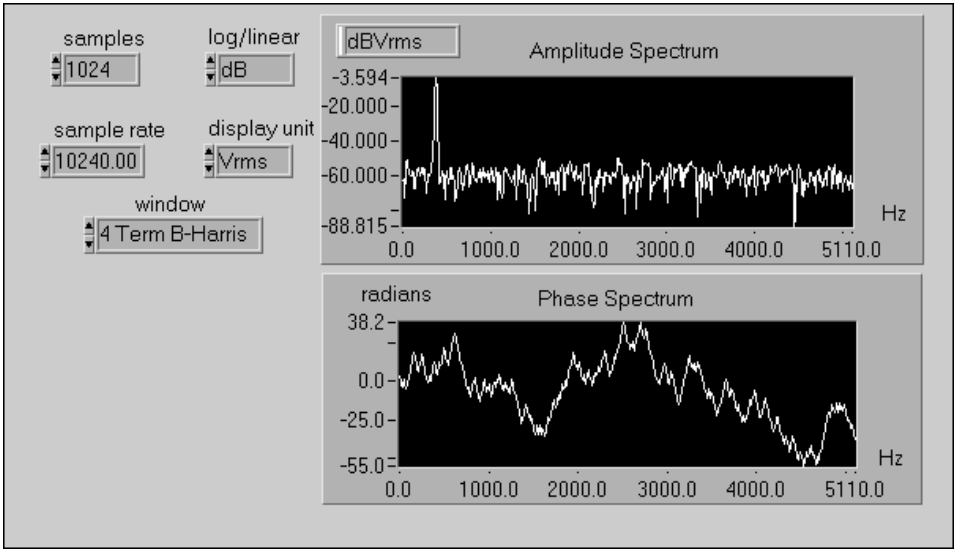


Übung 15-1. Das VI Amplituden und Phasenspektrum benutzen

In dieser Übung ist es Ihr Ziel, das Amplituden- und Phasenspektrum eines Signals zu berechnen.

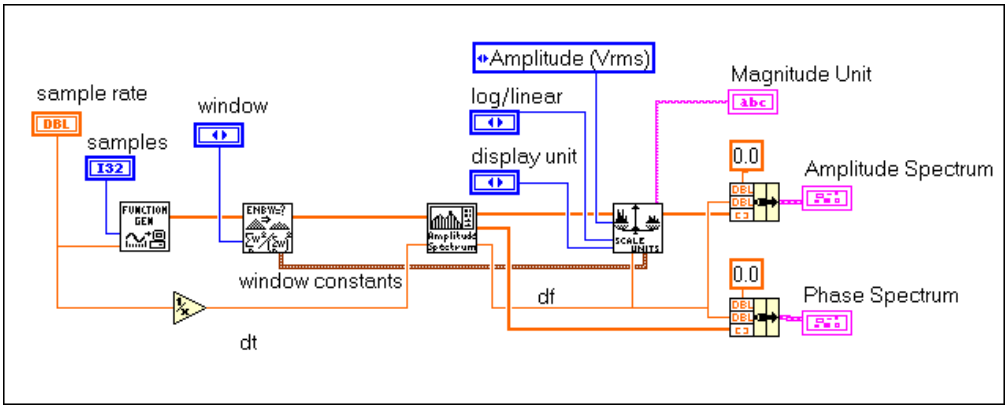
Frontpanel

1. Öffnen Sie das VI Amp Spectrum Example in der Bibliothek `examples\analysis\measure\measxmpl.llb`. Das Signal wird durch das VI Einfacher Funktionsgenerator erzeugt, das einen Multifunktionsgenerator mit additivem weißen Rauschen simuliert.



Blockdiagramm

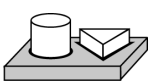
2. Öffnen Sie das Blockdiagramm, und untersuchen Sie es.



Das VI Amplituden- und Phasenspektrum berechnet das Amplitudenspektrum und das Phasenspektrum eines Signals im Zeitbereich. Die Verbindungen mit diesem VI werden unten diskutiert.

Das Eingangssignal im Zeitbereich wird an dem Bedienelement Signal (V) angelegt. Der Betrag und die Phase vom Spektrum des Eingangssignals sind an den Ausgängen Amp.-Spektrum Betrag (Vrms) bzw. Amp.-Spektrum Phase (Radianten) greifbar. Das VI Konvertieren der Spektraleinheit dient zur Umwandlung der Ausgabewerte, ursprünglich in Vrms, vom VI Amplituden- und Phasenspektrum in andere gängige Einheiten (Vrms, Vpk, Vrms^2, Vpk^2, $\text{Vrms}\sqrt{\text{Hz}}$, $\text{Vpk}\sqrt{\text{Hz}}$, Vrms^2/Hz und Vpk^2/Hz). Die vier letzten Einheiten sind die der Amplitude-Spektraldichte ($\text{Vrms}\sqrt{\text{Hz}}$, $\text{Vpk}\sqrt{\text{Hz}}$) und der Leistungs-Spektraldichte (Vrms^2/Hz, und Vpk^2/Hz). Der Ausgabe-Cluster **Fensterkonstanten** vom VI Skaliertes Zeitbereichfenster enthält Konstanten für das ausgewählte Fenster, die für Messungen der Spektraldichte nötig sind.

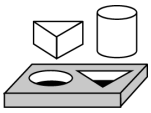
3. Führen Sie das VI aus.
4. Führen Sie das Beispiel Amp.-Spektrum kontinuierlich mit offenem Frontpanel vom einfachen Funktionsgenerator aus, damit Sie den simulierten Frequenztyp und den Kurvenformtyp sowie auch den Amplituden- und Geräuschpegel des Signals ändern können. Beachten Sie die Veränderungen im Amplitudenspektrum.



Ende der Übung 15-1.

Den Frequenzgang von einem System berechnen

Die Messung des Frequenzinhalts von einzelnen Signalen ist an sich nützlich, aber man verwendet den Frequenzgang von Systemen vielfach zur Analyse des Verhaltens von Netzwerken aller Arten, von der Impedanz bei elektrischen Bauteilen bis zur Analyse der Eigenfrequenz von dynamischen Strukturen. Der Frequenzgang charakterisiert das Ansprechen von einem Netzwerk auf eine bestimmte Eingabe vollständig.

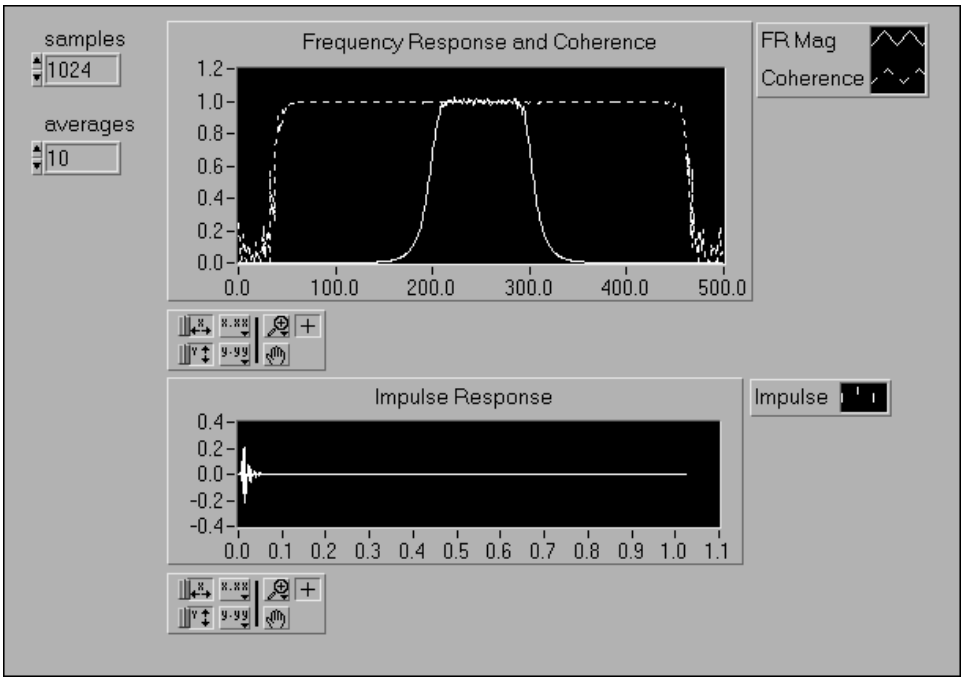


Übung 15-2. Den Frequenzgang und die Impulsantwort berechnen

Ihr Übungsziel ist es, den Frequenzgang und die Impulsantwort von einem System zu berechnen sowie die Kohärenz-Funktion zu berechnen und zu verstehen, wie die letztere zur Validierung von Ihren Messungen des Frequenzgangs eingesetzt wird.

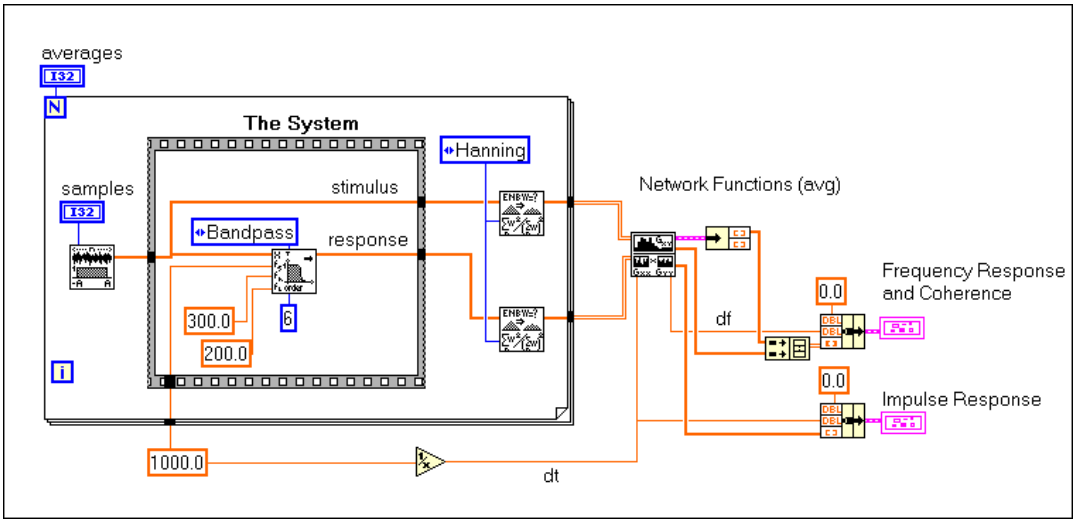
Frontpanel

1. Öffnen Sie ein neues Frontpanel, und fügen Sie die in der folgenden Abbildung dargestellten Objekte ein. Dieses Frontpanel zeigt den Betrag des Frequenzgangs und die Impulsantwort-Funktion für einen Bandpaßfilter. Die Kohärenz-Funktion wird im gleichen Maßstab dargestellt, weil sie auch eine spektrale Messung ist.



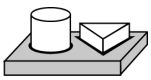
Blockdiagramm

2. Öffnen Sie das Blockdiagramm, und modifizieren Sie es, wie in der folgenden Abbildung dargestellt. Hier messen wir das Systemansprechen von einem Bandpaßfilter (VI Butterworth-Filter), indem wir weißes Rauschen (VI Uniformes Weißes Rauschen) als Systemerregung eingeben und die Filterausgabe als Systemansprechen aufnehmen. Erregung und Ansprechen werden beide durch das Hanning-Fenster (VI Skaliertes Zeitbereichfenster) geglättet und das Gesamtsystem wird über mehrere Rahmen oder *Mittelwerte* beobachtet. Die Erregungs- und Ansprechdaten werden an das VI Netzwerkfunktionen (avg) weitergeleitet, wo die eigentlichen Berechnungen bezüglich dem System-Frequenzgang durchgeführt werden.



Das VI Netzwerkfunktionen (avg) berechnet den Frequenzgang (Betrag und Phase), das Kreuzleistungsspektrum (Betrag und Phase), die Kohärenz-Funktion und die Impulsantwort. Durch die Erhöhung der Rahmenzahl für Ein- und Ausgabedaten (zunehmende Mittelwerte am Frontpanel) verbessern sich die Abschätzungen der Systemansprechfunktionen. In diesem Diagramm werden nur der Betrag des Frequenzgangs, die Kohärenz und die Impulsantwort angezeigt.

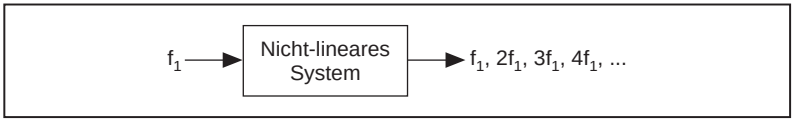
Die Kohärenz-Funktion mißt den Grad der Korrelation des Ausgabesignals mit dem Eingabesignal und bietet so eine Indiz der Gültigkeit von Ihrer Abschätzung des Frequenzgangs. Eingeführtes Rauschen und nicht-lineares Systemverhalten bei bestimmten Frequenzen lassen die Kohärenz-Funktion bei diesen Frequenzen unter eins sinken. Für unkorreliertes Systemrauschen nähert sich die Kohärenz-Funktion desto mehr eins an und die Abschätzung des Frequenzgangs wird desto besser, je mehr Mittelwerte genommen werden. Noch eine Tatsache, die man in Betracht ziehen sollte ist, daß die Kohärenz-Funktion nur definiert ist, wenn man mehr als einen Rahmen von Ein- und Ausgabedaten mittelt. Bei nur einem Mittelwert wird die Kohärenz eins für alle Frequenzen sein, auch wenn die Abschätzung des Frequenzgangs möglicherweise ungenau ist.



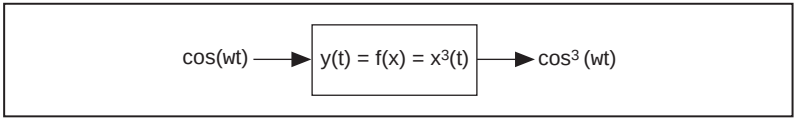
Ende der Übung 15-2.

Der Klirrfaktor

Wenn ein Signal, *x(t)*, von einer bestimmten Frequenz (z.B. f1) durch ein nicht-lineares System geleitet wird, besteht die Ausgabe des Systems nicht nur aus der eingegebenen Frequenz (f1), sondern auch aus deren Harmonischen (f2 = 2*f1, f3 = 3*f1, f4 = 4*f1 usw.). Die Anzahl der erzeugten Harmonischen und die entsprechenden Amplituden hängt von dem Grad der Nicht-Linearität des Systems ab. Es gilt im allgemeinen, daß je größer die Nicht-Linearität, desto höher sind die Harmonischen und umgekehrt.



Ein Beispiel von einem nicht-linearen System ist ein System, wo die Ausgabe *y(t)* die dritte Potenz des eingegebenen Signals *x(t)* ist.



Wenn also die Eingabe

$$x(t) = \cos(\omega t)$$

ist, so ist die Ausgabe

$$x^3(t) = 0{,}5*\cos(\omega t) + 0{,}25*[\ \cos(\omega t) + \cos(3\omega t)\ ]$$

Die Ausgabe enthält also nicht nur die eingegebene Grundfrequenz ω, sondern auch die dritte Harmonische 3ω.

Der Klirrfaktor

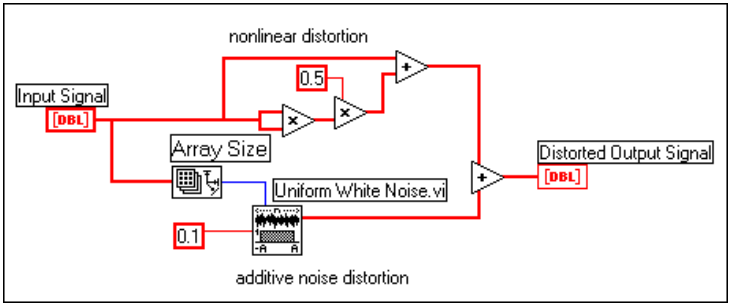
Zur Bestimmung des Grades der nicht-linearen Verzerrung, die ein System einführt, muß man die Amplituden der durch das System eingeführten Harmonischen in bezug auf die Amplitude der Fundamentalen messen. Der Klirrfaktor ist ein relatives Maß der Amplituden von den Harmonischen im Vergleich zur Amplitude der Fundamentalen. Wenn die Amplitude der Fundamentalen A_1 ist und die Amplituden der Harmonischen A_2 (zweite Harmonische), A_3 (dritte Harmonische), A_4 (vierte Harmonische), ...A_N (*Nte* Harmonische), dann ist der Klirrfaktor (THD) durch

$$THD = sqrt (A_1^2 + A_2^2 + A_3^2 + ... A_N^2)/A_1$$

gegeben und der Klirrfaktor als Prozentsatz (% THD) ist

$$\% THD = 100*sqrt (A_1^2 + A_2^2 + A_3^2 + ... A_N^2)/A_1$$

In der nächsten Übung werden Sie eine Sinus-Schwingung erzeugen und durch ein nicht-lineares System leiten. Das Blockdiagramm des nicht-linearen Systems ist unten dargestellt:



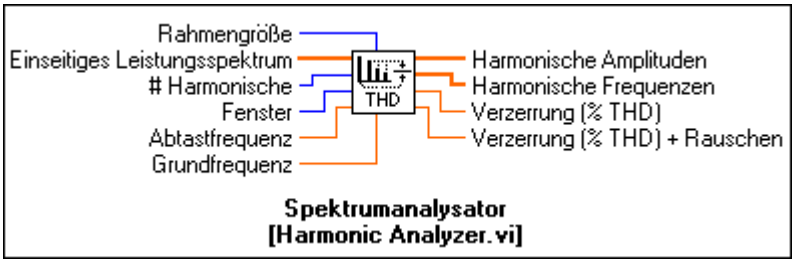
Überprüfen Sie vom Blockdiagramm her, daß für eine Eingabe x(t) = cos(ωt) die Ausgabe

$$\begin{aligned} y(t) &= \cos(\omega t) + 0{,}5\cos^2(\omega t) + 0{,}1n(t) \\ &= \cos(\omega t) + [1 + \cos(2\omega t)]/4 + 0{,}1n(t) \\ &= 0{,}25 + \cos(\omega t) + 0{,}25\cos(2\omega t) + 0{,}1n(t) \end{aligned}$$

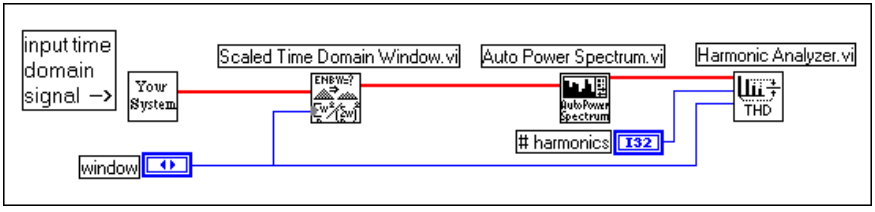
ist. Dieses nicht-lineare System erzeugt also eine zusätzliche DC-Komponente sowie auch die zweite Harmonische der Fundamentalen.

Das VI Spektrumanalysator benutzen

Sie können das VI Spektrumanalysator zur Berechnung der im Signal vorhandenen %THD am Ausgang des nicht-linearen Systems benutzen. Es findet die im Leistungsspektrum am Eingang vorhandenen fundamentalen und harmonischen Komponenten (ihre Amplituden und die entsprechenden Frequenzen) und berechnet den Klirrfaktor als Prozentsatz (%THD) und den Klirrfaktor plus Rauschen als Prozentsatz (%THD + Geräusch). Die Verbindungen zum VI **Spektrumanalysator** sind unten dargestellt:



Um dieses VI zu verwenden, müssen Sie ihm das Leistungsspektrum des Signals einspeisen, von dem es die THD berechnen soll. In diesem Beispiel müssen also die folgenden Verbindungen hergestellt werden:



Das VI Skaliertes Zeitbereichsfenster legt der Ausgabe y(t) des nicht-linearen Systems (**Ihr System**) ein Fenster auf. Dieses Signal wird dann an das VI **Einseitiges Leistungsspektrum** weitergeleitet, das dann das Leistungsspektrum von y(t) dem VI **Spektrumanalysator** sendet, das dann die Amplituden und Frequenzen der Harmonischen, die THD und die %THD berechnet.

Sie können die Zahl der vom VI zu findenden Harmonischen im Bedienelement **Anzahl der Harmonischen** spezifizieren. Ihre Amplituden und die entsprechenden Frequenzen werden in den Array-Anzeigeelementen **Amplituden der Harmonischen** und **Frequenzen der Harmonischen** zurückgegeben.

☞ **Hinweis** ***Die im Bedienelement Anzahl der Harmonischen spezifizierte Zahl schließt die Fundamentale ein. Wenn Sie also den Wert 2 im Bedienelement Anzahl der Harmonischen eingeben, bedeutet dies, daß die Fundamentale (von etwa freq f1) und die zweite Harmonische (von Frequenz f2 = 2*f1) zu finden sind. Wenn Sie einen Wert N eingeben, dann wird das VI die Fundamentale und die entsprechenden (N−1) Harmonischen finden.***

Nachfolgend finden Sie Erklärungen von einigen anderen Bedienelementen:

Fundamentalfrequenz ist eine Abschätzung der Frequenz von der fundamentalen Komponente. Wenn auf Null belassen (die Standardeinstellung), verwendet das VI die Frequenz der nicht-DC Komponente mit der höchsten Amplitude als die fundamentale Frequenz.

Fenster bezeichnet das Fenster, das Ihrem ursprünglichen zeitlichen Signal aufgelegt wurde. Es ist das Fenster, das Sie im VI **Skaliertes Zeitbereichfenster** auswählen. Für eine genaue Einschätzung der THD empfiehlt es sich, eine Windowing-Funktion auszuwählen. Die Standardeinstellung ist das uniforme Fenster.

Abtastrate ist die Abtastfrequenz der Eingabe in Hz.

Der Ausgang *% THD + Geräusch* bedarf der weiteren Erklärung. Die Berechnungen für *% THD + Geräusch* sind denen für *% THD* fast ähnlich, abgesehen davon, daß die Geräuschleistung mit derjenigen der Harmonischen addiert ist. Diese ist durch

$$\%\ \text{THD + Geräusch} = 100*\text{sqrt}\ (\ \text{sum(APS)})/\text{A1}$$

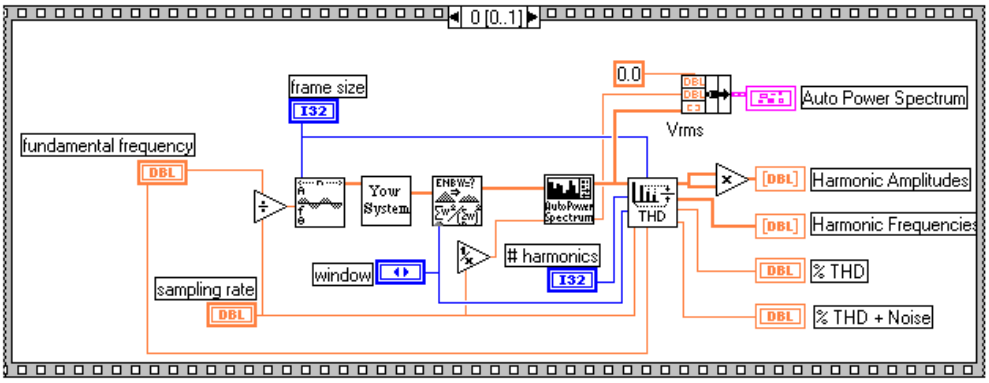
gegeben, wobei sum(APS) die Summe der Elemente des einseitigen Leistungsspektrum minus Elemente in der Nähe von DC und vom Index der fundamentalen Frequenz bedeutet.

Übung 15-3. Den Klirrfaktor berechnen

Ihr Übungsziel ist es, das VI Spektrumanalysator zur Berechnung vom Klirrfaktor zu verwenden.

Blockdiagramm

1. Öffnen Sie das THD Example VI aus `examples\analysis\measxmpl.llb`, und sehen Sie sich das Blockdiagramm an.

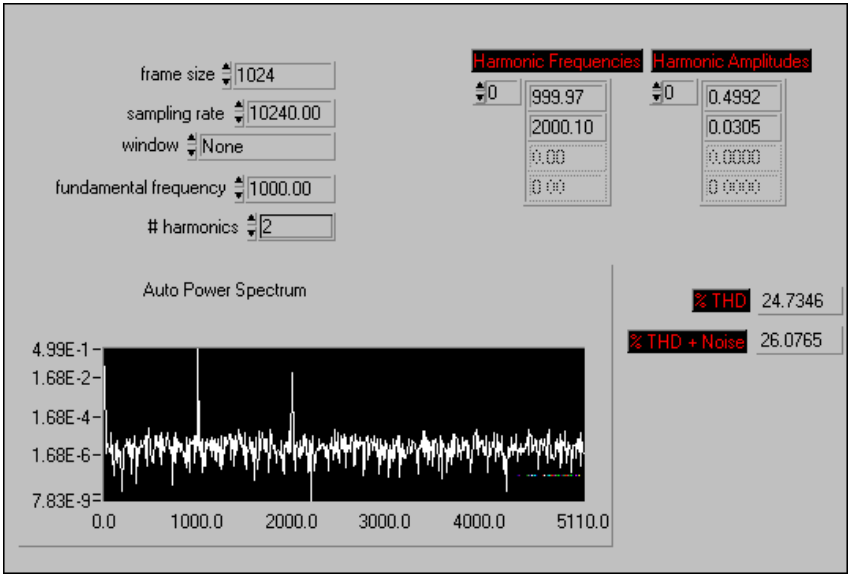


Manches hierin wird Ihnen schon vertraut sein. Ihr System ist das nicht-lineare System, das Sie schon vorher gesehen haben. Seine Ausgabe wird geglättet, das Leistungsspektrum berechnet und dem VI Spektrumanalysator zugeführt.

Das VI Sinus-Schwingung erzeugt eine Fundamentale der im Bedienelement **Fundamentalfrequenz** spezifizierten Frequenz.

Frontpanel

2. Schauen Sie sich das Frontpanel an. Unten sehen Sie eine Kurve vom Leistungsspektrum der Ausgabe des nicht-linearen Systems. Oben rechts sind die Array-Anzeigeelemente für die Frequenzen und Amplituden der Fundamentalen und ihrer Harmonischen. Die Größe des Arrays hängt von dem im Bedienelement **Anzahl der Harmonischen** eingegebenen Wert ab.



3. Ändern Sie die **Fundamentalfrequenz** auf 1000, die **Anzahl der Harmonischen** auf 2 und führen Sie das VI mehrmals aus. Notieren Sie sich jedesmal die Werte in den Ausgabe-Anzeigeelementen (**Frequenzen der Harmonischen**, **Amplituden der Harmonischen**, **% THD** und **% THD + Geräusch**).

   Warum bekommen Sie verschiedene Werte bei jeder Ausführung des VIs?

   Welcher Wert, % THD oder % THD + Geräusch, ist größer? Können Sie erklären warum?

4. Führen Sie das VI mit verschiedenen Auswahlen im Bedienelement **Fenster,** und beobachten Sie die Spitzen im Leistungsspektrum.

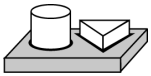
   Welches Fenster ergibt die schmalsten Spitzen? Welche die breitesten? Können Sie erklären warum?

5. Ändern Sie die fundamentale Frequenz auf 3000, und führen Sie das VI aus.

   Warum bekommen Sie einen Fehler?

   Hinweis: Betrachten Sie die Beziehung zwischen der Nyquist-Frequenz und der Frequenz der Harmonischen.

6. Wenn Sie fertig sind, schließen Sie das VI, und verlassen Sie LabVIEW.



Ende der Übung 15-3.

Zusammenfassung

Sie haben gesehen, daß die Messung-VIs allgemeine Messungsaufgaben durchführen können. Einige von diesen Aufgaben umfassen: die Berechnung des Amplituden- und Phasenspektrums von einem Signal und des Klirrfaktors. Andere VIs berechnen Eigenschaften von einem System, wie z.B. seine Transfer-Funktion, seine Impulsantwort, das Kreuzleistungsspektrum zwischen den Ein- und Ausgabesignalen usw. Die vorgefertigten VIs zur Durchführung von diesen Messungen sind in der Subpalette **Analyse»Messungen** zu finden.

16

Filterung

In diesem Kapitel wird beschrieben, wie unerwünschte Frequenzen aus Signalen mittels Infinite Impulse Response (Unendliche Impulsantwort) (IIR) Filtern, Finite Impulse Response (Endlichen Impulsantwort) (FIR) Filtern und nicht-linearen Filtern gefiltert werden. Beispiele zur Verwendung der Analysefilter-VIs entnehmen Sie bitte der Bibliothek `examples\analysis\fltrxmpl.llb`.

Einleitung in Digitalfilterungs-Funktionen

Der Entwurf von analogen Filtern stellt einen der wichtigsten Bereiche des Elektronikdesigns dar. Obwohl es Bücher über den Entwurf von Analogfiltern mit einfachen, getesteten Filterbeispielen gibt, ist der Entwurf von Filtern meist Spezialisten vorbehalten, da er fortgeschrittene mathematische Kenntnisse sowie ein gutes Verständnis der im System ablaufenden und den Filter beeinflussenden Vorgänge voraussetzt.

Moderne Werkzeuge zur Verarbeitung von Abtast- und Digitalsignalen ermöglichen es, bei Anwendungen, die Flexibilität und Programmierfähigkeit erfordern, Analog- durch Digitalfilter zu ersetzen. Dazu gehören Anwendungen in den Bereichen Audio, Telekommunikation, Geophysik und medizinische Überwachung.

Digitalfilter bieten gegenüber Analogfiltern die folgenden Vorteile:

- Sie sind softwareprogrammierbar.
- Sie sind stabil und vorhersehbar.
- Sie unterliegen keinen Temperatur- und Luftfeuchtigkeitsschwankungen und erfordern keine Präzisionselemente.
- Sie bieten ein hervorragendes Preis-Leistungs-Verhältnis.

Sie können Digitalfilter in LabVIEW verwenden, um Parameter wie Filterordnung, Grenzfrequenzen, Welligkeitsgrad und Sperrdämpfung zu steuern.

Die in diesem Abschnitt beschriebenen Digitalfilter-VIs entsprechen der Philosophie der virtuellen Instrumente. Die VIs handhaben alle Aspekte des Entwurfs, Berechnungen, Speicherverwaltung und die tatsächliche Datenfilterung intern und für den Anwender transparent. Sie müssen kein Experte auf dem Gebiet der Digitalfilter oder ihrer Theorie sein, um die Daten verarbeiten zu können.

Die folgende Erläuterung der Abtasttheorie soll Ihnen ein besseres Verständnis der Filterparameter und ihrer Beziehung zu den Eingabeparametern geben.

Das Abtasttheorem besagt, daß ein zeitkontinuierliches Signal aus diskreten, in gleichen Abständen gemessenen Abtastwerten rekonstruiert werden kann, wenn die Abtastfrequenz mindestens doppelt so groß wie die größte Frequenz des Zeitsignals ist. Nehmen Sie einmal an, daß Sie das entsprechende Zeitsignal in Δt gleichen Intervallen ohne Informationsverlust abtasten können. Der Parameter Δt stellt das Abtastintervall dar.

Sie erhalten die Abtastrate oder Abtastfrequenz f_s aus dem Abtastintervall

$$f_s = \frac{1}{\Delta t}.$$

Laut Abtasttheorem lautet die höchste Frequenz, die das digitale System folglich verarbeiten kann:

$$f_{Nyq} = \frac{f_s}{2}.$$

Die höchste Frequenz, die vom System verarbeitet werden kann, wird als Nyquist-Frequenz bezeichnet. Dies gilt ebenso für Digitalfilter. Wenn Ihr Abtastintervall beispielsweise

$$\Delta t = 0{,}001 \text{ sec}$$

beträgt, dann beträgt die Abtastfrequenz

$$f_s = 1000 \text{ Hz},$$

und die höchste Frequenz, die das System verarbeiten kann, ist

$$f_{Nyq} = 500 \text{ Hz}.$$

Die folgenden Arten von Filteroperationen basieren auf Filterentwurftechniken:

- Glätten von Fenstern
- Infinite Impulse Response (IIR) oder rekursive Digitalfilter
- Finite Impulse Response (FIR) oder nicht-rekursive Digitalfilter
- Nicht-lineare Filter

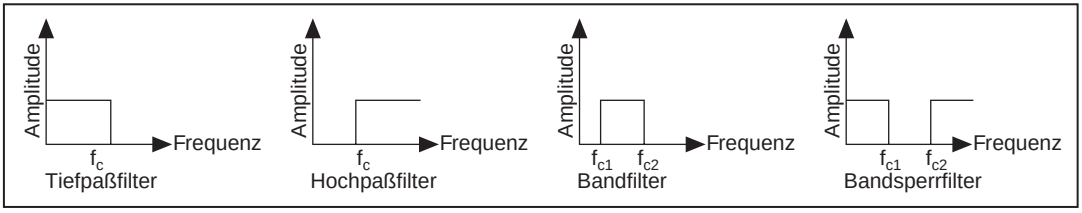
Der Rest dieses Kapitels präsentiert einen kurzen theoretischen Hintergrund zu den IIR-, FIR- und nicht-linearen Methoden und erörtert die Digitalfilter-VIs, die den jeweiligen Methoden entsprechen. Weitere Informationen zu den VIs, die die Fensterglättung implementieren entnehmen Sie bitte dem Kapitel 14, *Fensterglättung*.

Ideale Filter

Filter verändern oder entfernen unerwünschte Frequenzen. Je nach dem Frequenzbereich, den sie entweder durchlassen oder dämpfen, können sie in folgende Typen eingeteilt werden:

- *Ein Tiefpaßfilter* läßt tiefe Frequenzen durch und dämpft hohe Frequenzen.
- Ein *Hochpaßfilter* läßt hohe Frequenzen durch und dämpft tiefe Frequenzen.
- Ein *Bandfilter* läßt bestimmte Frequenzbänder durch.
- Ein *Bandsperrfilter* dämpft ein bestimmtes Frequenzband.

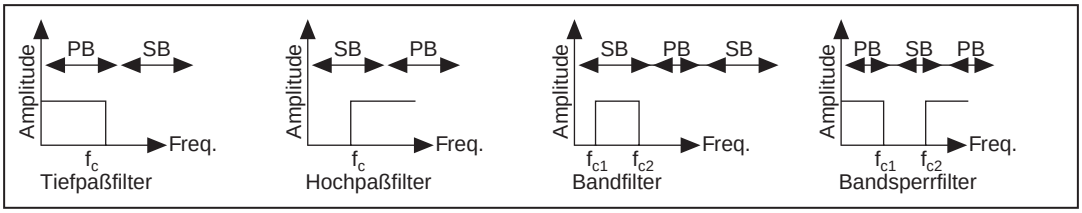
Das ideale Frequenzverhalten dieser Filter wird in der folgenden Abbildung gezeigt:



Das Tiefpaßfilter läßt alle Frequenzen unterhalb f_c durch, während das Hochpaßfilter alle Frequenzen oberhalb f_c durchläßt. Das Bandfilter läßt alle Frequenzen zwischen f_{c1} und f_{c2} durch, während das Bandsperrfilter alle Frequenzen zwischen f_{c1} und f_{c2} dämpft. Die Frequenzpunkte f_c, f_{c1}

und f_{c2} sind die Grenzfrequenzen des Filters. Beim Entwurf von Filtern müssen diese Grenzfrequenzen festgelegt werden.

Der Frequenzbereich, der durch das Filter gelassen wird, wird als *Durchlaßbereich* (Passband = PB) des Filters bezeichnet. Ein idealer Filter besitzt eine Verstärkung von Eins (0 dB) im Durchlaßbereich, so daß die Amplitude des Signals weder ansteigt noch abfällt. Der *Sperrbereich* (SB) entspricht dem Frequenzbereich, der überhaupt nicht durch den Filter gelassen und zurückgewiesen (gedämpft) wird. Durchlaß- und Sperrbereich der verschiedenen Filtertypen sind in der folgenden Abbildung dargestellt:

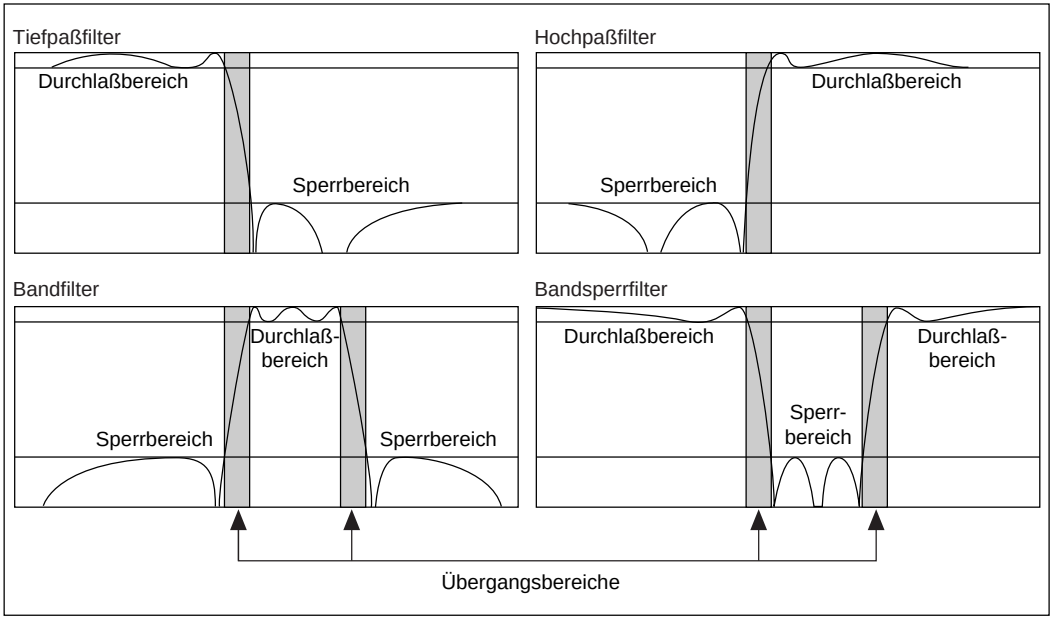


Beachten Sie, daß Tiefpaß- und Hochpaßfilter über einen Durchlaß- und einen Sperrbereich verfügen, während das Bandfilter einen Durchlaß- und zwei Sperrbereiche und das Bandsperrfilter zwei Durchlaß- und einen Sperrbereich aufweist.

Praktische (nicht-ideale) Filter

Der Übergangsbereich

Im Idealfall sollte ein Filter die Verstärkung Eins (0 dB) im Durchlaßbereich und die Verstärkung Null (-unendlich dB) im Sperrbereich aufweisen. Bei einer echten Implementierung können jedoch nicht alle diese Kriterien erfüllt werden. In der Praxis gibt es stets einen endlichen Übergangsbereich zwischen dem Durchlaß- und dem Sperrbereich. In diesem Bereich ändert sich die Verstärkung des Filters graduierlich von 1 (0 dB) im Durchlaßbereich bis auf 0 (-unendlich) im Sperrbereich. Die folgenden Diagramme stellen Durchlaß-, Sperr- und Übergangsbereich (TR) für die verschiedenen Typen von nicht-idealen Filtern dar. Beachten Sie, daß der Frequenzbereich des Durchlaßbereichs der Bereich ist, in dem die Verstärkung des Filters von 0 dB bis –3 dB variiert.



Welligkeit im Durchlaßbereich und Sperrdämpfung

Bei vielen Anwendungen ist es durchaus akzeptabel, wenn die Verstärkung im Durchlaßbereich ein wenig von Eins abweicht. Diese Abweichung im Durchlaßbereich wird *Welligkeit im Durchlaßbereich* genannt und ist die Differenz zwischen der tatsächlichen Verstärkung und der gewünschten Verstärkung Eins. Die *Sperrdämpfung* kann in der Praxis nicht unendlich sein, und Sie müssen einen für Sie zufriedenstellenden Wert dafür festlegen. Sowohl die Welligkeit im Durchlaßbereich als auch die Sperrdämpfung werden in Dezibel bzw. dB gemessen und sind durch folgende Gleichung definiert:

$$dB = 20*\log_{10}(A_o(f)/A_i(f)),$$

wobei $\log_{10}$ den Logarithmus zum Basiswert 10 bezeichnet und $A_i(f)$ und $A_o(f)$ die Amplituden einer bestimmten Frequenz *f* vor bzw. nach der Filterung darstellen.

Beispielsweise sieht die Formel für eine –0,02 dB Welligkeit im Durchlaßbereich folgendermaßen aus:

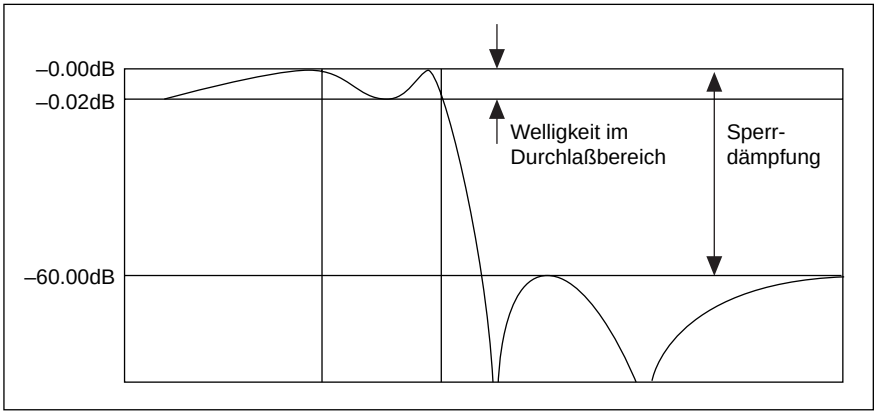
$$-0{,}02 = 20*\log_{10}(A_o(f)/A_i(f))$$
$$A_o(f)/A_i(f) = 10^{-0{,}001} = 0{,}9977$$

Dies zeigt, daß das Verhältnis der Eingangs- und Ausgangsamplituden nahe Eins liegt.

Bei einer –60 dB Sperrdämpfung sieht die Formel folgendermaßen aus:

$$-60 = 20*\log_{10}(A_o(f)/A_i(f))$$
$$A_o(f)/A_i(f) = 10^{-3} = 0{,}001$$

Das bedeutet, daß die Ausgangsamplitude 1/1000 der Eingangsamplitude beträgt. Dieses Konzept ist in der folgenden Abbildung dargestellt, die jedoch nicht maßstabsgerecht ist.

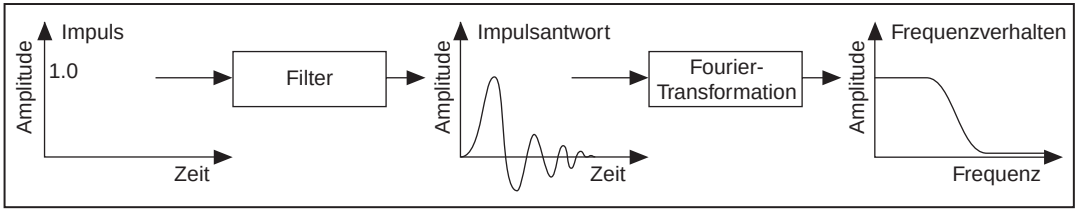


Hinweis ***Die Dämpfung wird üblicherweise in Dezibel ohne Minuszeichen angegeben, wobei normalerweise von einem negativen dB-Wert ausgegangen wird.***

IIR- und FIR-Filter

Eine weitere Methode zur Klassifizierung von Filtern basiert auf ihrer Impulsantwort. Was ist eine Impulsantwort? Die Reaktion eines Filters auf eine Eingabe in Form eines Impulses ($x[0] = 1$ und $x[i] = 0$ für alle $i \neq 0$) wird *Impulsantwort* des Filters genannt (siehe Abbildung unten). Die Fourier-Transformierte der Impulsantwort wird *Frequenzverhalten* des Filters genannt. Das Frequenzverhalten eines Filters gibt Aufschluß über den Ausgang des Filters bei verschiedenen Frequenzen. Das heißt, es gibt die Verstärkung des Filters bei verschiedenen Frequenzen an. Bei einem idealen Filter beträgt die Verstärkung im Durchlaßbereich 1 und im Sperrbereich 0. Alle Frequenzen im Durchlaßbereich werden demgemäß

unverändert an den Ausgang durchgelassen, während es für Frequenzen im Sperrbereich keinen Ausgang gibt.



Fällt die Impulsantwort des Filters nach einer endlichen Zeitspanne auf Null ab, wird dieser Filter *Finite Impulse Response (FIR)* Filter genannt. Besteht die Impulsantwort jedoch unendlich fort, handelt es sich um einen *Infinite Impulse Response (IIR)* Filter. Ob die Impulsantwort endlich oder unendlich ist (d.h., ob es sich um einen FIR- oder IIR-Filter handelt) hängt von der Berechnung des Ausgangs ab.

Der grundlegende Unterschied zwischen FIR- und IIR-Filtern besteht darin, daß die Ausgabe bei FIR-Filtern nur von den aktuellen und vorherigen Eingabewerten abhängt, während die Ausgabe bei IIR-Filtern nicht nur von den aktuellen und vorherigen Eingabewerten sondern auch von den vorherigen Ausgabewerten abhängt.

Ein Beispiel ist eine Registrierkasse in einem Supermarkt. *x*[*k*] ist der Preis eines Artikels, der gerade von einem Kunden gekauft wird, und x(k-1) ist der Preis des vorherigen Artikels, wobei $1 \leq k \leq N$ gilt und *N* die Gesamtanzahl der Artikel ist. Die Kasse addiert den Preis jedes Artikels und produziert so eine “laufende” Summe. Diese “laufende” Summe *y*[k] wird bis zum k^{ten} Artikel durch die folgende Gleichung wiedergegeben:

$$y[k] = x[k] + x[k–1] + x[k–2] + x[k–3] + ...+ x[1] \qquad (16\text{-}1A)$$

Die Summe von *N* Artikeln beträgt demnach *y[N]*. Da *y[k]* die Summe bis zum k^{ten} Artikel und *y[k–1]* die Summe bis zum $(k–1)^{ten}$ Artikel ist, kann die Gleichung 16-1A auch durch folgende ersetzt werden:

$$y[k] = y[k–1] + x[k] \qquad (16\text{-}1B)$$

Wenn Sie eine Umsatzsteuer von 8,25% hinzufügen, können die Gleichungen 16-1A und 16-1B folgendermaßen wiedergegeben werden:

$$y[k] = 1{,}0825x[k] + 1{,}0825x[k–1] + 1{,}0825\,x[k–2] + 1{,}0825x[k–3] + ... + 1{,}0825x[1] \quad (16\text{-}2A)$$

$$y[k] = y[k–1] + 1{,}0825x[k] \quad (16\text{-}2B)$$

Beachten Sie, daß die beiden Gleichungen 16-2A und 16-2B in der Beschreibung der Kasse übereinstimmen. Der Unterschied besteht darin, daß 16-2A nur bezüglich der Eingaben und 16-2B bezüglich Ein- und Ausgaben implementiert wird. Die Gleichung 16-2A ist die *nicht-rekursive* bzw. FIR-Implementierung. Die Gleichung 16-2B ist die *rekursive* bzw. IIR-Implementierung.

Filterkoeffizienten

Bei Gleichung 16-2A lautet die Multiplikationskonstante für jeden Term 1,0825, während diese Konstanten bei Gleichung 16-2B 1 (für *y[k–1])* und 1,0825 (für *x[k]*) lauten. Diese Multiplikationskonstanten werden *Koeffizienten* des Filters genannt. Bei einem IIR-Filter werden die Koeffizienten, die mit den Eingaben multipliziert werden, als *Vorwärts-Koeffizienten* und die, die mit den Ausgaben multipliziert werden, als *Rückwärts-Koeffizienten* bezeichnet.

Gleichungen der Formen 16-1A, 16-1B, 16-2A oder 16-2B, welche die Arbeitsweise des Filters beschreiben, werden *Differentialgleichungen* genannt.

Der Nachteil von IIR-Filtern liegt in einem nicht-linearen Phasengang. Wenn die Anwendung keine Phaseninformationen erfordert, wie z.B. eine einfache Signalüberwachung, können IIR-Filter durchaus geeignet sein. Bei Anwendungen, die lineare Phasengänge voraussetzen, sollten Sie FIR-Filter einsetzen. Aufgrund ihrer rekursiven Art sind der Entwurf und die Implementierung von IIR-Filtern schwieriger.

Infinite Impulse Response Filter

Infinite Impulse Response Filter (IIR) sind Digitalfilter mit Impulsantworten, die theoretisch von unendlicher Länge (Dauer) sein können. Die allgemeine Differentialgleichung für IIR-Filter lautet:

$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{N_b-1} b_j x_{i-j} - \sum_{k=1}^{N_a-1} a_k y_{i-k}\right), \qquad (16\text{-}3)$$

wobei N_b für die Anzahl der *Vorwärts*-Koeffizienten (b_j) und N_a für die Anzahl der *Rückwärts*-Koeffizienten (a_k) steht.

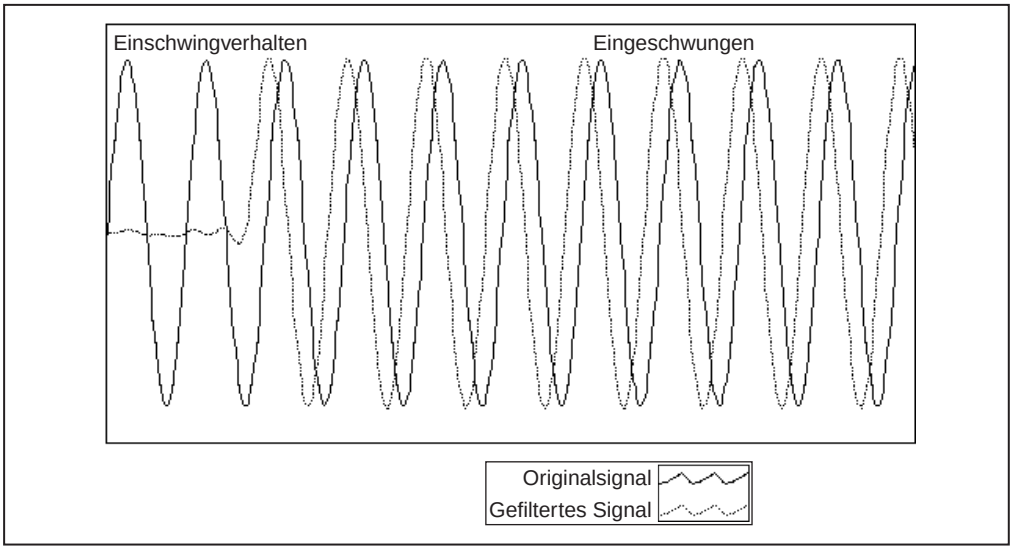
Beim Entwurf der meisten IIR-Filter (und bei allen LabVIEW IIR-Filtern) beträgt der Koeffizient a_0 1. Der Ausgabeabtastwert beim aktuellen Abtastwertindex i entspricht der Summe der skalierten aktuellen und vorherigen Eingaben (x_i und x_{i-j} wenn $\neq 0$) und der skalierten vorherigen Ausgaben (y_{i-k}). Aus diesem Grund sind IIR-Filter auch als rekursive oder Autoregressive Moving-Average (ARMA) Filter bekannt.

Die Antwort eines allgemeinen IIR-Filters auf einen Impuls ($x_0 = 1$ und $x_i = 0$ für alle $i \neq 0$) wird Impulsantwort des Filters genannt. Die von Gleichung 16-3 beschriebene Impulsantwort des Filters ist bei Koeffizienten, die von Null abweichen, tatsächlich von unendlicher Dauer. Bei Filteranwendungen in der Praxis klingt die Impulsantwort eines stabilen IIR-Filters jedoch bei einer endlichen Anzahl von Abtastwerten auf nahe Null ab.

Die IIR-Filter in LabVIEW verfügen über folgende Eigenschaften:

- Für aus Gleichung 16-3 resultierende negative Indizes wird Null vorausgesetzt, wenn das VI zum ersten mal aufgerufen wird.
- Da für den Anfangszustand des Filters Null vorausgesetzt wird (negative Indizes), tritt ein zur Filterordnung proportionaler Einschwingzustand auf, bevor der Filter den Dauerzustand erreicht. Die Dauer des Einschwingverhaltens bzw. der Verzögerung für Tiefpaß- und Hochpaßfilter entspricht der Filterordnung.
- Verzögerung = Filterordnung.
- Die Dauer des Einschwingverhaltens für Band- und Bandsperrfilter entspricht der doppelten Filterordnung.
- Verzögerung = 2* Ordnung.

Das Einschwingverhalten kann bei nachfolgenden Aufrufen beseitigt werden, indem der Zustandsspeicher aktiviert wird. Zum Aktivieren des Zustandsspeichers müssen Sie das Bedienelement **init/cont** des VIs auf TRUE (kontinuierliche Filterung) setzen.



Die Anzahl der Elemente in der gefilterten Sequenz entspricht der Anzahl der Elemente in der Eingabesequenz.

Nach Abschluß der Filterung behält das Filter die Werte des internen Filterzustands bei.

Der Vorteil digitaler IIR-Filter gegenüber FIR-Filtern liegt darin, daß sie normalerweise weniger Koeffizienten zur Durchführung ähnlicher Filteroperationen erfordern. Deshalb laufen IIR-Filter viel schneller ab, und sie erfordern keinen zusätzlichen Speicher, da sie an Ort und Stelle ablaufen.

Der Nachteil von IIR-Filtern ist ein nicht-linearer Phasengang. Wenn die Anwendung keine Phaseninformationen erfordert, wie z.B. eine einfache Signalüberwachung, können IIR-Filter durchaus geeignet sein. Bei Anwendungen, die lineare Phasengänge voraussetzen, sollten Sie FIR-Filter einsetzen.

Kaskadenförmige IIR-Filterung

Filter, die implementiert werden, indem sie die in Gleichung 16-4 beschriebene Struktur direkt verwenden, werden *Direktform*-IIR-Filter genannt. Direktform-Implementierungen sind oft anfällig für Fehler, die durch Koeffizientenquantisierung und Berechnungs- und Genauigkeitstoleranzen eingeführt werden. Darüber hinaus kann ein Filter, der stabil konzipiert wurde, mit erhöhter Koeffizientenlänge, die zur Filterordnung proportional ist, unstabil werden.

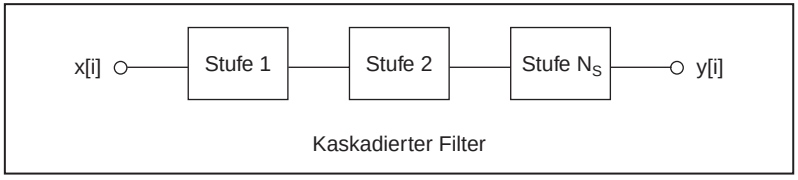
Eine weniger anfällige Struktur kann erzielt werden, indem die Direktform-Übertragungsfunktion in Abschnitte von niedrigerer Ordnung bzw. Filterstufen zerlegt wird. Die in der Gleichung 16-4 (mit $a_0 = 1$) angegebene Direktform-Übertragungsfunktion des Filters kann wie folgt als das Verhältnis von *z* Transformierten wiedergegeben werden:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{N_b - 1} z^{-(N_b - 1)}}{1 + a_1 z^{-1} + \ldots + a_{N_a - 1} z^{-(N_a - 1)}} \quad . \tag{16-4}$$

Durch Faktorisierung der Gleichung 16-4 in Abschnitte zweiter Ordnung wird die Übertragungsfunktion des Filters zu einem Produkt von Filterfunktionen zweiter Ordnung,

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 + a_{1k} z^{-1} + a_{2k} z^{-2}} \quad , \tag{16-5}$$

wobei $N_s = \lfloor N_a/2 \rfloor$ die größte Ganzzahl $\leq N_a/2$ und $N_a \geq Nb$ (N_s ist die *Anzahl der Stufen*) ist. Diese neue Filterstruktur kann als *Kaskade* von Filtern zweiter Ordnung beschrieben werden.



Jede einzelne Stufe wird mit Hilfe der Direktform-II-Filterstruktur implementiert, da sie eine Mindestanzahl von arithmetischen Operationen und Verzögerungselementen (interne Filterstufen) voraussetzt. Jede Stufe verfügt über einen Eingang, einen Ausgang und zwei vergangene interne Zustände ($s_k[i–1]$ und $s_k[i–2]$).

Wenn *n* die Anzahl der Abtastwerte der Eingabesequenz ist, läuft die Filteroperation wie in folgender Gleichung dargestellt ab:

$$y_0[i] = x[i],$$

$$s_k[i] = y_{k-1}[i-1] - a_{1k}s_k[i-1] - a_{2k}s_k[i-2],\ k = 1, 2,..., N_s$$

$$y_k[i] = b_{Ok}s_k[i] + b_{1k}s_k[i-1] + b_{2k}s_k[i-2],\ k = 1, 2,..., N_s$$

$$y[i] = y_{Ns}[i],$$

wobei für jeden Abtastwert folgendes gilt:

$$i = 0, 1, 2,..., n-1.$$

Bei Filtern mit einer einzigen Grenzfrequenz (Tiefpaß- und Hochpaßfilter) können Filterstufen zweiter Ordnung direkt entworfen werden. Das gesamte IIR-Tiefpaß- oder Hochpaßfilter enthält kaskadierte Filter zweiter Ordnung.

Für Filter mit zwei Grenzfrequenzen (Band- und Sperrbandfilter) werden eher Filterstufen vierter Ordnung eingesetzt. Das gesamte IIR-Band- oder Sperrbandfilter enthält kaskadierte Filter vierter Ordnung. Die Filteroperation für Stufen vierter Ordnung verläuft gemäß folgender Gleichungen:

$$y_0[i] = x[i],$$

$$s_k[i] = y_{k-1}[i-1] - a_{1k}s_k[i-1] - a_{2k}s_k[i-2] - a_{3k}s_k[i-3] - a_{4k}s_k[i-4],\ k = 1, 2,..., N_s$$

$$y_k[i] = b_{0k}s_k[i] + b_{1k}s_k[i-1] + b_{2k}s_k[i-2] + b_{3k}s_k[i-3] + b_{4k}s_k[i-4],\ k = 1, 2,..., N_s$$

$$y[i] = y_{Ns}[i].$$

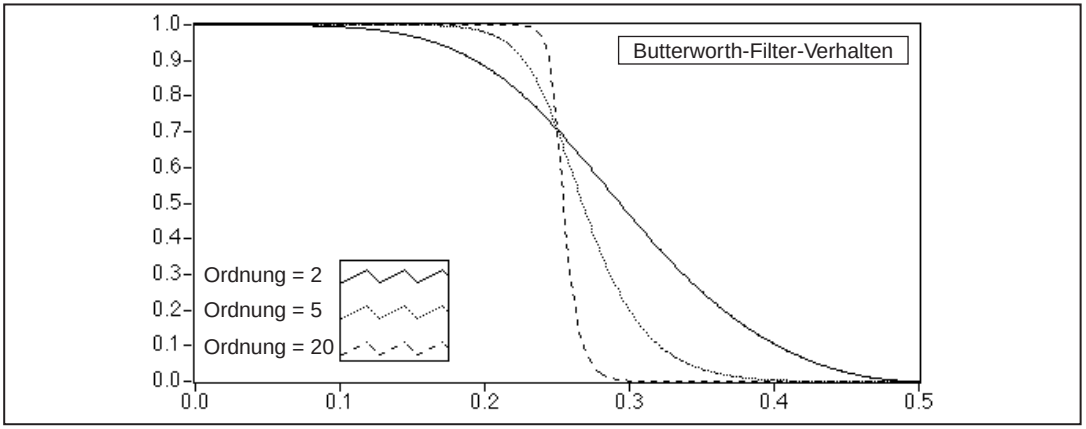
Beachten Sie, daß bei Filterstufen vierter Ordnung $N_s = \lfloor (N_a + 1)/4 \rfloor$ gilt.

Butterworth-Filter

Butterworth-Filter werden von einem glatten Frequenzgang bei allen Frequenzen und einem monotonen Abklingen von der festgelegten Grenzfrequenz charakterisiert. Butterworth-Filter haben einen maximal flachen Frequenzgang, was in einem idealen Verhalten von Eins im

Durchgangsbereich und 0 im Sperrbereich resultiert. Die Frequenz von halber Leistung bzw. die Grenzfrequenz für 3 dB Abfall entspricht den festgelegten Grenzfrequenzen.

In der folgenden Abbildung ist das Verhalten eines Tiefpaß-Butterworth-Filters dargestellt. Der Vorteil von Butterworth-Filtern liegt in einem glatten, monoton abklingenden Frequenzgang. Nachdem Sie die Grenzfrequenz bestimmt haben, legt LabVIEW die *Steilheit* des Übergangs proportional zur Filterordnung fest. Butterworth-Filter von höherer Ordnung erreichen nahezu das ideale Tiefpaß-Filterverhalten.



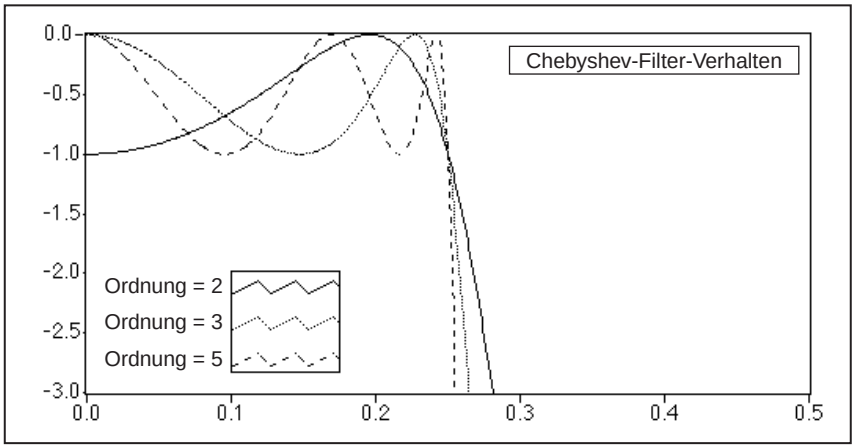
Chebyshev-Filter

Butterworth-Filter sorgen aufgrund der langsamen Dämpfung zwischen dem Durchlaßbereich (dem interessanten Teil des Spektrums) und dem Sperrbereich (dem unerwünschten Teil des Spektrums) nicht immer für eine gute Annäherung des idealen Filterverhaltens.

Chebyshev-Filter minimieren Spitzenfehler im Durchlaßbereich, indem sie die maximal zulässige Abweichung zwischen dem idealen Filter und dem gewünschten Filterverhalten (maximaler tolerierbarer Fehler im Durchlaßbereich) berücksichtigen. Die Charakteristiken des Frequenzverhaltens bei Chebyshev-Filtern sind ein Equi-Ripple-Amplitudenverhalten im Durchlaßbereich, ein monoton abklingendes Amplitudenverhalten im Sperrbereich und eine steilere Dämpfung als bei Butterworth-Filtern.

Im folgenden Graphen ist das Verhalten eines Tiefpaß-Chebyshev-Filters dargestellt. Beachten Sie, daß das Equi-Ripple-Verhalten im Durchlaßbereich vom maximal tolerierbaren Welligkeitsfehler begrenzt

wird und daß die steile Dämpfung im Sperrbereich erscheint. Der Vorteil von Chebyshev-Filtern gegenüber Butterworth-Filtern liegt in einem deutlicheren Übergang zwischen Durchlaßbereich und Sperrbereich bei einem Filter niedriger Ordnung. Dadurch werden kleinere absolute Fehler und höhere Ablaufgeschwindigkeiten erzielt.

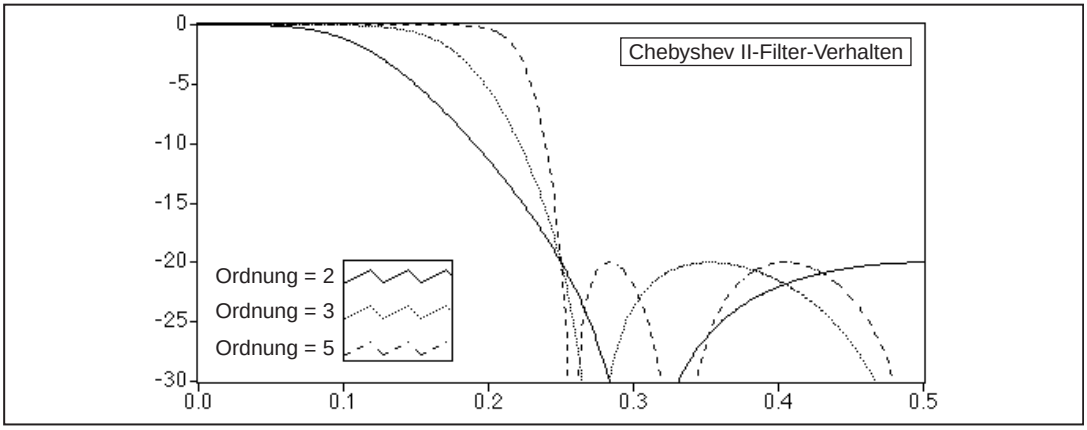


Chebyshev II oder inverse Chebyshev-Filter

Chebyshev-II-Filter, die auch als inverse Chebyshev- bzw. Typ II Chebyshev-Filter bekannt sind, ähneln Chebyshev-Filtern, mit der Ausnahme, daß sie den Fehler über den Sperrbereich (anstatt über den Durchlaßbereich) verteilen und im Durchlaßbereich (anstatt im Sperrbereich) einen maximal flachen Frequenzgang aufweisen.

Chebyshev-II-Filter minimieren Spitzenwertfehler im Sperrbereich, indem sie den maximalen absoluten Wert der Differenz zwischen dem idealen und dem gewünschten Filterverhalten berücksichtigen. Die Charakteristiken des Frequenzverhaltens bei Chebyshev-II-Filtern sind ein Equi-Ripple-Amplitudenverhalten im Sperrbereich, ein monoton abklingendes Amplitudenverhalten im Durchlaßbereich und eine steilere Dämpfung als bei Butterworth-Filtern.

Im folgenden Graphen ist das Verhalten eines Tiefpaß-Chebyshev-II-Filters dargestellt. Beachten Sie, daß das Equi-Ripple-Verhalten im Sperrbereich vom maximal tolerierbaren Fehler begrenzt wird und daß die sanfte monotone Dämpfung im Sperrbereich erscheint. Der Vorteil von Chebyshev-II-Filtern gegenüber Butterworth-Filtern liegt in einem deutlicheren Übergang zwischen dem Durchlaßbereich und dem Sperrbereich bei einem Filter niedriger Ordnung. Dadurch werden kleinere absolute Fehler und höhere Ablaufgeschwindigkeiten erzielt. Einer der Vorteile von Chebyshev-II-Filtern gegenüber normalen Chebyshev-Filtern ist die Verteilung des Fehlers über den Sperrbereich anstatt über den Durchlaßbereich.

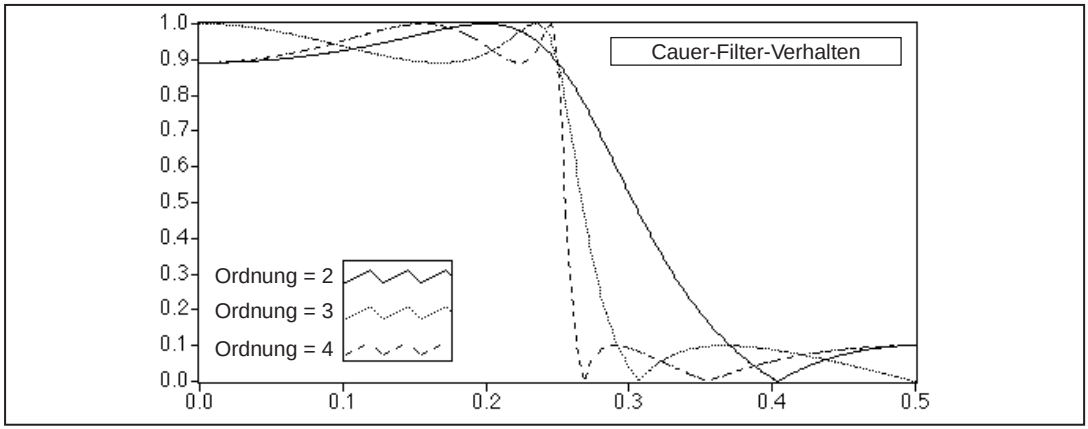


Elliptische bzw. Cauer-Filter

Cauer-Filter minimieren den Spitzenwertfehler, indem sie ihn über den Durchlaß- und den Sperrbereich verteilen. Das Amplitudenverhalten von Cauer-Filtern wird durch konstante Welligkeit im Durchlaß- und Sperrbereich charakterisiert. Im Vergleich zu Butterworth- und Chebyshev-Filtern der gleichen Ordnung bietet das Cauer-Filter den deutlichsten Übergang zwischen Durchlaß- und Sperrbereich. Aus diesem Grund werden Cauer-Filter vielfach verwendet.

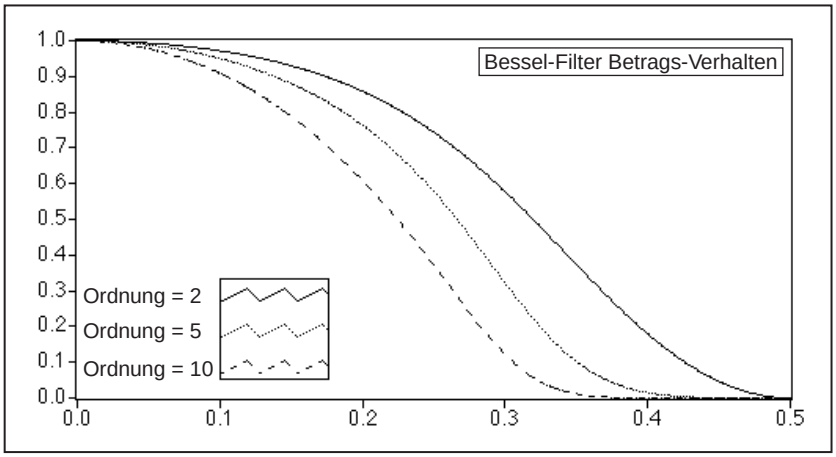
Im folgenden Graphen ist das Verhalten eines Tiefpaß-Cauer-Filters dargestellt. Beachten Sie, daß die Welligkeit im Durchlaß- und im Sperrbereich durch den gleichen maximal tolerierbaren Fehler (durch den

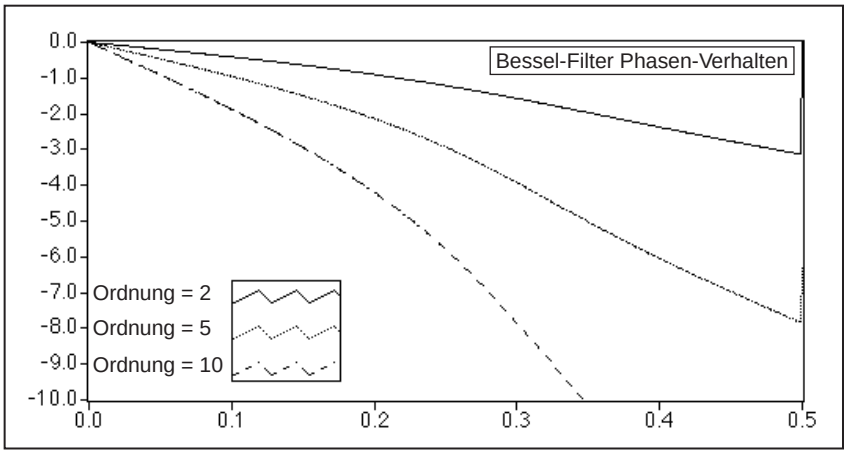
Welligkeitsgrad in dB bestimmt) begrenzt ist. Beachten Sie darüber hinaus die deutliche Übergangskante, selbst bei Cauer-Filtern niedriger Ordnung.



Bessel-Filter

Bessel-Filter werden zur Reduzierung von nicht-linearen Phasenverzerrungen, die ein Merkmal von IIR-Filtern sind, eingesetzt. Bei Filtern höherer Ordnung und Filtern mit einer steileren Dämpfung tritt diese Bedingung verstärkt auf, vor allem in den Übergangsbereichen der Filter. Bessel-Filter haben einen maximal flachen Frequenzgang sowohl in der Amplitude als auch der Phase. Darüber hinaus ist der Phasengang im Durchlaßbereich, dem interessanten Bereich, von Bessel-Filtern nahezu linear. Wie Butterworth-Filter erfordern Bessel-Filter zur Minimierung von Fehlern den Einsatz von Filtern hoher Ordnung und werden deshalb nicht häufig eingesetzt. Ein linearer Phasengang kann auch mit FIR-Filter-Entwürfen erzielt werden. Im folgenden Graphen ist das Verhalten eines Tiefpaß-Bessel-Filters dargestellt. Beachten Sie, daß der Frequenzgang bei allen Frequenzen glatt ist und sowohl für die Amplitude als auch die Phase monoton abklingt. Beachten Sie ferner, daß die Phase im Durchlaßbereich nahezu linear ist.


Bessel-Filter Betrags-Verhalten
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Ordnung = 2
Ordnung = 5
Ordnung = 10
0.0
0.1
0.2
0.3
0.4
0.5


Bessel-Filter Phasen-Verhalten
0.0
-1.0
-2.0
-3.0
-4.0
-5.0
-6.0
-7.0
-8.0
-9.0
-10.0
Ordnung = 2
Ordnung = 5
Ordnung = 10
0.0
0.1
0.2
0.3
0.4
0.5

Finite Impulse Response Filter

Finite Impulse Response (FIR) Filter sind Digitalfilter, die eine endliche Impulsantwort aufweisen. FIR-Filter sind auch unter der Bezeichnung nicht-rekursive, Faltungs- oder Moving-Average (MA) Filter bekannt, da der Ausgang eines FIR-Filters als endliche Faltung beschrieben werden kann:

$$y_i = \sum_{k=0}^{n-1} h_k x_{i-k},$$

wobei *x* für die zu filternde Eingabesequenz und *y* für die gefilterte Ausgabesequenz steht, während *h* die Koeffizienten des FIR-Filters darstellt.

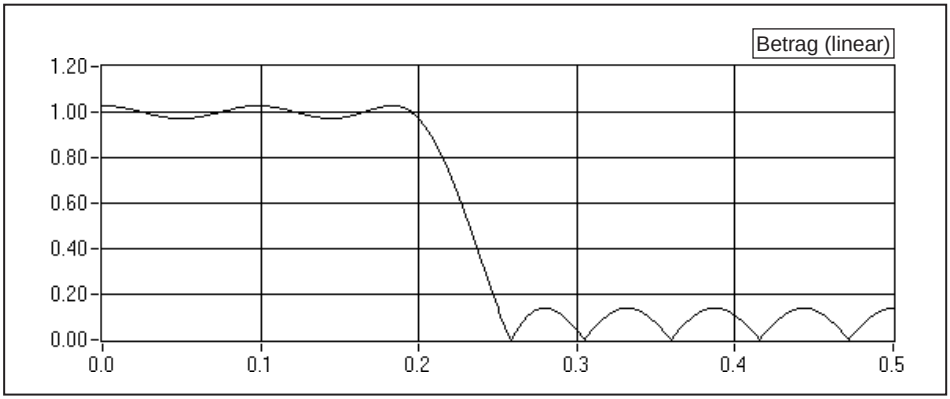
Die folgende Liste enthält die wichtigsten Charakteristiken von FIR-Filtern:

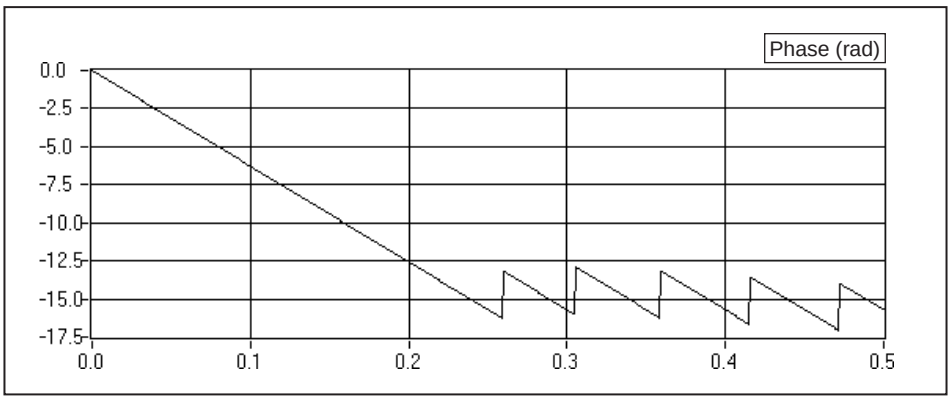
- Dank der Symmetrie der Filter-Koeffizienten in der Praxis können sie eine lineare Phase erreichen.
- Sie sind grundsätzlich stabil.
- Die Filterfunktion kann mit Hilfe der Faltung durchgeführt werden und deshalb normalerweise mit einer Verzögerung der Ausgabesequenz assoziiert werden.

  $$\text{Verzögerung} = \frac{n-1}{2}$$

  wobei *n* die Anzahl der Koeffizienten des FIR-Filters darstellt.

In den folgenden Abbildungen ist ein typisches Amplituden- und Phasenverhalten von FIR-Filtern gegenüber einer normalisierten Frequenz dargestellt.





Die Unstetigkeiten im Phasengang werden von den Unstetigkeiten hervorgerufen, die sich ergeben, wenn das Amplitudenverhalten unter Verwendung des Absolutwerts berechnet wird. Beachten Sie, daß die Unstetigkeiten in der Phase der Größe *Pi* entsprechen. Die Phase ist jedoch deutlich linear. Siehe Anhang A, *Analyse-Referenzen*, für weitere Informationen zu diesem Thema.

Sie können ein FIR-Filter entwerfen, indem Sie ein festgelegtes, gewünschtes Frequenzverhalten eines zeitdiskreten Systems nähern. Die am häufigsten verwendeten Methoden nähern das gewünschte Amplitudenverhalten, während sie einen linearen Phasengang beibehalten.

Entwerfen von FIR-Filtern durch Fensterung

Die einfachste Methode zum Entwerfen von FIR-Filtern mit linearer Phase ist die Methode der *Fensterung*. Für den Entwurf eines FIR-Filters mittels Fensterung beginnen Sie bei einem idealen Frequenzverhalten, berechnen seine Impulsantwort und machen diese anschließend ganzzahlig, um eine endliche Anzahl von Koeffizienten zu erhalten. Um die lineare Phasen-Begrenzung zu erreichen, muß die Symmetrie um die zentralen Koeffizienten beibehalten werden. Die Verkürzung der idealen Impulsantwort hat einen Effekt namens Gibbssches Phänomen zur Folge, ein Schwingungsverhalten in der Nähe von abrupten Übergängen (Grenzfrequenzen) im Frequenzverhalten des FIR-Filters.

Die Auswirkungen des Gibbsschen Phänomens können verringert werden, indem die Verkürzung der idealen Impulsantwort mit Hilfe der Fensterglättungsfunktion geglättet wird. Die Höhe der Nebenkeulen im Frequenzverhalten kann verringert werden, indem beide Enden mit Hilfe der FIR-Koeffizienten verjüngt werden. Der Nachteil bei dieser Methode ist jedoch, daß die Hauptkeule breiter wird und einen breiteren Übergangsbereich an den Grenzfrequenzen zur Folge hat. Die Wahl einer Fensterfunktion ähnelt in diesem Fall der Wahl zwischen Chebyshev- und Butterworth-IIR-Filtern insofern, als daß sie einen Kompromiß zwischen den Nebenkeulenpegeln in der Nähe der Grenzfrequenzen und der Breite des Übergangsbereichs darstellt.

Der Entwurf von FIR-Filtern mittels Fensterung ist simpel und preiswert in der Berechnung. Deshalb ist dies die schnellste, jedoch nicht unbedingt die beste Methode zum Entwurf von FIR-Filtern.

Entwerfen von optimalen FIR-Filtern mit dem Parks-McClellan Algorithmus

Der Parks-McClellan-Algorithmus bietet eine optimale Entwurfmethode für FIR-Filter, mit der das bestmögliche FIR-Filter für eine gegebene Anzahl von Koeffizienten entworfen werden kann. Ein solcher Entwurf reduziert die negativen Effekte an den Grenzfrequenzen. Er sorgt darüber hinaus für eine bessere Kontrolle der Näherungsfehler in verschiedenen Frequenzbereichen, was mit der Fensterungsmethode nicht möglich ist.

Die Verwendung des Parks-McClellan-Algorithmusses für den Entwurf von FIR-Filtern ist preiswert in der Berechnung. Diese Methode liefert zwar optimale FIR-Filter, setzt dazu jedoch zeitraubende iterative Techniken ein.

Entwerfen von Schmalband FIR-Filtern

Bei der Verwendung von herkömmlichen Methoden zum Entwerfen von FIR-Filtern mit besonders schmalen Bandbreiten können sehr lange Filter entstehen. FIR-Filter von großer Länge erfordern meist einen zeitraubenden Entwurf und eine langwierige Implementierung und sind anfälliger für numerische Ungenauigkeiten. In einigen Fällen können herkömmliche Filterentwurfsmethoden, wie der Parks-McClellan-Algorithmus, überhaupt nicht verwendet werden.

Für den Entwurf von Schmalband FIR-Filtern kann ein sehr effizienter Algorithmus, die Methode des Interpolated Finite Impulse Response (IFIR) Filterentwurfs eingesetzt werden. Mit dieser Entwurfmethode werden Schmalband-Filter erzeugt, die wesentlich weniger Koeffizienten (und deshalb weniger Berechnungen) erfordern als Filter, die durch die direkte Anwendung des Parks-McClellan-Algorithmusses entworfen wurden. LabVIEW verwendet diese Methode außerdem, um Breitband-, Tiefpaß- (Grenzfrequenz nahe Nyquist-Frequenz) und Hochpaßfilter (Grenzfrequenz nahe Null) zu erstellen. Weitere Informationen über den Entwurf von IFIR-Filtern entnehmen Sie bitte *Multirate Systems and Filter Banks* von P.P. Vaidyanathan oder der Abhandlung über interpolierte endliche Impulsantwort-Filter von Neuvo, et al., die beide in Anhang A, *Analyse-Referenzen*, in diesem Handbuch angegeben sind.

Gefensterte FIR-Filter

Zum Auswählen der Art des gewünschten gefensterten FIR-Filters verwenden Sie den Parameter **Filtertyp** des FIR-VIs: Tiefpaß, Hochpaß, Bandpaß oder Bandsperre. Die folgende Liste enthält die beiden verwandten FIR-VIs:

- FIR Gefensterte Koeffizienten—Generiert die gefensterten (oder ungefensterten) Koeffizienten.
- FIR Fenster-Filter—Filtert die Eingabe durch gefensterte (oder ungefensterte) Koeffizienten.

Optimale FIR-Filter

Sie können den Parks-McClellan-Algorithmus verwenden, um optimale linearphasige FIR Filter-Koeffizienten zu entwerfen, so daß das daraus resultierende Filter den Filterspezifikationen für eine gegebene Anzahl von Koeffizienten optimal entspricht. Das Parks-McClellan-VI verwendet als Eingabe ein Array von Bandbeschreibungen, die jeweils Informationen über das gewünschte Verhalten des gegebenen Bands enthalten. Das VI gibt die FIR-Koeffizienten zusammen mit der berechneten Welligkeit aus,

die das Maß der Abweichung des daraus resultierenden Filters von den idealen Filterspezifikationen ist.

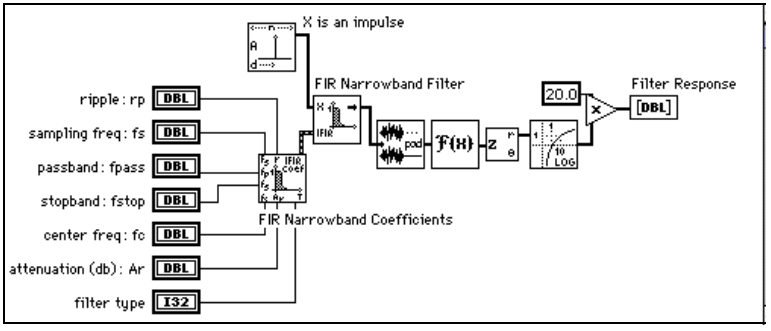
Vier VIs verwenden das Parks-McClellan-VI, um Filter zu implementieren, deren Welligkeitsgrade im Sperr- und Durchlaßbereich übereinstimmen: Equi-Ripple Tiefpaß, Equi-Ripple Hochpaß, Equi-Ripple Bandpaß und Equi-Ripple Bandsperre.

FIR Schmalband-Filter

Sie können FIR Schmalband-Filter mit dem FIR Schmalband-Koeffizienten-VI entwerfen und anschließend die Filterung mit Hilfe des FIR Schmalband-Filter-VIs implementieren. Der Entwurf und die Implementierung müssen separat vorgenommen werden, da viele Schmalband-Filter langwierige Entwurfzeiten erfordern, während der tatsächliche Filtervorgang sehr schnell und effizient ist. Sie sollten dies beim Erstellen Ihrer Schmalband-Filter-Diagramme bedenken.

Die für eine Schmalband-Filter-Spezifikation erforderlichen Parameter sind Filtertyp, Abtastrate, Durchlaß- und Sperrfrequenzen, Welligkeit im Durchlaßbereich (lineare Skala) und Sperrdämpfung (Dezibel). Bei Band- und Bandsperrfiltern beziehen sich die Durchlaß- und Sperrfrequenzen auf Bandbreiten, und Sie müssen einen zusätzlichen Parameter für die Mittenfrequenz angeben. Mit den Schmalband-Filter-VIs können Sie auch Breitband-Tiefpaßfilter (Grenzfrequenz nahe Nyquist-Frequenz) und Breitband-Hochpaßfilter (Grenzfrequenz nahe Null) entwerfen.

In der folgenden Abbildung wird dargestellt, wie das FIR-Schmalband-Koeffizienten-VI und das FIR-Schmalband-Filter-VI verwendet werden, um die Antwort eines Schmalband-Filters auf einen Impuls zu schätzen.



Nicht-lineare Filter

Geglättete Fenster, IIR- und FIR-Filter sind linear, da sie den Prinzipien der Überlagerung und Proportionalität entsprechen:

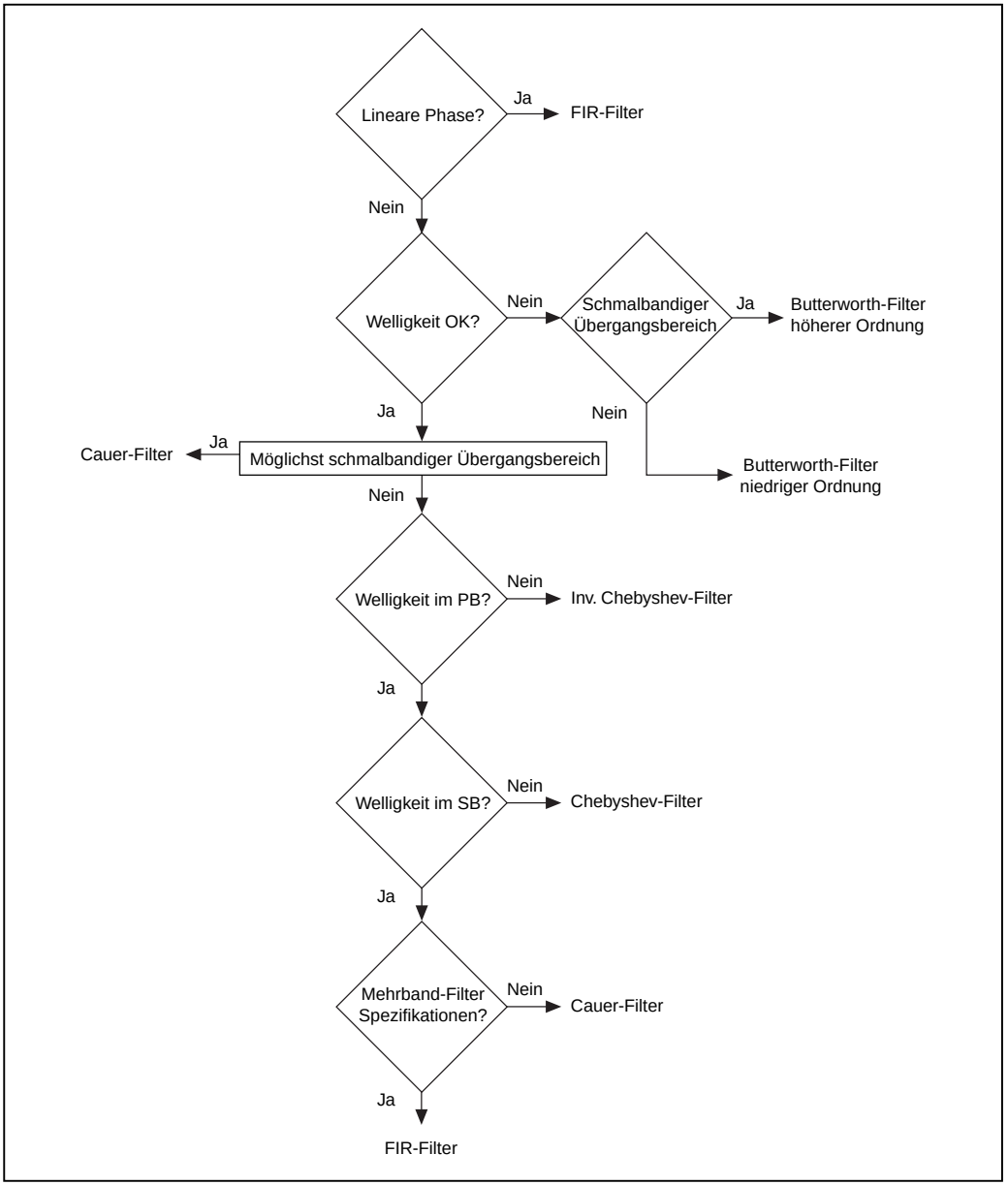
$$L\{ax(t) + by(t)\} = aL\{x(t)\} + bL\{y(t)\},$$

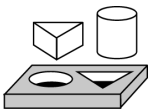
wobei a und b Konstanten, $x(t)$ und $y(t)$ Signale und $L\{\bullet\}$ eine lineare Filterungsoperation darstellt und ihre Ein- und Ausgänge über die Faltungsoperation zueinander in Beziehung stehen.

Ein nicht-linearer Filter entspricht nicht den vorangegangenen Bedingungen, und Sie können seine Ausgangssignale nicht über die Faltungsoperation erhalten, da ein Koeffizientensatz die Impulsantwort des Filters nicht charakterisieren kann. Nicht-lineare Filter verfügen über bestimmte Filterungscharakteristiken, die mit linearen Methoden schwer zu erzielen sind. Das Median-Filter ist ein nicht-linearer Filter, der die Charakteristiken eines Tiefpaß-Filters (zum Entfernen von Hochfrequenzrauschen) mit denen von hohen Frequenzen (zum Erkennen von Kanten) kombiniert.

Wie entscheidet man, welches Filter verwendet werden soll?

Nachdem Sie die verschiedenen Filtertypen und ihre Charakteristiken kennengelernt haben, ergibt sich die Frage, welches Filter am besten für Ihre Anwendung geeignet ist. Einige Faktoren, die die Wahl des geeigneten Filters beeinflussen, sind normalerweise, ob Sie eine lineare Phase benötigen, ob Welligkeit tolerierbar ist und ob ein schmalbandiger Übergangsbereich erforderlich ist. Das folgende Flußdiagramm soll Ihnen als Richtlinie für die Auswahl des richtigen Filters dienen. Bedenken Sie, daß Sie in der Praxis eventuell mit mehreren Möglichkeiten experimentieren müssen, bis Sie die beste Lösung finden.


Lineare Phase?
Ja
FIR-Filter
Nein
Welligkeit OK?
Nein
Schmalbandiger Übergangsbereich
Ja
Butterworth-Filter höherer Ordnung
Ja
Nein
Cauer-Filter
Ja
Möglichst schmalbandiger Übergangsbereich
Butterworth-Filter niedriger Ordnung
Nein
Welligkeit im PB?
Nein
Inv. Chebyshev-Filter
Ja
Welligkeit im SB?
Nein
Chebyshev-Filter
Ja
Mehrband-Filter Spezifikationen?
Nein
Cauer-Filter
Ja
FIR-Filter

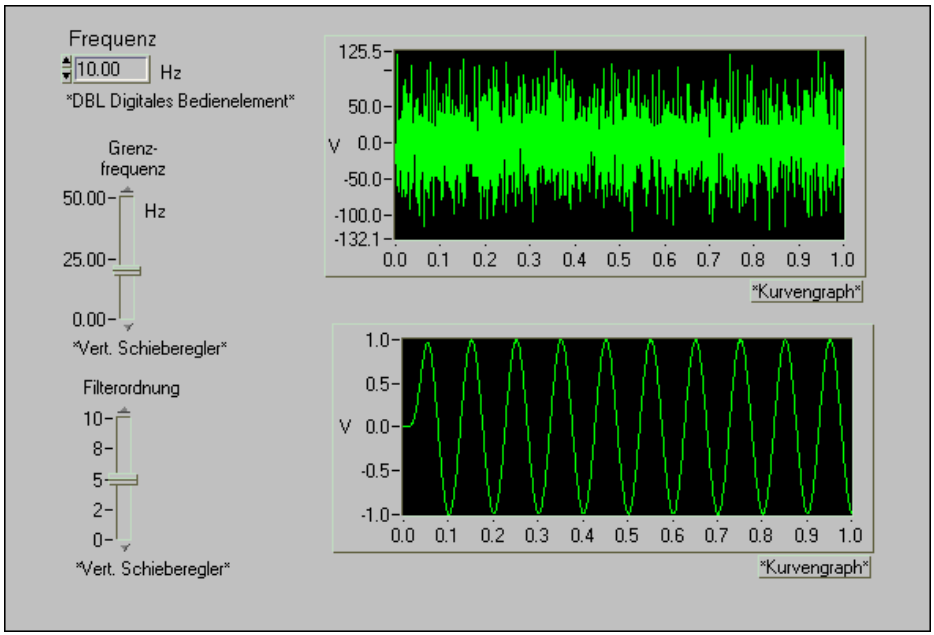


Übung 16-1. Eine Sinuskurve extrahieren

Das Ziel ist es, Datenabtastwerte zu filtern, die sowohl aus Hochfrequenzrauschen als auch einem sinusförmigen Signal bestehen.

Bei dieser Übung kombinieren Sie eine Sinuskurve, die vom Sinusmuster-VI generiert wurde, mit Hochfrequenzrauschen. (Das Hochfrequenzrauschen wird erzielt, indem gleichförmiges weißes Rauschen mit einem Butterworth-Filter hochpaßgefiltert wird.) Das kombinierte Signal wird anschließend von einem anderen Butterworth-Filter tiefpaßgefiltert, um die Sinuskurve zu extrahieren.

Frontpanel



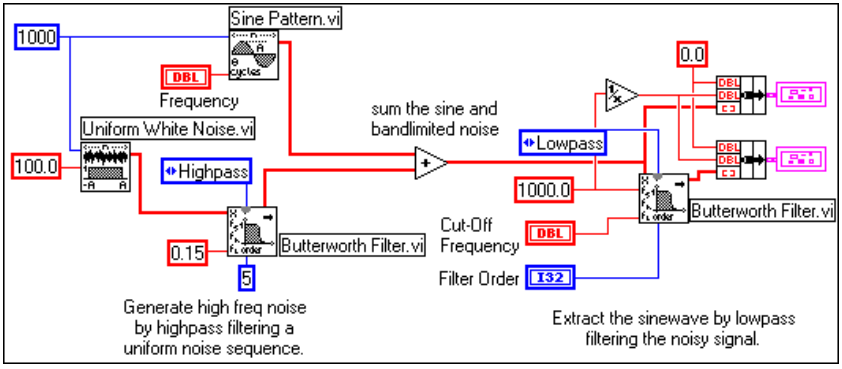
1. Öffnen Sie ein neues VI, und erstellen Sie das Frontpanel wie oben dargestellt.
   a. Wählen Sie eine numerische Eingabe aus der Palette **Elemente» Numerisch,** und beschriften Sie sie mit *Frequenz*.
   b. Wählen Sie Vert. Schieberegler aus der Palette **Elemente» Numerisch,** und beschriften Sie ihn mit *Grenzfrequenz*.

c. Wählen Sie einen weiteren Vert. Schieberegler aus der Palette **Elemente»Numerisch,** und beschriften Sie ihn mit *Filterordnung*.

d. Wählen Sie eine Kurvengraph aus der Palette **Elemente» Numerisch»Graph** für die Darstellung des rauschenden Signals und einen weiteren Kurvengraph für die Darstellung des Originalsignals.

Blockdiagramm

2. Erstellen Sie ein Blockdiagramm wie unten gezeigt.



Sinusmuster-VI (Palette **Funktionen»Analyse»Signalerzeugung**) erzeugt eine Sinuskurve der gewünschten Frequenz.

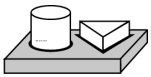
Uniformes weißes Rauschen-VI (Palette **Funktionen»Analyse» Signalerzeugung**) erzeugt uniformes weißes Rauschen, das zum sinusförmigen Signal hinzugefügt wird.

Butterworth Filter-VI (Palette **Funktionen»Analyse»Filter**) tiefpaßfiltert das Rauschen.

Beachten Sie, daß Sie 10 Sinuskurvenzyklen erzeugen und daß es 1000 Abtastwerte gibt. Die Abtastfrequenz zum **Butterworth Filter**-VI auf der rechten Seite ist mit 1000 Hz angegeben. Sie erzeugen also effektiv ein 10 Hz Signal.

3. Speichern Sie das VI als `Extract the Sine Wave.vi` im Verzeichnis `LabVIEW\Activity`.

4. Kehren Sie zurück zum Frontpanel. Wählen Sie eine **Frequenz** von 10 Hz, eine **Grenzfrequenz** von 25 Hz und eine **Filterordnung** von 5. Führen Sie das VI aus.
5. Verringern Sie die **Filterordnung** auf 4, 3 und 2 und beobachten Sie den Unterschied im gefilterten Signal. Erklären Sie, was geschieht, wenn Sie die Filterordnung verringern.
6. Anschließend speichern Sie das VI als `Extract the Sine Wave.vi` in der Bibliothek `Dig.filt.llb`.
7. Schließen Sie das VI.



Ende der Übung 16-1.

Zusammenfassung

Wie Sie den Charakteristiken des Frequenzverhaltens entnehmen konnten, unterscheiden sich praktische Filter von idealen Filtern. Bei praktischen Filtern kann die Verstärkung im Durchlaßbereich nicht immer Eins betragen, die Sperrdämpfung kann nicht immer minus unendlich sein und es gibt einen Übergangsbereich von endlicher Breite. Die Breite des Übergangsbereichs hängt von der Filterordnung ab. Sie nimmt mit erhöhter Ordnung ab.

Sie haben außerdem FIR- und IIR-Digitalfilter kennengelernt. Der Ausgang von FIR-Filtern hängt nur von den aktuellen und vergangenen Eingabewerten ab, während der Ausgang von IIR-Filtern sowohl von den aktuellen und vergangenen Eingabewerten als auch von den vergangenen Ausgabewerten abhängt. Sie haben das Frequenzverhalten von verschiedenen Entwürfen von IIR-Filtern kennengelernt und diese je nach der im Durchlaß- oder Sperrbereich vorhandenen Welligkeit klassifiziert. Aufgrund der Abhängigkeit seines Ausgangs von vergangenen Ausgängen erscheint jedesmal ein Übergangsbereich am Ausgang eines IIR-Filters, wenn das VI aufgerufen wird. Dieser Übergangsbereich kann nach dem ersten Aufruf des VIs beseitigt werden, indem das Bedienelement **init/control** auf den Wert TRUE gesetzt wird.

17

Kurvenanpassung

In diesem Kapitel wird beschrieben, wie Informationen aus einem Datensatz extrahiert werden, um eine Funktionsbeschreibung zu erhalten. Beispiele zur Verwendung von Regressions-VIs entnehmen Sie bitte der Bibliothek `examples\analysis\regressn.llb`.

Einführung in die Kurvenanpassung

Die Kurvenanpassungsanalyse ist eine Methode zur Extraktion eines Satzes von Kurvenparametern oder -koeffizienten aus dem Datensatz, um eine Funktionsbeschreibung des Datensatzes zu erhalten. Der Algorithmus, der eine Kurve an einen bestimmten Datensatz anpaßt, wird Methode der kleinsten Quadrate genannt und ist in den meisten Einführungsbüchern in die Wahrscheinlichkeit und Statistik beschrieben. Der Fehler ist wie folgt definiert:

$$e(a) = [f(x,a) - y(x)]^2, \qquad (17\text{-}1)$$

wobei $e(a)$ der Fehler, $y(x)$ der beobachtete Datensatz, $f(x,a)$ die Funktionsbeschreibung des Datensatzes und a der Satz der Kurvenkoeffizienten ist, die die Kurve am besten beschreiben.

Beispielsweise ist $a = \{a_0, a_1\}$. Dann lautet die Funktionsbeschreibung einer Linie

$$f(x,a) = a_0 + a_1 x.$$

Der Algorithmus der kleinsten Quadrate ermittelt a, indem das folgende System gelöst wird:

$$\frac{\partial}{\partial a} e(a) = 0 \qquad (17\text{-}2)$$

Zum Lösen dieses Systems richten Sie das durch die Erweiterung der Gleichung 17-2 erzeugte Jacobsche System ein und lösen es. Nachdem das System für a gelöst ist, können Sie eine Schätzung des beobachteten Datensatzes für irgendeinen Wert von x mit Hilfe der Funktionsbeschreibung $f(x, a)$ erzielen.

In LabVIEW wird das Jacobsche System automatisch von den Kurvenanpassung-VIs eingerichtet und gelöst. Sie geben zudem den Koeffizientensatz zurück, der Ihren Datensatz am besten beschreibt. Sie können sich auf die Funktionsbeschreibung Ihrer Daten konzentrieren und brauchen sich nicht um die Lösung des Systems in Gleichung 17-2 zu kümmern.

Zwei Eingabesequenzen, die Y- und die X-Werte, stellen den Datensatz $y(x)$ dar. Ein Abtastwert oder Punkt im Datensatz ist

$$(x_i, y_i),$$

wobei x_i das i^{te} Element der X-Werte-Sequenz und y_i das i^{te} Element der Y-Werte-Sequenz ist.

Im allgemeinen gibt es für jeden vordefinierten Typ der Kurvenanpassung zwei Arten von VIs, es sei denn, es wäre anders angegeben. Ein Typ gibt nur die Koeffizienten zurück, so daß die Daten weiter verändert werden können. Der andere Typ gibt die Koeffizienten, die entsprechende erwartete oder angepaßte Kurve und den mittleren quadratischen Fehler (Mean Square Error = MSE) zurück. Da es sich um ein diskretes System handelt, berechnet das VI den MSE, der ein relativer Meßwert der Differenz zwischen den erwarteten und den tatsächlich beobachteten Kurvenwerten ist, wobei folgende Formel verwendet wird:

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - y_i)^2, \qquad (17\text{-}3)$$

wobei f die Sequenz der angepaßten Werte, y die Sequenz der beobachteten Werte und n die Anzahl der beobachteten Abtastpunkte darstellt.

Die Analysis-Bibliothek enthält sowohl lineare als auch nicht-lineare Kurvenanpassungs-Algorithmen. Die verschiedenen Arten der Kurvenanpassung in LabVIEW werden im folgenden beschrieben:

- *Linearanpassung*—paßt experimentelle Daten an eine gerade Linie der folgenden Form an:
  $y = mx + c$.

$$y[i]=a_0+a_1*x[i]$$

- *Exponentialanpassung*—paßt Daten an eine Exponentialkurve der folgenden Form an:

$$y = aexp(bx)$$
$$y[i] = a_0*\exp(a_1*x[i])$$

- *Allgemeine Polynomanpassung*—paßt Daten an eine polynome Funktion der folgenden Form an:

$$y = a + bx + cx\text{^}2 + ...$$
$$y[i] = a_0+a_1*x[i]+a_2*x[i]^2...$$

- *Allgemeine Linearanpassung*—paßt Daten an die folgende Gleichung an:

$$y[i] = a_0+a_1*f_1(x[i])+a_2*f_2(x[i]) + ...,$$

wobei y[i] eine lineare Kombination der Parameter a_0, a_1, a_2... ist. Die allgemeine Linearanpassung bietet zudem wählbare Algorithmen, so daß eine größere Genauigkeit erzielt wird. Beispielsweise ist $y = a_0 + a_1*\sin(x)$ eine Linearanpassung, da *y* eine lineare Beziehung zu den Parametern a_0 und a_1 hat. Polynomanpassungen sind aus diesem Grund stets Linearanpassungen. Aber es ist möglich, besondere Algorithmen für die Polynomanpassung zu entwerfen, mit denen der Anpassungsvorgang beschleunigt und die Genauigkeit verbessert werden kann.

- *Nicht-lineare Levenberg-Marquardt-Anpassung*—paßt Daten an die folgende Gleichung an:

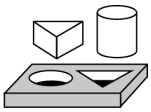
$$y[i] = f(x[i], a_0, a_1, a_2...),$$

wobei a_0, a_1, a_2... die Parameter darstellen. Diese Methode ist die allgemeinste Methode und setzt nicht voraus, daß *y* eine lineare Beziehung zu $_0$, a_1, a_2... hat. Sie kann zur Anpassung von linearen oder nicht-linearen Kurven verwendet werden, wird aber fast ausschließlich zur Anpassung von nicht-linearen Kurven verwendet, da für die lineare Kurvenanpassung die allgemeine Linearanpassung besser geeignet ist. Die Levenberg-Marquardt-Methode gewährleistet nicht immer ein korrektes Ergebnis, das deshalb stets überprüft werden muß.

Anwendungen der Kurvenanpassung

Es gibt zahlreiche praktische Anwendungen der Kurvenanpassung. Einige werden im folgenden aufgezählt:

- Entfernen von Meßrauschen.
- Auffüllen von fehlenden Datenpunkten (wenn z. B. eine oder mehrere Messungen versäumt oder falsch aufgezeichnet wurden).
- Interpolation (Schätzung von Daten zwischen Datenpunkten; wenn z. B. die Zeitspanne zwischen den Messungen nicht klein genug ist).
- Extrapolation (Schätzung von Daten außerhalb von Datenpunkten; wenn Sie z.B. Datenwerte benötigen, bevor oder nachdem die Messungen vorgenommen wurden.)
- Differenzierung von digitalen Daten. (Wenn Sie z. B. die Ableitung der Datenpunkte finden müssen. Die diskreten Daten können von einem Polynom modelliert werden. Die daraus resultierende polynome Gleichung kann anschließend differenziert werden.)
- Integration von digitalen Daten (Wenn Sie z. B. den Bereich unterhalb der Kurve ermitteln wollen und nur die diskreten Punkte der Kurve verfügbar sind.)
- Um den Verlauf eines Objektes auf der Basis von diskreten Messungen seiner Geschwindigkeit (erste Ableitung) oder Beschleunigung (zweite Ableitung) zu erhalten.

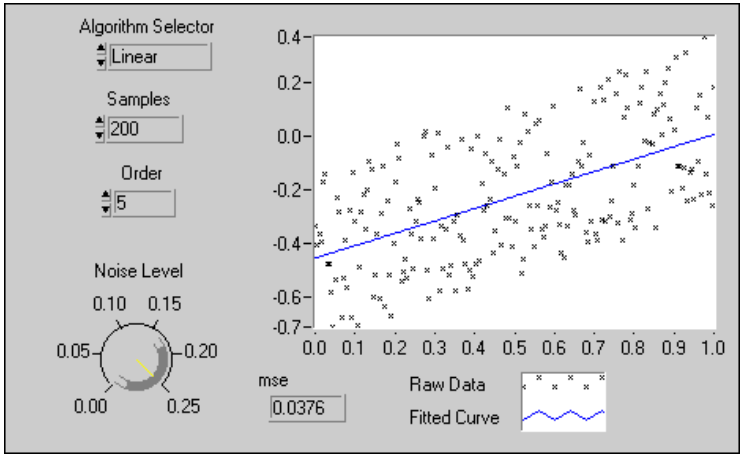


Übung 17-1. Kurvenanpassung-VIs verwenden

Das Übungsziel ist, die Linear-, Exponential- und Polynomkurvenanpassung-VIs zu verwenden und zu vergleichen, um den Satz der kleinsten quadratischen Koeffizienten zu erhalten, der am besten einen Satz von Datenpunkten darstellt.

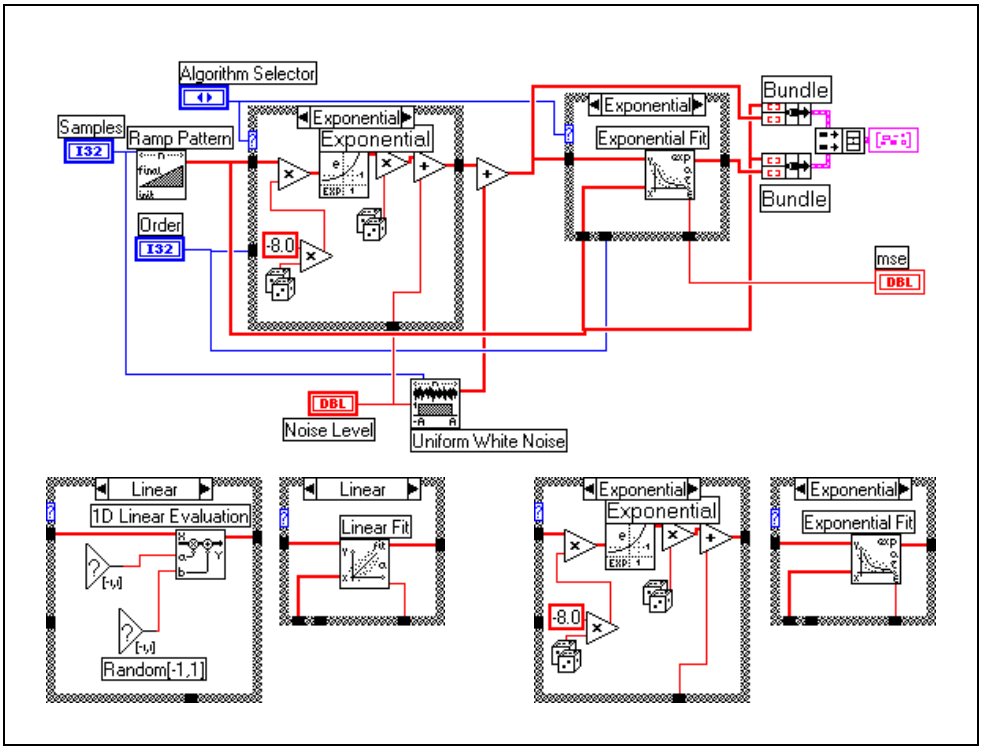
Frontpanel

1. Öffnen Sie das Regressions-Demo-VI aus der Bibliothek `regressn.llb`. Das Frontpanel und das Blockdiagramm wurden bereits für Sie erstellt.



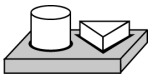
Dieses VI erzeugt ”rauschende” Datenabtastwerte, die annähernd linear, exponentiell oder polynom sind. Es verwendet anschließend die entsprechenden Analyse-Kurvenanpassung-VIs zur Bestimmung der Parameter der Kurve, die am besten zu diesen Datenpunkten passen. (An dieser Stelle brauchen Sie sich nicht darum zu kümmern, wie die rauschenden Datenabtastwerte erzeugt werden.) Sie können die Rauschamplitude mit dem Bedienelement **Rausch-Level** auf dem Frontpanel steuern.

Blockdiagramm



2. Wählen Sie *Linear* im Bedienelement **Algorithmus-Auswahl**, und stellen Sie das Bedienelement **Rausch-Level** auf ca. 0,1. Führen Sie das VI aus. Beachten Sie die Ausdehnung der Datenpunkte und die angepaßte Kurve (gerade Linie).
3. Experimentieren Sie mit verschiedenen Werten für **Ordnung** und **Rausch-Level**. Was fällt Ihnen auf? Wie verändert sich der MSE?
4. Ändern Sie die **Algorithmus-Auswahl** auf *Exponentiell*, und führen Sie das VI aus. Experimentieren Sie mit verschiedenen Werten für **Ordnung** und **Rausch-Level**. Was fällt Ihnen auf?
5. Ändern Sie die **Algorithmus-Auswahl** auf *Polynom*, und führen Sie das VI aus. Experimentieren Sie mit verschiedenen Werten für **Ordnung** und **Rausch-Level**. Was fällt Ihnen auf?
6. Insbesondere, wenn das Bedienelement **Algorithmus-Auswahl** auf *Polynom* eingestellt ist, ändern Sie die **Ordnung** auf 0, und führen Sie anschließend das VI aus. Stellen Sie anschließend 1 dafür ein, und führen Sie das VI erneut aus. Erklären Sie Ihre Beobachtungen.

7. Je nachdem, wie Ihre Beobachtungen aus den Schritten 2, 3, 4 und 5 ausfielen, für welchen der Algorithmen (Linear, Exponentiell, Polynom) ist das Bedienelement **Ordnung** am wirkungsvollsten? Warum?
8. Schließen Sie das VI, und beenden Sie das Programm. Bitte speichern Sie Ihre Änderungen nicht.



Ende der Übung 17-1.

Theorie der allgemeinen LS Linearanpassung

Das Problem der allgemeinen LS Linearanpassung kann folgendermaßen beschrieben werden.

Finden Sie bei einem Beobachtungsdatensatz einen Koeffizientensatz, der dem linearen "Modell" entspricht.

$$y_i = b_o x_{i0} + \ldots + b_{k-1} x_{ik-1}$$

$$= \sum_{j=0}^{k-1} b_j x_{ij} \qquad \text{i=0, 1, ..., n–1,} \tag{17-4}$$

wobei *B* ein Satz von **Koeffizienten**, *n* eine Anzahl von Elementen in **Y-Werten** und die Anzahl der Zeilen von **H** und *k* die Anzahl der **Koeffizienten** ist**.**

x_{ij} stellt Ihre Beobachtungsdaten dar, die in **H** enthalten sind.

$$H = \begin{bmatrix} x_{00} & x_{01} \cdots & x_{0k-1} \\ x_{10} & x_{11} \cdots & x_{1k-1} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ x_{n-10} & x_{n-12} \cdots & x_{n-1k-1} \end{bmatrix}$$

Die Gleichung 17-4 kann auch als $Y = HB$ wiedergegeben werden.

Dieses Modell stellt ein mehrfaches lineares Regressionsmodell dar, das mehrere Variablen $x_{i0}, x_{i1}, \ldots, x_{ik-1}$ verwendet, um eine Variable y_i vorauszusagen. Im Gegensatz dazu basieren die Linear-, Exponential- und Polynomanpassung-VIs auf einer einzigen Prädiktionsvariablen, die eine Variable zur Voraussage einer anderen verwenden.

In den meisten Fällen stehen mehr Beobachtungsdaten als Koeffizienten zur Verfügung. Die Gleichungen in 17-4 liefern eventuell kein Ergebnis. Das Anpassungsproblem lautet in diesem Fall, den Koeffizienten *B* zu finden, der die Differenz zwischen den beobachteten Daten y_i und dem vorausgesagten Wert auf ein Minimum reduziert:

$$z_i = \sum_{j=0}^{k-1} b_j x_{ij} .$$

Dieses VI verwendet die Methode der kleinsten Chi-Quadratfläche, um die Koeffizienten in 17-4 zu erhalten, d.h.die Lösung *B*, die folgende Menge auf ein Minimum reduziert:

$$\chi^2 = \sum_{i=0}^{n-1} \left(\frac{y_i - z_i}{\sigma_i} \right)^2 = \sum_{i=0}^{n-1} \left(\frac{y_i - \sum_{j=0}^{k-1} b_j x_{ij}}{\sigma_i} \right)^2 = |H_0 B - Y_0|^2, \qquad (17\text{-}5)$$

wobei

$$h_{oij} = \frac{x_{ij}}{\sigma_i}, \; y_{oi} = \frac{y_i}{\sigma_i}, \; i=0, 1, \ldots, n-1; \; j=0, 1, \ldots, k-1.$$

In dieser Gleichung ist σ_i die **Standardabweichung**. Wenn die Messungsfehler unabhängig und mit einer konstanten Standardabweichung $\sigma_i = \sigma$ normalverteilt sind, dann ist die Gleichung oben auch die Schätzung der kleinsten Quadrate.

Es gibt verschiedene Arten, um χ^2 auf ein Minimum zu reduzieren. Beispielsweise können die partiellen Ableitungen von χ^2 mit Bezug auf b_0, b_1, ... , b_{k-1} auf Null gesetzt werden.

$$\begin{cases} \frac{\partial \chi^2}{\partial b_0} = 0 \\ \frac{\partial \chi^2}{\partial b_1} = 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial \chi^2}{\partial b_{k-1}} = 0 \end{cases}$$

Aus den Gleichungen oben kann folgendes abgeleitet werden:

$$H_0^T H_0 B = H_0^T Y \,, \tag{17-6}$$

wobei H_0^T die Transponierte von $\mathbf{H}_0$ ist.

Die Gleichungen in 17-6 werden auch als normale Gleichungen der kleinsten quadratischen Probleme bezeichnet. Sie können mit den LU- oder Cholesky-Faktorisierungs-Algorithmen gelöst werden, jedoch ist die Lösung der normalen Gleichungen anfällig für Abrundungsfehler.

Eine alternative und empfehlenswertere Methode zur Minimierung von χ^2 ist, die kleinste quadratische Lösung der folgenden Gleichungen zu finden:

$$H_0 B = Y_0 .$$

Um die Lösung *B* zu finden, können Sie die QR- oder SVD-Faktorisierung verwenden. Bei der QR-Faktorisierung können Sie Householder, Givens und Givens2 (auch schnelles Givens) wählen.

Verschiedene Algorithmen bieten eine unterschiedliche Genauigkeit, und in manchen Fällen kann die Gleichung mit einem anderen Algorithmus gelöst werden, wenn dies mit dem einen nicht gelingt. Sie können verschiedene Algorithmen ausprobieren, um den zu finden, der am besten für Ihre Beobachtungsdaten geeignet ist.

Die Kovarianz-Matrix C wird folgendermaßen berechnet:

$$C = (H_0^T H_0)^{-1} .$$

Die beste Anpassung Z wird durch folgende Gleichung gegeben:

$$z_i = \sum_{j=0}^{k-1} b_j x_{ij}$$

Den MSE erhält man aus der Formel

$$\boldsymbol{mse} = \frac{1}{n} \sum_{i=0}^{n-1} \left(\frac{y_i - z_i}{\sigma_i} \right)^2 .$$

Die Polynomanpassung, die eine einzige Prädiktionsvariable besitzt, kann als Sonderfall der mehrfachen Regression betrachtet werden. Wenn die Beobachtungsdatensätze $\{x_i, y_i\}$ lauten, wobei i = 0, 1, …, n–1, dann sieht das Modell der Polynomanpassung folgendermaßen aus:

$$y_i = \sum_{j=0}^{k-1} b_j x_i^j = b_0 + b_1 x_i + b_2 x_i^2 + \ldots + b_{k-1} x_i^{k-1} \qquad (17\text{-}7)$$

$$i = 0, 1, 2, \ldots, n-1.$$

Ein Vergleich der Gleichungen 17-4 und 17-7 zeigt, daß $x_{ij} = x_i^j$.
Folglich gilt

$$x_{i0} = x_i^0 , \quad x_{i1} = x_i , \quad x_{i2} = x_i^2 , \ldots \quad x_{ik-1} = x_i^{k-1} = 1.$$

In diesem Fall können Sie **H** folgendermaßen erstellen:

$$H = \begin{Bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{k-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{k-1} \\ . & & & & \\ . & & & & \\ . & & & & \\ . & & & & \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{k-1} \end{Bmatrix}$$

Anstatt $x_{ij} = x_j^i$ zu verwenden, können Sie auch eine andere Funktionsformel wählen, um die Datensätze $\{x_i,y_i\}$ anzupassen. Im allgemeinen können Sie $x_{ij} = f_j(x_i)$ wählen. Hierbei ist $f_j(x_i)$ das Funktionsmodell, das Sie zum Anpassen Ihrer Beobachtungsdaten wählen. Bei der Polynomanpassung gilt $f_j(x_i) = x_i^j$.

Im allgemeinen können Sie **H** folgendermaßen erstellen:

$$H = \begin{Bmatrix} f_0(x_0) & f_1(x_0) & f_2(x_0) & \dots & f_{k-1}(x_0) \\ f_0(x_1) & f_1(x_1) & f_2(x_1) & \dots & f_{k-1}(x_1) \\ . & & & & \\ . & & & & \\ . & & & & \\ . & & & & \\ f_0(x_{n-1}) & f_1(x_{n-1}) & f_2(x_{n-1}) & \dots & f_{k-1}(x_{n-1}) \end{Bmatrix}$$

Ihr Anpassungsmodell lautet:

$$y_i = b_0 f_0(x) + b_1 f_1(x) + \dots + b_{k-1} f_{k-1}(x) \ .$$

Verwenden des allgemeinen LS Linearanpassung-VIs

Das Linearanpassung-VI berechnet die Koeffizienten a_0 und a_1, welche die Versuchsdaten (x[i] und y[i]) am besten an ein geradliniges Modell anpassen, das folgendermaßen definiert ist:

$$y[i] = a_0 + a_1*x[i]$$

Hierbei ist y[i] eine Linearkombination der Koeffizienten a_0 und a_1. Dieses Konzept kann noch erweitert werden, so daß der Multiplikator für a_1 irgendeine Funktion *x* ist. Beispiel:

$$y[i] = a_0 + a_1*\sin(\omega x[i])$$

oder

$$y[i] = a_0 + a_1*x[i]^2$$

oder

$$y[i] = a_0 + a_1*\cos(\omega x[i]^2),$$

wobei ω die Kreisfrequenz darstellt. In jedem dieser Fälle ist y[i] eine Linearkombination der Koeffizienten a_0 und a_1. Dies ist der grundlegende Gedanke, auf dem das allgemeine LS Linearanpassung-VI basiert, wobei y[i] Linearkombinationen von mehreren Koeffizienten darstellen kann, die jeweils mit irgendeiner Funktion x[i] multipliziert werden können. Deshalb können Sie es zur Berechnung von Koeffizienten der Funktionsmodelle verwenden, die als Linearkombinationen der Koeffizienten dargestellt werden können, wie z.B.

$$y = a_0 + a_1*\sin(\omega x)$$

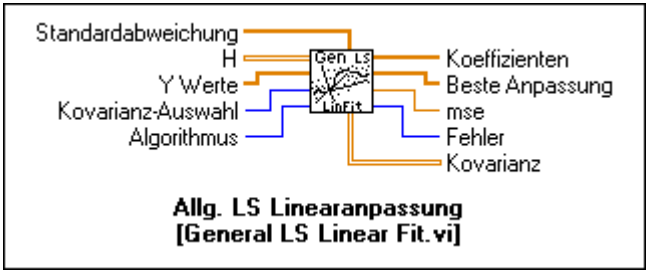
oder

$$y = a_0 + a_1*x^2 + a2*\cos(\omega x^2)$$
$$y = a_0 + a_1*(3\sin(\omega x)) + a_2*x^3 + a_3 / x + ...$$

Beachten Sie, daß *y* in jedem Fall eine *Linearfunktion* der Koeffizienten ist (obwohl es eine nicht-lineare Funktion von *x* sein kann).

Im folgenden wird erklärt, wie das allgemeine LS Linearanpassung-VI verwendet wird, um die beste Linearanpassung an einen Satz von

Datenpunkten zu finden. Die Ein- und Ausgaben des allgemeinen LS Linearanpassung-VI sind in der folgenden Abbildung dargestellt:



Die von Ihnen gesammelten Daten (x[i] und y[i]) werden an die Eingaben **H** und **Y Werte** gegeben. Die Ausgabe **Kovarianz** bezeichnet die Matrix von Kovarianzen zwischen den Koeffizienten a_k, wobei c_{ij} die Kovarianz zwischen a_i und a_j und c_{kk} die Varianz von a_k ist. An dieser Stelle brauchen Sie sich nicht über die Eingaben **Standardabweichung**, **Kovarianz-Auswahl** und **Algorithmus** zu kümmern. Sie verwenden hier nur die Standardwerte. Weitere Informationen über diese Eingaben entnehmen Sie bitte der *Analyse-Online-Referenz.*

Die Matrix **H** wird mit *Beobachtungsmatrix* bezeichnet und später noch genauer beschrieben. **Y Werte** ist der Satz der beobachteten Datenpunkte y[i]. Nehmen Sie z.B. an, daß Sie Abtastwerte (**Y Werte**) mit einem Wandler erfaßt haben und die Lösung für die Koeffizienten des folgenden Modells finden wollen:

$$y = a_o + a_1 \sin(\omega x) + a_2 \cos(\omega x) + a_3 x^2$$

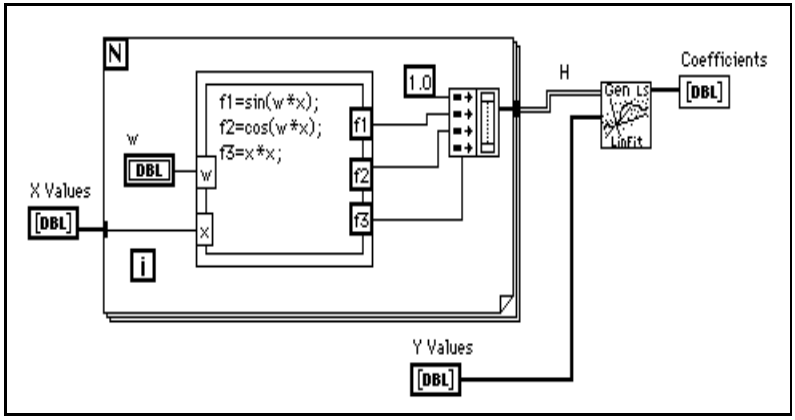
Der Multiplikator für jedes a_j (0 ð *j* ð3) ist jeweils eine andere Funktion. Beispielsweise wird a_0 mit 1, a_1 mit sin(ω*x*), a_2 mit cos(ω*x*), usw. multipliziert. Zum Erstellen von **H** setzen Sie jede Spalte von **H** auf die unabhängigen Funktionen, die bei jedem *x*-Wert x[i] berechnet wurden.

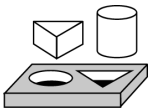
Wenn es beispielsweise 100 *x*-Werte gäbe, würde **H** folgendermaßen lauten:

$$H = \begin{bmatrix} 1 & \sin(\omega x_0) & \cos(\omega x_0) & x_0^2 \\ 1 & \sin(\omega x_1) & \cos(\omega x_1) & x_1^2 \\ 1 & \sin(\omega x_2) & \cos(\omega x_2) & x_2^2 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \sin(\omega x_{99}) & \cos(\omega x_{99}) & x_{99}^2 \end{bmatrix}$$

Bei *N* Datenpunkten und *k* Koeffizienten (a_0, a_1, ...a_{k-1}), die zu lösen sind, ist H eine *N-mal-k*-Matrix mit *N* Reihen und *k* Spalten. So ist die Anzahl der Reihen von **H** gleich der Anzahl der Elemente in **Y Werte**, während die Anzahl der Spalten von **H** der Anzahl der Koeffizienten entspricht, die Sie zu lösen versuchen.

In der Praxis steht **H** nicht zur Verfügung und muß daher erstellt werden. Da Ihnen die von *N* unabhängigen **X Werte** und die beobachteten **Y Werte** zur Verfügung stehen, demonstriert das folgende Blockdiagramm, wie **H** erstellt und das allgemeine LS Linearanpassung-VI verwendet wird.





Übung 17-2. Allgemeines LS Linearanpassung-VI verwenden

Bei dieser Übung sollen Sie lernen, wie die Eingabeparameter eingerichtet werden und wie das allgemeine LS Linearanpassung-VI verwendet wird.

Diese Übung zeigt, wie Sie das allgemeine LS Linearanpassung-VI verwenden, um den Satz der kleinsten quadratischen Koeffizienten a und die angepaßten Werte zu erhalten und wie die Eingabeparameter des VIs eingerichtet werden.

Das Ziel dabei ist, den Satz der kleinsten quadratischen Koeffizienten **a** zu finden, der am besten den Datenpunktsatz (x[i], y[i]) darstellt. Als Beispiel

nehmen Sie an, daß es einen physischen Vorgang gibt, der Daten unter Verwendung der folgenden Beziehung erzeugt:

$$= 2h_0(x) + 3h_1(x) + 4h_2(x) + Rauschen\,, \qquad (17\text{-}8)$$

wobei

$$h_0(x) = \sin(x^2),$$
$$h_1(x) = \cos(x),$$
$$h_2(x) = \frac{1}{x+1},$$

und *Rauschen* ein Zufallswert ist. Nehmen Sie außerdem an, daß Sie die Beziehung zwischen *x* und *y* ungefähr kennen, jedoch nicht ganz sicher sind, wie die Koeffizientenwerte lauten. Sie nehmen also an, daß die Beziehung zwischen *x* und *y* folgendermaßen aussieht:

$$y = a_0f_0(x) + a_1f_1(x) + a_2f_2(x) + a_3f_3(x) + a_4f_4(x)\,, \qquad (17\text{-}9)$$

wobei

$$f_o(x) = 1{,}0$$

$$f_1(x) = \sin(x^2),$$

$$f_2(x) = 3\cos(x),$$

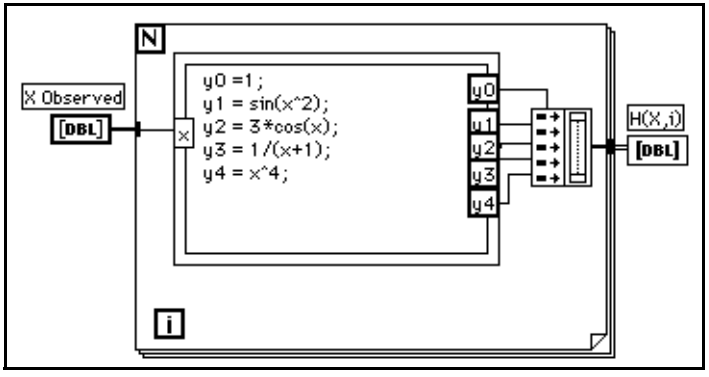
$$f_3(x) = \frac{1}{x+1},$$

$$f_4(x) = x^4 .$$

Die Gleichungen 17-8 und 17-9 entsprechen jeweils dem tatsächlichen physischen Vorgang sowie ihren Erwartungen für diesen Vorgang. Die Koeffizienten, die Sie für Ihre Annahme wählen, können den tatsächlichen Werten nahe kommen oder aber von diesen abweichen. Ihr Ziel ist es nun, die Koeffizienten *a* genau zu bestimmen.

Erstellen der Beobachtungsmatrix

Um die Koeffizienten *a* zu erhalten, müssen Sie den Satz der Punkte (x[i], y[i]) in den Arrays **H** und **Y Werte** (wobei die Matrix H ein 2D-Array ist) des allgemeinen LS Linearanpassung-VI liefern. Die Punkte x[i] und y[i] sind die in Ihrem Versuch beobachteten Werte. Mit dem Formel-Element, das im folgenden Blockdiagramm dargestellt ist, läßt sich die Matrix **H** auf einfache Weise erstellen.

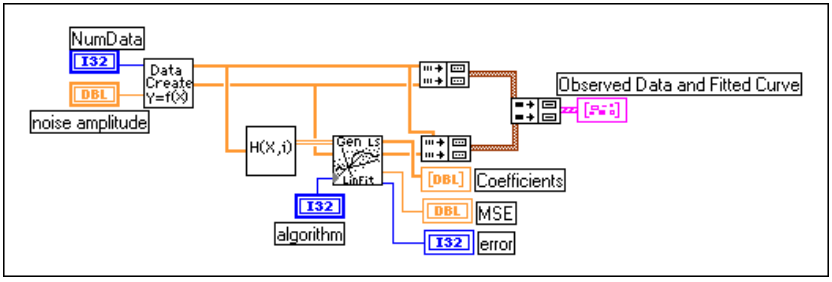


Sie können das Formel-Element bearbeiten und Funktionen ändern, hinzufügen oder löschen. Jetzt haben Sie alle erforderlichen Eingaben, um mit Hilfe des allgemeinen LS Linearanpassung-VI eine Lösung für *a* zu finden. Um Gleichung (1) aus Gleichung (2) zu erhalten, müssen Sie $f_0(x)$ mit 0,0, $f_1(x)$ mit 2,0, $f_2(x)$ mit 1,0, $f_3(x)$ mit 4,0 und $f_4(x)$ mit 0,0

multiplizieren. Beim Betrachten von Gleichung (1) und Gleichung (2) beachten Sie, daß der erwartete Koeffizientensatz folgendermaßen lautet:

$$a = \{0{,}0,\ 2{,}0,\ 1{,}0,\ 4{,}0,\ 0{,}0\}.$$

Das folgende Blockdiagramm verdeutlicht, wie das allgemeine LS Linearanpassung-VI eingerichtet wird, um die Koeffizienten und einen neuen *y*-Wertesatz zu erhalten.



Das mit Daten erzeugen bezeichnete SubVI erzeugt die **X** und **Y** Arrays. Sie können dieses Icon durch eines ersetzen, das die Daten in Ihren Versuchen tatsächlich erfaßt. Das mit **H(X,i)** bezeichnete Icon erzeugt die 2D-Matrix **H.**

Der letzte Teil des Blockdiagramms überlagert die ursprünglichen und die geschätzten Datenpunkte und erstellt eine visuelle Aufzeichnung der allgemeinen LS Linearanpassung. Die Ausführung des allgemeinen LS Linearanpassung-VIs mit den Werten **X**, **Y** und **H** ergibt den folgenden Koeffizientensatz:

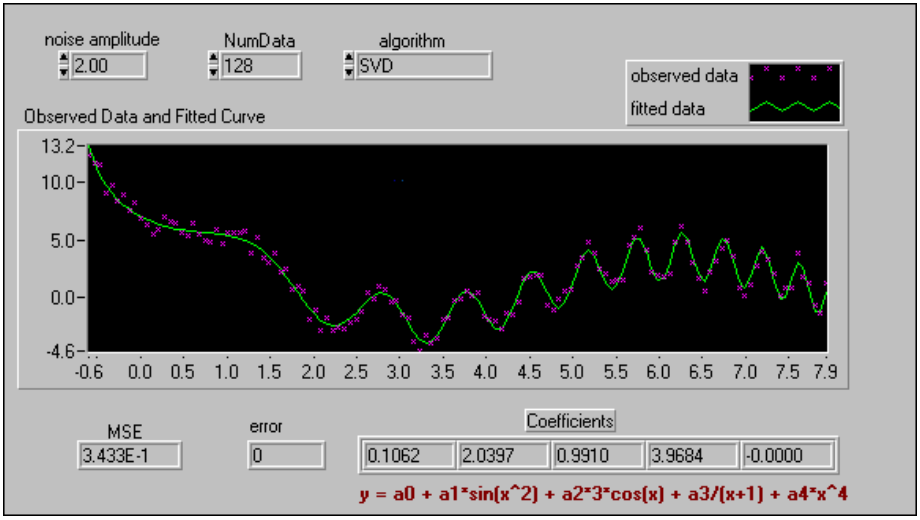
Coefficients

-0.0298	2.1670	1.0301	3.9226	0.0000

Daraus ergibt sich die Gleichung

$$y = 0{,}0298(1) + 2{,}1670\sin(x^2) + 1{,}0301(3\cos(x)) + 3{,}9226/(x+1) + 0{,}00(x^4)$$
$$= 0{,}0298 + 2{,}1670\sin(x^2) + 1{,}0301(3\cos(x)) + 3{,}9226/(x+1)$$

Der folgende Graph stellt die Ergebnisse dar.



Sie werden jetzt das VI kennenlernen, in das dieses bestimmte Beispiel implementiert wurde.

1. Öffnen Sie das allgemeine LS Anpassung-Beispiel-VI aus der Bibliothek `examples\analysis\regressn.llb`.
2. Untersuchen Sie das Blockdiagramm. Stellen Sie sicher, daß Sie es verstehen.
3. Untersuchen Sie das Frontpanel.

   noise amplitude SubVI kann die Amplitude des Rauschens ändern, das den Datenpunkten hinzugefügt wurde. Je größer dieser Wert ist, desto mehr breiten sich die Datenpunkte aus.

   NumData: die Anzahl der Datenpunkte, die Sie erzeugen wollen.

   Algorithmus: bietet die Wahl von sechs verschiedenen Algorithmen zur Berechnung des Koeffizientensatzes und der angepaßten Werte. Bei diesem Beispiel gibt es keinen bedeutenden Unterschied zwischen den verschiedenen Algorithmen. Sie können verschiedene Algorithmen vom Frontpanel wählen, um die Ergebnisse anzusehen.

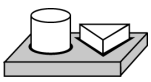
In einigen Fällen können verschiedene Algorithmen, je nach Ihrem Beobachtungsdatensatz, bedeutende Unterschiede aufweisen.

MSE: liefert den mittleren quadratischen Fehler. Je kleiner dieser ist, desto besser die Anpassung.

error: teilt ggf. den Fehlercode mit. Wenn der Fehlercode = 0, bedeutet das, daß kein Fehler aufgetreten ist. Eine Liste der Fehlercodes entnehmen Sie bitte Anhang A, *Fehlercodes*, im *LabVIEW Funktionen- und VI-Referenzhandbuch*.

Coefficients: die berechneten Werte der Koeffizienten (a_0, a_1, a_2, a_3 und a_4) des Modells.

4. Führen Sie das VI mit stetig größer werdenden Werten für **noise amplitude** aus. Was geschieht mit den beobachteten Daten, die in den Graphen eingezeichnet sind? Was geschieht mit dem mittleren quadratischen Fehler?
5. Führen Sie das VI für einen feststehenden Wert für **noise amplitude** aus, indem Sie verschiedene Algorithmen aus dem Bedienelement **Algorithmus** wählen. Ist ein Algorithmus besser geeignet als ein anderer? Aus welchem erhalten Sie den kleinsten quadratischen Fehler?
6. Wenn Sie die Übung beendet haben, schließen Sie das VI. Bitte speichern Sie Ihre Änderungen nicht.



Ende der Übung 17-2.

Theorie der nicht-linearen Levenberg-Marquardt-Anpassung

Dieses VI bestimmt den Koeffizientensatz, der die Chi-Quadrat-Menge auf ein Minimum reduziert:

$$\chi^2 = \sum_{i=0}^{N-1} \left(\frac{y_i - f(x_i; a_1 \ldots a_M)}{\sigma_i} \right)^2 \qquad (17\text{-}10)$$

In dieser Gleichung stellen (x_i, y_i) die Eingabedatenpunkte dar und $f(x_i;a_1...a_M) = f(X, A)$ ist die nicht-lineare Funktion, wobei $a_1...a_M$ für die Koeffizienten steht. Wenn die Messungsfehler unabhängig und mit einer

konstanten Standardabweichung von $\sigma_i = \sigma$ normalverteilt sind, dann entspricht diese Gleichung der kleinsten quadratischen Schätzung.

Sie müssen die nicht-lineare Funktion $f = f(X, A)$ im Formel-Element im Blockdiagramm des Ziel-Fkt. & nicht-lineare Abltg.-VIs angeben. Dieses VI ist ein SubVI des nicht-linearen Levenberg-Marquardt-Anpassung-VIs. Sie können auf das Ziel-Fkt. & nicht-lineare Abltg.-VI zugreifen, indem Sie es aus dem Menü auswählen, das angezeigt wird, wenn Sie **Projekt»SubVIs des VIs** wählen.

Dieses VI liefert zwei Methoden, um die Jacobsche Funktion (partielle Ableitungen in Bezug auf die Koeffizienten) zu berechnen, die für den Algorithmus erforderlich ist. Diese Methoden lauten:

- Numerische Berechnung—Verwendet eine numerische Approximation zur Berechnung der Jacobschen Funktion.
- Formelberechnung—Verwendet eine Formel zur Berechnung der Jacobschen Funktion. Sie müssen die Jacobsche Funktion $\partial f / \partial A$ sowie die nicht-lineare Funktion $f = f(X, A)$ im Formel-Element des Blockdiagramms des Ziel-Fkt. & nicht-lineare Abltg.-VIs angeben. Diese Berechnung ist effizienter als die numerische, da es keine numerische Approximation an die Jacobsche Funktion voraussetzt.

Die Eingabearrays **X** und **Y** definieren den Satz der Eingabedatenpunkte. Das VI setzt voraus, daß Sie die nicht-lineare Beziehung zwischen x und y Koordinaten bereits kennen. Das heißt, $f = f(X, A)$, wobei der Koeffizientensatz A vom Levenberg-Marquardt-Algorithmus bestimmt wird.

Der erfolgreiche Einsatz dieser Funktion hängt bisweilen davon ab, wie genau die von Ihnen geschätzten Koeffizienten der Lösung entsprechen. Deshalb lohnt es sich stets, sich die Zeit zu nehmen und sich zu bemühen, anhand aller für die Lösung verfügbaren Ressourcen eine möglichst gute Schätzung der Koeffizienten vorzunehmen, bevor die Funktion eingesetzt wird.

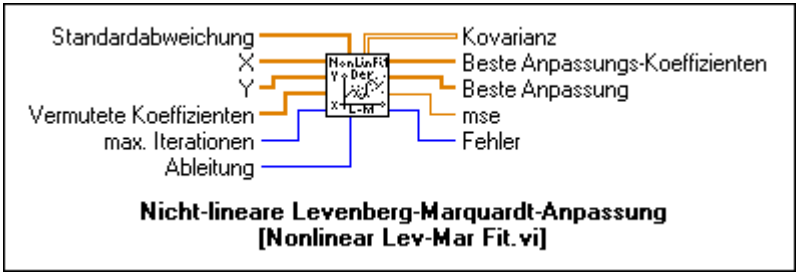
Verwenden des nicht-linearen Levenberg-Marquardt-Anpassung-VIs

Bis jetzt haben Sie VIs kennengelernt, die bei einer linearen Beziehung zwischen *y* und den Koeffizienten a_0, a_1, a_2, ... eingesetzt werden. Wenn jedoch eine nicht-lineare Beziehung vorliegt, können Sie das nicht-lineare Levenberg-Marquardt-Anpassung-VI verwenden, um die Koeffizienten zu ermitteln. Das VI verwendet die sehr robuste Levenberg-Marquardt-Methode zur Ermittlung der Koeffizienten **A** = $\{a_0, a_1, a_2, ..., a_k\}$ der nicht-linearen Beziehung zwischen **A** und y[i]. Das VI setzt voraus, daß Sie die nicht-lineare Beziehung zwischen den *x*- und *y*-Koordinaten bereits kennen.

Zunächst müssen Sie die nicht-lineare Funktion im Formel-Element des Blockdiagramms eines der SubVIs des nicht-linearen Levenberg-Marquardt-Anpassung-VIs festlegen. Dieses bestimmte SubVI repräsentiert das Ziel-Fkt. & nicht-lineare Abltg.-VI. Sie können darauf zugreifen, indem Sie es im Menü auswählen, das angezeigt wird, wenn Sie **Projekt»SubVIs des VIs** wählen.

Hinweis ***Beim Einsatz des nicht-linearen Levenberg-Marquardt-Anpassung-VIs müssen Sie außerdem die nicht-lineare Funktion im Formel-Element des Blockdiagramms des Ziel-Fkt. & nicht-lineare Abltg.-VIs angeben.***

Die Verbindungen zum nicht-linearen Levenberg-Marquardt-Anpassung-VI werden in der folgenden Abbildung dargestellt:



X und **Y** stellen die Eingabedatenpunkte x[i] und y[i] dar.

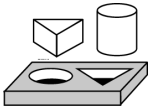
Initial Guess Coefficients stellen Ihre erste Schätzung bezüglich der Koeffizientenwerte dar. Die Koeffizienten sind die Werte, die in der Formel verwendet werden, die Sie in das **Formel-Element** des **Ziel-Fkt. & nicht-lineare Abltg.**-VI eingegeben haben. Der erfolgreiche Einsatz des **nicht-linearen Levenberg-Marquardt-Anpassung**-VIs hängt manchmal davon

ab, wie genau die von Ihnen geschätzten Koeffizienten der tatsächlichen Lösung entsprechen. Deshalb lohnt es sich stets, sich die Zeit zu nehmen und sich zu bemühen, die Lösung anhand aller verfügbaren Ressourcen so gut wie möglich vorauszuschätzen.

Im Moment können Sie für alle anderen Eingaben die Standardwerte beibehalten. Weitere Informationen über diese Eingaben entnehmen Sie der *Analyse-Online-Referenz.*

Koeffizienten für beste Anpassung: die Werte der Koeffizienten (a_0, a_1, ...), die am besten zum Modell der Versuchsdaten passen.

Übung 17-3. Nicht-lineares Levenberg-Marquardt-Anpassung-VI verwenden

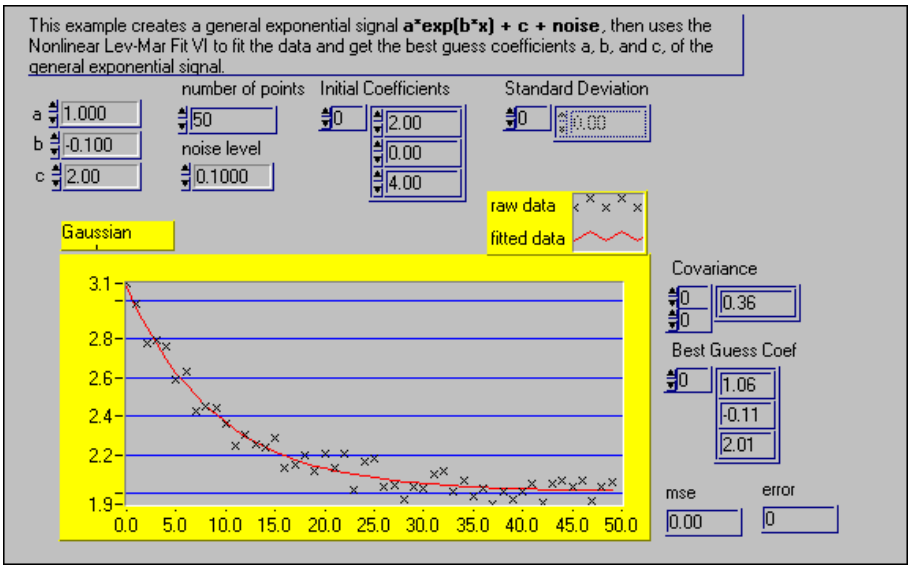


Das Übungsziel ist, ein allgemeines exponentielles Signal a*exp(b*x) + c + Rauschen zu erzeugen, und anschließend die Daten mit Hilfe des nicht-linearen Levenberg-Marquardt-Anpassung-VIs anzupassen und die besten geschätzten Koeffizienten a, b und c des allgemeinen exponentiellen Signals zu ermitteln.

Bei dieser Übung lernen Sie, wie das nicht-lineare Levenberg-Marquardt-Anpassung-VI verwendet wird, um die Koeffizienten a, b und c einer nicht-linearen Funktion zu bestimmen, die durch a*exp(b*x) + c gegeben ist.

Frontpanel

1. Öffnen Sie das nicht-lineare Levenberg-Marquardt-Anpassung-VI aus der Bibliothek `examples\analysis\regressn.llb`. Das Frontpanel ist in der folgenden Abbildung dargestellt.



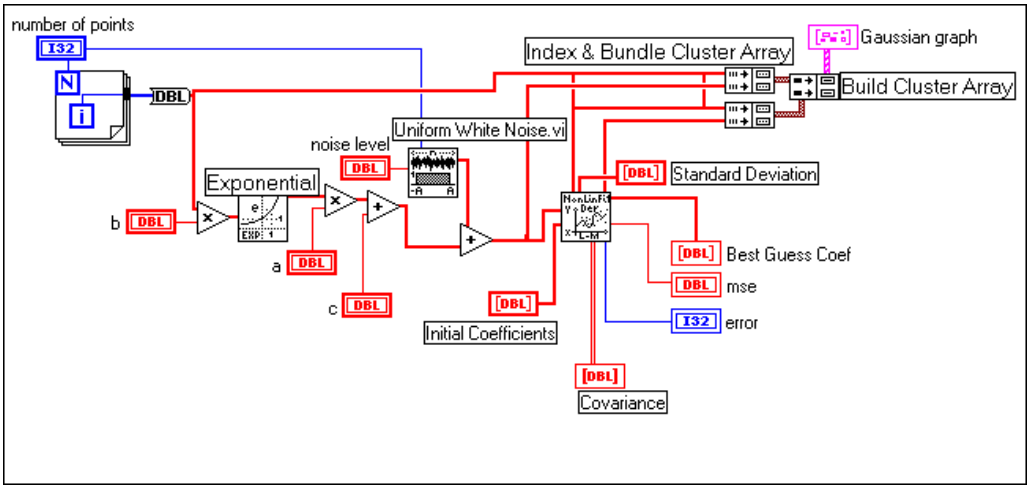
Die Bedienelemente **a**, **b** und **c** bestimmen die tatsächlichen Werte der Koeffizienten *a*, *b* und *c*. Das Bedienelement **Initialkoeffizienten** ist Ihre beste Schätzung der tatsächlichen Werte von *a*, *b* und *c*. Das Anzeigeelement **Ermittelte Koeffizienten** enthält die Werte von *a*, *b* und *c*, die vom **nicht-linearen Levenberg-Marquardt-Anpassung**-VI berechnet wurden. Um ein praxisnäheres Beispiel zu simulieren, wird dieser Gleichung obendrein Rauschen hinzugefügt, so daß sie jetzt folgendermaßen aussieht:

a*exp(b*x) + c + Rauschen

Das Bedienelement **Rausch-Level** reguliert den Rauschpegel. Beachten Sie, daß die tatsächlichen gewählten Werte von a, b und c +1,0, –0,1 und 2,0 betragen. Im Bedienelement **Initialkoeffizienten** beträgt die Standardschätzung für diese Werte *a* = 2,0, *b* = 0 und *c* = 4,0.

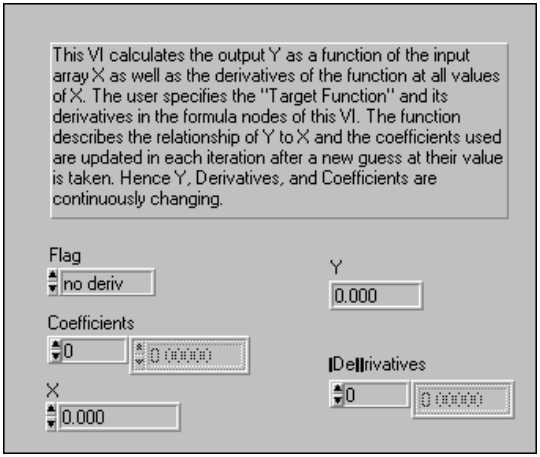
Blockdiagramm

2. Untersuchen Sie das Blockdiagramm.

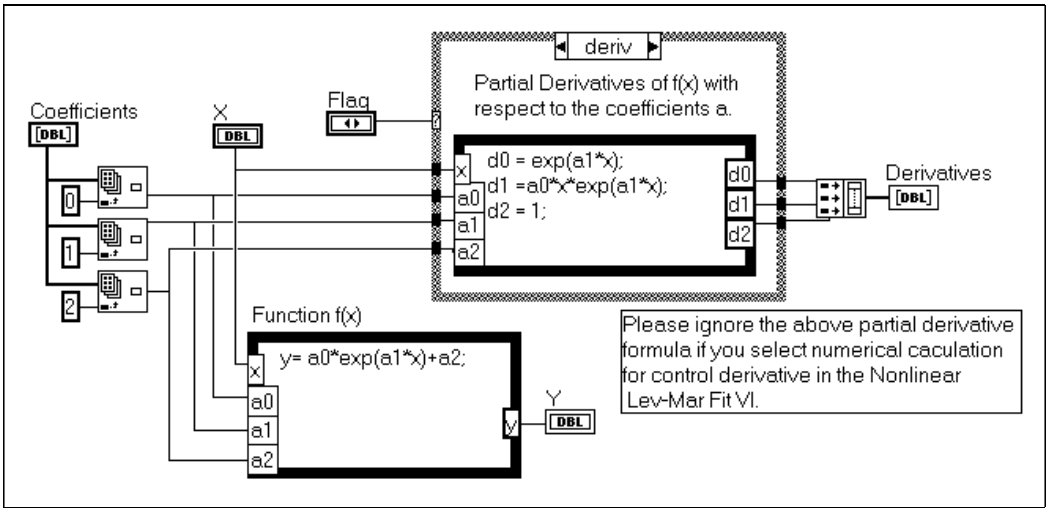


Die Datenabtastwerte der exponentiellen Funktion werden mit dem Exponential-VI (Unterpalette **Numerisch»Logarithmisch**) simuliert, und es wird ihnen mit Hilfe des Uniformes weißes Rauschen-VIs (Unterpalette **Analysis»Signalerzeugung**) uniformes weißes Rauschen hinzugefügt.

3. Im Menü **Projekt** wählen Sie **Ungeöffnete SubVIs»Ziel-Fkt. & nicht-lineare Abltg.**-VI. Das Frontpanel des Ziel-Fkt. & nicht-lineare Abltg.-VIs wird geöffnet (wie unten dargestellt).



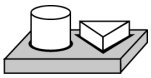
4. Schalten Sie zum Blockdiagramm um.



Beobachten Sie das Formel-Element am unteren Rand. Es hat die Form der Funktion, deren Parameter (a0, a1, und a2) Sie zu berechnen versuchen.

5. Schließen Sie das Frontpanel und das Blockdiagramm des Ziel-Fkt. & nicht-lineare Abltg.-VIs.
6. Führen Sie das nicht-lineare Levenberg-Marquardt Exponentialanpassung-VI aus. Beachten Sie, daß die Werte der Koeffizienten, die in **Ermittelte Koeffizienten** zurückgegeben werden, sehr nahe an die tatsächlichen im Bedienelement **Initialkoeffizienten** eingegebenen Werte herankommen. Beachten Sie auch die Werte des mittleren quadratischen Fehlers.
7. Erhöhen Sie den **Rausch-Level** von 0,1 auf 0,5. Was geschieht mit dem mittleren quadratischen Fehler und den Koeffizientenwerten in **Ermittelte Koeffizienten**? Warum?
8. Stellen Sie den **Rausch-Level** zurück auf 0,1 und die **Initialkoeffizienten** auf 5,0, –2,0 und 10,0, und führen Sie das VI aus. Beachten Sie die in den Anzeigeelementen **Ermittelte Koeffizienten** und **mse** zurückgegebenen Werte.

9. Während der **Rausch-Level** weiterhin auf 0,1 eingestellt ist, ändern Sie Ihre Schätzung für die **Initial Coefficients** auf 5,0, 8,0 und 10,0 und führen anschließend das VI aus. Diesmal ist Ihre Schätzung nicht so genau wie in Schritt 4. Beachten Sie den Fehler. Dies beweist, wie wichtig es ist, eine verhältnismäßig gute Schätzung für die Koeffizienten zu erzielen.
10. Wenn Sie die Übung beendet haben, schließen Sie das VI. Bitte speichern Sie Ihre Änderungen nicht.



Ende der Übung 17-3.

18

Lineare Algebra

In diesem Kapitel wird beschrieben, wie die Lineare Algebra-VIs eingesetzt werden, um Matrixberechnungen und -analysen durchzuführen. Beispiele zum Einsatz der Lineare Algebra-VIs entnehmen Sie bitte der Bibliothek `examples\analysis\linxmpl.llb`.

Lineare Systeme und Matrixanalysen

Gleichungssysteme der linearen Algebra kommen bei vielen Anwendungen vor, die wissenschaftliche Berechnungen umfassen, wie z.B. Signalverarbeitung, berechenbare Flüssigkeitsdynamik u.a. Solche Systeme können auf natürliche Weise vorkommen oder das Ergebnis der Approximation von Differentialgleichungen durch algebraische Gleichungen sein.

Matrixtypen

Gleich, um welche Anwendung es sich handelt, es ist stets notwendig, auf sehr effiziente Weise eine genaue Lösung für das Gleichungssystem zu finden. Bei der Matrix-Vektor-Darstellung sieht solch ein System von linearen Algebragleichungen folgendermaßen aus:

$$Ax = b,$$

wobei A eine $n \times n$ -Matrix, b ein gegebener Vektor aus n Elementen und x der unbekannte Lösungsvektor ist, der bestimmt werden soll. Eine Matrix wird durch ein 2D-Array von Elementen dargestellt. Diese Elemente können reale Zahlen, komplexe Zahlen, Funktionen oder Operatoren sein. Die unten dargestellte Matrix A ist ein Array von m Zeilen und n Spalten mit $m \times n$ -Elementen.

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix}$$

Hier bezeichnet $a_{i,j}$ das *(i,j)te* Element, das sich in der i^{ten} Reihe und j^{ten} Spalte befindet. Im allgemeinen wird diese Matrix als *rechteckige Matrix* bezeichnet. Wenn $m = n$, so daß die Anzahl der Zeilen mit der Anzahl der Spalten übereinstimmt, wird die Matrix *quadratische Matrix* genannt. Eine $m \times 1$-Matrix (*m* Zeilen in einer Spalte) wird als *Spaltenvektor* bezeichnet. Ein *Zeilenvektor* ist eine $1 \times n$-Matrix (1 Zeile und *n* Spalten). Wenn alle Elemente außer den diagonalen Elementen Null betragen (das heißt, $a_{i,j} = 0,\ i \neq j$), wird die Matrix als *diagonale Matrix* bezeichnet. Das Beispiel

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

stellt eine diagonale Matrix dar. Eine diagonale Matrix, deren diagonale Elemente alle gleich Eins sind, wird als *Identitätsmatrix* bzw. *Einheitsmatrix* bezeichnet. Wenn alle Elemente unterhalb der Hauptdiagonalen gleich Null sind, wird die Matrix als *obere Dreiecksmatrix* bezeichnet. Im Gegensatz dazu wird eine Matrix, deren Elemente unterhalb der Hauptdiagonalen alle gleich Null sind, als *untere Dreiecksmatrix* bezeichnet. Wenn alle Elemente reale Zahlen sind, wird die Matrix als *reale Matrix* bezeichnet. Wenn mindestens eines der Elemente der Matrix eine komplexe Zahl ist, wird sie als *komplexe Matrix* bezeichnet. Der Einfachheit halber arbeiten Sie in dieser Lektion hauptsächlich mit realen Matrizen. Falls Sie es sich zutrauen, gibt es jedoch auch einige Übungen mit komplexen Matrizen.

Determinanten einer Matrix

Eines der wichtigsten Attribute einer Matrix ist ihre *Determinante*. Im einfachsten Fall wird die Determinante einer 2 x 2-Matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

durch $ad - bc$ gegeben. Die Determinante einer quadratischen Matrix erhält man aus den Determinanten ihrer Elemente. Wenn beispielsweise

$$A = \begin{bmatrix} 2 & 5 & 3 \\ 6 & 1 & 7 \\ 1 & 6 & 9 \end{bmatrix},$$

dann lautet die Determinante von **A**, die mit $|A|$ bezeichnet wird,

$$|A| = \left|\begin{bmatrix} 2 & 5 & 3 \\ 6 & 1 & 7 \\ 1 & 6 & 9 \end{bmatrix}\right| = \left(2\left|\begin{bmatrix} 1 & 7 \\ 6 & 9 \end{bmatrix}\right| - 5\left|\begin{bmatrix} 6 & 7 \\ 1 & 9 \end{bmatrix}\right| + 3\left|\begin{bmatrix} 6 & 1 \\ 1 & 6 \end{bmatrix}\right|\right) =$$

$$2(-33) - 5(47) + 3(35) = -196.$$

Die Determinante gibt über viele wichtige Eigenschaften der Matrix Auskunft. Wenn die Determinante der Matrix z.B. Null ist, handelt es sich um eine *singuläre* Matrix. Folglich ist die oben dargestellte Matrix (deren Determinante ungleich Null ist) *nichtsingulär*. Das Konzept der Singularität taucht später im Abschnitt *Matrixinverse und Lösen von linearen Gleichungssystemen* erneut auf, wenn die Lektion die Lösung von linearen Gleichungen und Matrixinversen behandelt.

Die Transponierte einer Matrix

Die *Transponierte* einer realen Matrix wird gebildet, indem ihre Zeilen und Spalten miteinander vertauscht werden. Wenn die Matrix B die Transponierte von A darstellt, die mit $\mathbf{A}^T$ bezeichnet wird, dann gilt $b_{j,i}=a_{i,j}$. Für die oben definierte Matrix A gilt dann

$$B = A^T = \begin{bmatrix} 2 & 6 & 1 \\ 5 & 1 & 6 \\ 3 & 7 & 9 \end{bmatrix}$$

Bei komplexen Matrizen wird die konjugiert-komplexe Transposition definiert. Wenn die Matrix D die *konjugiert-komplexe Transponierte* (wenn $a = x + iy$, dann ist die konjugiert-Komplexe $a^* = x - iy$) einer komplexen Matrix C darstellt, dann gilt

$$D = C^H \Rightarrow d_{i,j} = c^*_{j,i}$$

Das bedeutet, daß man Matrix D erhält, indem jedes Element in C durch seine konjugiert-Komplexe ersetzt wird und die Zeilen und Spalten der resultierenden Matrix anschließend miteinander vertauscht werden.

Eine reale Matrix wird als *symmetrische Matrix* bezeichnet, wenn die Transponierte der Matrix mit der Matrix übereinstimmt. Die Beispielmatrix *A* ist keine symmetrische Matrix. Wenn eine komplexe Matrix C der Beziehung $C = C^H$ entspricht, dann wird *C hermetische Matrix* genannt.

Ist es möglich, einen Vektor durch Bildung der Linearkombination von anderen Vektoren zu erhalten? (Lineare Unabhängigkeit)

Ein Satz von Vektoren x_1, x_2, ..., x_n wird genau dann als *linear abhängig* bezeichnet, wenn es Skalare gibt, die nicht alle gleich Null sind, so daß

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n = 0 .$$

Einfacher ausgedrückt, wenn einer der Vektoren als Linearkombination der anderen geschrieben werden kann, werden die Vektoren als linear abhängig bezeichnet.

Wenn der einzige Satz von α_i, für den die Gleichung oben zutrifft, $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$ ist, dann ist der Vektorsatz x_1, x_2, ..., x_n *linear unabhängig*. In diesem Fall kann keiner der Vektoren als Linearkombination der anderen geschrieben werden. Bei einem gegebenen Vektorsatz trifft die Gleichung oben für $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$ stets zu. Um die lineare Unabhängigkeit eines Satzes zu beweisen, müssen Sie deshalb beweisen, daß $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$ der einzige Satz von α_i ist, für den die Gleichung gilt.

Betrachten Sie z.B. zunächst die Vektoren

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 3 \\ 4 \end{bmatrix} .$$

Beachten Sie, daß $\alpha_1 = 0$ und $\alpha_2 = 0$ die einzigen Werte sind, bei denen die Beziehung $\alpha_1 x + \alpha_2 y = 0$ gilt. Also sind diese beiden Vektoren linear unabhängig voneinander. Betrachten Sie nun die Vektoren

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Beachten Sie, wenn $\alpha_1 = -2$ und $\alpha_2 = 1$, dann gilt $\alpha_1 x + \alpha_2 y = 0$. Deshalb sind diese beiden Vektoren linear abhängig voneinander. Sie müssen diese Definition der linearen Unabhängigkeit von Vektoren völlig verstehen, um das Konzept des *Matrixranges*, das als nächstes beschrieben wird, vollkommen zu begreifen.

Wie wird die lineare Unabhängigkeit bestimmt? (Rang einer Matrix)

Der *Rang* einer Matrix *A*, der mit $\rho(A)$ bezeichnet wird, ist die maximale Anzahl der linear unabhängigen Spalten in *A*. Wenn Sie die Beispielmatrix *A* betrachten, sehen Sie, daß alle Spalten von *A* linear unabhängig voneinander sind. Das heißt, das Sie keine der Spalten erhalten können, indem Sie eine Linearkombination der anderen Spalten bilden. Folglich ist der Rang der Matrix 3. Betrachten Sie eine weitere Beispielmatrix B, wobei

$$B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 0 & 2 \end{bmatrix}$$

Diese Matrix verfügt über lediglich zwei linear unabhängige Spalten, da die dritte Spalte von B von den ersten zwei Spalten linear abhängig ist. Folglich ist der Rang dieser Matrix 2. Es kann bewiesen werden, daß die Anzahl der linear unabhängigen Spalten einer Matrix mit der Anzahl der unabhängigen Zeilen übereinstimmt. Der Rang kann also niemals größer sein, als die kleinste Dimension der Matrix. Wenn *A* eine $n \times m$-Matrix ist, dann ist folglich

$$\rho(A) \leq min(n, m),$$

wobei *min* den Mindestwert der zwei Zahlen angibt. In der Matrixtheorie ist der Rang einer quadratischen Matrix zu der nichtsingulären Matrix der höchsten Ordnung zugehörig, die von ihr gebildet werden kann. Erinnern Sie sich, daß eine Matrix singulär ist, wenn ihre Determinante Null beträgt. Der Rang ist also zu der Matrix der höchsten Ordnung zugehörig, die Sie erhalten können und deren Determinante ungleich Null ist. Betrachten Sie z.B. eine 4 x 4-Matrix

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 2 \end{bmatrix}$$

Für diese Matrix gilt $det(B) = 0$, aber

$$\left|\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}\right| = -1$$

Folglich ist 3 der Rang von *B*. Eine quadratische Matrix besitzt genau dann einen ganzen Rang, wenn ihre Determinante ungleich Null ist. Matrix B ist keine Matrix mit ganzem Rang.

Betrag (Normen) von Matrizen

Zur Messung von Fehlern müssen Sie einen Sinn für den "Betrag" von Vektoren und Matrizen entwickeln sowie Sensibilität für die Lösung eines linearen Gleichungssystems. Beispielsweise können Sie diese linearen Systeme aus Anwendungen in den Bereichen Steuersysteme und berechenbare Flüssigkeitsdynamik erhalten. Bei zwei Dimensionen können Sie z.B. nicht zwei Vektoren $\boldsymbol{x} = \begin{bmatrix} x1 & x2 \end{bmatrix}$ und $y = \begin{bmatrix} y1 & y2 \end{bmatrix}$ vergleichen, da möglicherweise $x1 > y1$ aber $x2 < y2$ ist. Eine Vektornorm ist eine Methode, um diesen Vektoren eine skalare Menge zuzuweisen, so daß sie miteinander verglichen werden können. Dies entspricht dem Konzept des Betrags bzw. absoluten Betrags oder Werts für skalare Zahlen.

Es gibt Methoden zur Berechnung der Norm einer Matrix. Dazu gehören die *2-Norm* (Euklidische Norm), die *1-Norm*, die *Frobenius Norm* (F-Norm) sowie die *unendliche Norm* (Inf-Norm). Jede Norm verfügt über eine eigene physikalische Interpretation. Betrachten Sie eine Elementarkugel, die den Ursprung enthält. Die euklidische Norm stellt einfach den Faktor dar, um den die Kugel vergrößert oder verkleinert werden muß, damit sie den gegebenen Vektor genau umschließt. Dies ist in den folgenden Abbildungen dargestellt:

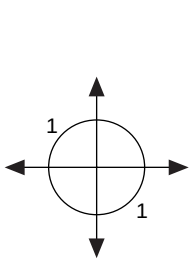


Abbildung 1a

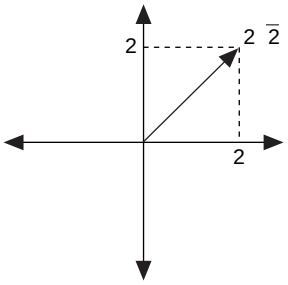


Abbildung 1b

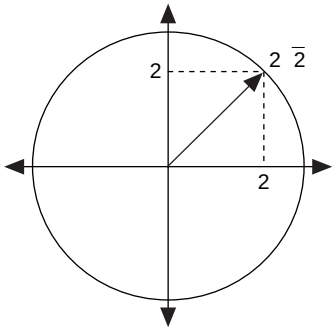


Abbildung 1c

Abbildung 1a zeigt die Elementarkugel mit dem Radius = 1 Einheit. Abbildung 1b zeigt einen Vektor der Länge $\sqrt{2^2 + 2^2} = \sqrt{8} = 2\sqrt{2}$. Wie in Abbildung 1c gezeigt, muß die Einheitskugel um einen Faktor von $2\sqrt{2}$ erweitert werden, bevor sie den gegebenen Vektor genau umschließen kann. Folglich lautet die euklidische Norm des Vektors $2\sqrt{2}$.

Die Norm einer Matrix wird hinsichtlich einer zugrundeliegenden Vektornorm definiert. Sie ist die maximale relative Dehnung, mit der der Vektor durch die Matrix gestreckt wird. Mit der Vektor *2-Norm* wird die Einheitskugel um einen Faktor vergrößert, der der Norm entspricht. Auf der anderen Seite kann mit der Matrix *2-Norm* aus der Einheitskugel ein Ellipsoid (3-D-Ellipse) werden, bei dem einige Achsen länger als die anderen sind. Die längste Achse bestimmt die Matrixnorm.

Einige Matrixnormen sind einfacher zu berechnen als andere. Die *1-Norm* wird ermittelt, indem man die Summe des absoluten Wertes aller Elemente in jeder Spalte der Matrix findet. Die größte dieser Summen wird *1-Norm* genannt. Mathematisch gesehen ist die 1-Norm einfach die höchste absolute Summe der Spalten einer Matrix.

$$\|\boldsymbol{A}\|_1 = max_j \sum_{i=0}^{n-1} |a_{i,j}|$$

Beispielsweise ist

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}.$$

Dann ist

$$\|A\|_1 = max(3, 7) = 7.$$

Die *Inf-Norm* einer Matrix ist die maximale absolute Summe der Zeilen der Matrix

$$\|\boldsymbol{A}\|_\infty = max_i \sum_{j=0}^{n-1} |a_{i,j}|$$

In diesem Fall addieren Sie den Betrag aller Elemente aus jeder Matrixzeile. Der sich daraus ergebende maximale Werte wird mit Inf-Norm bezeichnet. Für die Beispielmatrix oben gilt:

$$\|\boldsymbol{A}\|_\infty = max(4, 6) = 6.$$

Die *2-Norm* ist am schwierigsten zu berechnen, da sie durch den größten singulären Wert der Matrix definiert wird. Singuläre Werte werden im Abschnitt *Matrixfaktorisierung* beschrieben.

Bestimmen der Singularität (Bedingungszahl)

Während die Norm der Matrix eine Methode für die Messung der Größe einer Matrix liefert, ist die *Bedingungszahl* einer Matrix ein Maß dafür, wie nahe die Matrix daran ist, singulär zu sein. Die Bedingungszahl einer quadratischen nichtsingulären Matrix wird folgendermaßen definiert:

$$cond(\boldsymbol{A}) = \|\boldsymbol{A}\|_p \cdot \|A^{-1}\|_p,$$

wobei *p* eine der vier oben beschriebenen Normentypen sein kann. Um die Bedingungszahl einer Matrix *A* zu ermitteln, können Sie die 2-Norm von *A*, und die 2-Norm der Inversen der Matrix *A*, die mit A^{-1}bezeichnet wird, ermitteln und anschließend beide miteinander multiplizieren. (Die Inverse einer quadratischen Matrix A ist eine quadratische Matrix B, so daß $B=I$, wobei I die Identitätsmatrix ist.) Wie bereits erwähnt, ist die 2-Norm schwierig auf Papier zu berechnen. Sie können dazu das **Matrix Norm**-VI aus der LabVIEW Analysis-Bibliothek verwenden. Ein Beispiel:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, A^{-1} = \begin{bmatrix} -2 & 1 \\ 1,5 & -0,5 \end{bmatrix}, \|A\|_2 = 5,4650, \left\|A^{-1}\right\|_2$$

$$= 2,7325, cond(A) = 14,9331$$

Die Bedingungszahl kann zwischen 1 und unendlich liegen. Eine Matrix mit einer hohen Bedingungszahl ist nahezu singulär, während eine Matrix mit einer Bedingungszahl nahe 1 weit davon entfernt ist, singulär zu sein. Die Matrix *A* oben ist nichtsingulär. Betrachten Sie jedoch die folgende Matrix:

$$B = \begin{bmatrix} 1 & 0,99 \\ 1,99 & 2 \end{bmatrix}.$$

Die Bedingungszahl dieser Matrix lautet 47168, und folglich ist die Matrix nahe daran, singulär zu sein. Erinnern Sie sich, daß eine Matrix singulär ist, wenn ihre Determinante gleich Null ist. Die Determinante ist jedoch kein guter Indikator, um zu beurteilen, wie nahe die Matrix daran ist, singulär zu sein. Bei der Matrix *B* oben ist die Determinante (0,0299) ungleich Null. Die hohe Bedingungszahl weist jedoch darauf hin, daß die Matrix nahe daran ist, singulär zu sein. Erinnern Sie sich, daß die Bedingungszahl immer größer oder gleich Eins ist, wobei letzteres auf Identitäts- und Permutationsmatrizen (eine Permutationsmatrix ist eine Identitätsmatrix, bei der einige Spalten und Zeilen vertauscht wurden) zutrifft. Die Bedingungszahl ist ein sehr nützlicher Betrag zur Beurteilung der Genauigkeit von Lösungen für Linearsysteme.

In diesem Abschnitt haben Sie sich mit der grundlegenden Darstellung und den fundamentalen Konzepten von Matrizen, wie Determinanten und Ränge, vertraut gemacht. Die folgende Übung soll Ihnen helfen, diese Begriffe, die in der gesamten Lektion häufig verwendet werden, noch besser zu verstehen.

Grundlegende Matrixoperationen und Eigenwert-Eigenvektor-Probleme

In diesem Abschnitt werden einige sehr grundlegende Matrixoperationen behandelt. Zwei Matrizen, *A* und *B*, stimmen überein, wenn sie die gleiche Anzahl von Zeilen und Spalten enthalten und die dazugehörigen Elemente alle gleich sind. Die Multiplikation einer Matrix *A* mit einem Skalar α entspricht der Multiplikation aller ihrer Elemente mit dem Skalar. Das heißt,

$$\boldsymbol{C} = \alpha\boldsymbol{A} \Rightarrow c_{i,j} = \alpha a_{i,j}.$$

Beispiel:

$$2\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} = \begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix}.$$

Zwei (oder mehr) Matrizen können dann und nur dann addiert oder subtrahiert werden, wenn sie über die gleiche Anzahl von Zeilen und Spalten verfügen. Wenn beide Matrizen *A* und *B* *m* Zeilen und *n* Spalten aufweisen, ist ihre Summe *C* eine *m*-mal-*n*-Matrix, die als $C = A \pm B$ definiert ist, wobei $c_{i,j} = a_{i,j} \pm b_{i,j}$. Beispiel:

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} + \begin{bmatrix}2 & 4\\ 5 & 1\end{bmatrix} = \begin{bmatrix}3 & 6\\ 8 & 5\end{bmatrix}$$

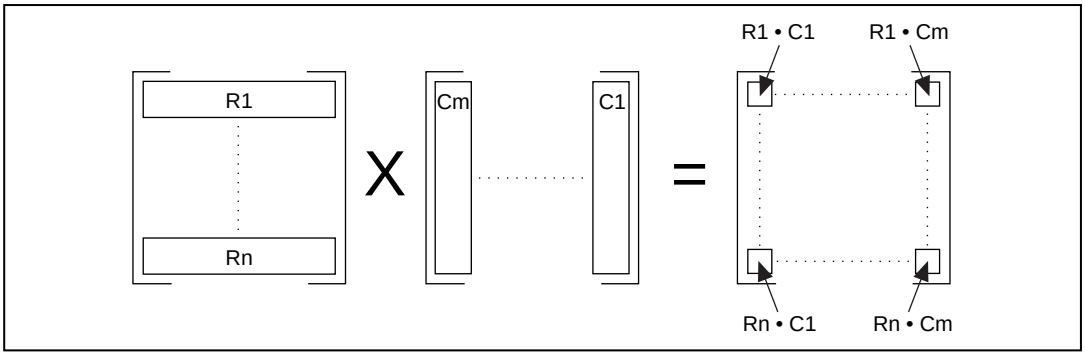
Zur Multiplikation von zwei Matrizen muß die Anzahl der Spalten der ersten Matrix mit der Anzahl der Zeilen der zweiten Matrix übereinstimmen. Wenn Matrix *A* über *m* Zeilen und *n* Spalten und Matrix *B* über *n* Zeilen und *p* Spalten verfügt, dann ist ihr Produkt *C* eine als $C = AB$ definierte *m*-mal-*p*-Matrix, wobei

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}$$

Beispiel:

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \times \begin{bmatrix}2 & 4\\ 5 & 1\end{bmatrix} = \begin{bmatrix}12 & 6\\ 26 & 16\end{bmatrix}$$

Sie multiplizieren also die Elemente der ersten Zeile von A mit den entsprechenden Elementen der ersten Spalte von *B* und addieren alle Ergebnisse, um die Elemente in der ersten Zeile und ersten Spalte von *C* zu erhalten. Zur Berechnung der Elemente der i^{ten} Zeile und der j^{ten} Spalte von *C* gehen Sie ähnlich vor. Multiplizieren Sie die Elemente der i^{ten} Zeile von *A* mit den entsprechenden Elementen in der j^{ten} Spalte von *C*, und addieren Sie die Ergebnisse. Bildlich dargestellt sieht dies folgendermaßen aus:



Die Matrixmultiplikation ist im allgemeinen nicht kommutativ, d. h., $AB \neq BA$. Denken Sie deshalb daran, daß die Multiplikation einer Matrix mit einer Identitätsmatrix die Originalmatrix ergibt.

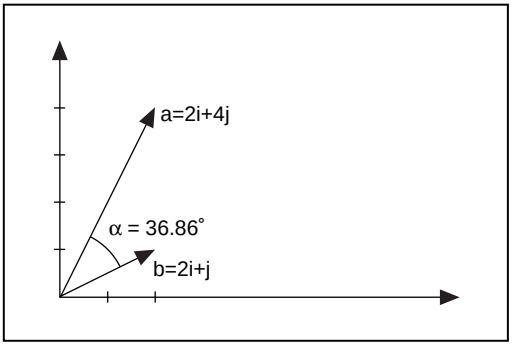
Skalares Produkt und Vektorprodukt

Wenn *X* für einen Vektor und *Y* für einen anderen Vektor steht, dann erhält man das *skalare Produkt* dieser beiden Vektoren, indem man die entsprechenden Elemente jedes Vektors multipliziert und die Ergebnisse addiert. Dies wird durch die folgende Gleichung dargestellt:

$$X \bullet Y = \sum_{i=0}^{n-1} x_i y_i \quad ,$$

wobei *n* die Anzahl der Elemente in *X* und *Y* ist. Beachten Sie, daß beide Vektoren über die gleiche Anzahl von Elementen verfügen müssen. Das skalare Produkt ist eine skalare Größe und besitzt viele praktische Anwendungen.

Betrachten Sie beispielsweise die Vektoren $a = 2i + 4j$ und $b = 2i + j$ in einem zweidimensionalen rechteckigen Koordinatensystem.



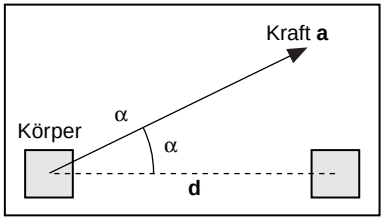
Folglich wird das skalare Produkt dieser beiden Vektoren durch folgende Gleichung dargestellt:

$$d = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \bullet \begin{bmatrix} 2 \\ 1 \end{bmatrix} = (2 \times 2) + (4 \times 1) = 8$$

Der Winkel α zwischen diesen beiden Vektoren wird durch folgende Gleichung dargestellt:

$$\alpha = inv\cos\left(\frac{a \bullet b}{|a||b|}\right) = inv\cos\left(\frac{8}{10}\right) = 36,86°,$$

wobei |**a**| den Betrag von *a* bezeichnet.



Als zweite Anwendung betrachten Sie einen Körper, auf den eine konstante Kraft *a* einwirkt. Die Arbeit W, die von *a* beim Verschieben des Körpers geleistet wird, ist als Produkt von |*d*| und der Komponente von *a* in Richtung der Verschiebung definiert. Das bedeutet,

$$W = |a||d|\cos\alpha = a \bullet d$$

Auf der anderen Seite ist das *Vektorprodukt* dieser zwei Vektoren eine Matrix. Das $(i,j)^{te}$ Element dieser Matrix erhält man durch die Formel

$$a_{i,j} = x_i \times y_j$$

Beispiel:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \times \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}$$

Eigenwerte und Eigenvektoren

Um Eigenwerte und -vektoren verstehen zu können, gehen Sie von der klassischen Definition aus. Die $n \times n$-Matrix A ist gegeben, und nun soll ein Skalar λ und ein von Null abweichender Vektor x gefunden werden, so daß

$$Ax = \lambda x .$$

Dieser Skalar λ wird *Eigenwert* genannt, und x ist der entsprechende *Eigenvektor.*

Die Berechnung der Eigenwerte und -vektoren gehört zu den grundlegenden Prinzipien der linearen Algebra und ermöglicht Ihnen, viele Probleme, wie z.B. Systeme von Differentialgleichungen, zu lösen, wenn Sie verstehen, wofür sie stehen. Betrachten Sie einen Eigenvektor x einer Matrix A als einen von Null abweichenden Vektor, der sich nicht dreht, wenn x mit A multipliziert wird (außer vielleicht, wenn er in die genau entgegengesetzte Richtung zeigt). Die Länge von x kann sich ändern oder seine Richtung kann sich umkehren, aber er dreht sich nicht zur Seite. Es gibt also irgendeine skalare Konstante λ, so daß die Gleichung oben gilt. Der Wert λ ist ein *Eigenwert* von A.

Betrachten Sie das folgende Beispiel. Einer der Eigenvektoren der Matrix A, wobei

$$A = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix},$$

lautet

$$x = \begin{bmatrix} 0,62 \\ 1,00 \end{bmatrix}.$$

Wenn die Matrix *A* mit dem Vektor *x* multipliziert wird, wird der Vektor *x* einfach um den Faktor 6,85 gestreckt. Folglich ist der Wert 6,85 einer der Eigenwerte des Vektors *x*. Für jede Konstante α stellt der Vektor αx auch einen Eigenvektor mit dem Eigenwert λ dar, weil

$$A(\alpha x) = \alpha Ax = \lambda \alpha x .$$

Ein Eigenvektor einer Matrix bestimmt also die Richtung, in die die Matrix jeden Vektor, der in dieser Richtung liegt, um ein skalares Vielfaches streckt oder staucht. Der Streckungs- bzw. Stauchungsfaktor wird durch den dazugehörigen Eigenwert gegeben. Ein *verallgemeinertes* Eigenwertproblem ist, einen Skalar λ und einen von Null abweichenden Vektor *x* zu finden, so daß

$$Ax = \lambda Bx ,$$

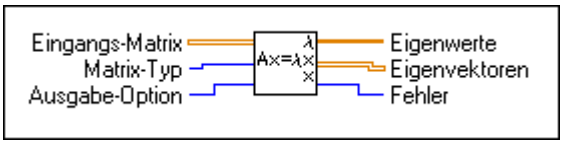
wobei *B* eine andere $n \times n$-Matrix darstellt.

Im folgenden werden einige wichtige Eigenschaften von Eigenwerten und -vektoren aufgezählt:

- Die Eigenwerte einer Matrix sind nicht unbedingt alle verschieden. Das heißt, eine Matrix kann mehrere Eigenwerte haben.
- Die Eigenwerte einer realen Matrix müssen nicht alle real sein. Komplexe Eigenwerte einer realen Matrix müssen jedoch in konjugiert-komplexen Paaren auftreten.
- Die Eigenwerte einer diagonalen Matrix sind ihre diagonalen Elemente, und die Eigenvektoren sind die entsprechenden Spalten einer Identitätsmatrix der gleichen Dimension.
- Eine reale symmetrische Matrix verfügt stets über reale Eigenwerte und Eigenvektoren.
- Wie bereits erwähnt, können Eigenvektoren willkürlich skaliert werden.

Im Bereich Wissenschaft und Technik gibt es viele praktische Anwendungen für Eigenwertprobleme. Die Stabilität einer Struktur und ihre natürlichen Vibrationsmodi und -frequenzen werden von den Eigenwerten und -vektoren einer geeigneten Matrix bestimmt. Eigenwerte sind zudem sehr nützlich bei der Analyse von numerischen Methoden, wie z.B. der Konvergenzanalyse von iterativen Methoden zur Lösung von algebraischen Gleichungssystemen und der Stabilitätsanalyse von Methoden zur Lösung von Differentialgleichungssystemen.

Das Eigenwerte und -vektoren-VI ist unten dargestellt. Die **Eingangs-Matrix** ist eine N x N reale quadratische Matrix. **Matrix-Typ** bestimmt den Typ der Eingangs-Matrix. **Matrix-Typ** könnte z.B. *0* sein, was bedeutet, daß es sich um eine *allgemeine Matrix* handelt, oder aber *1*, wobei es sich um eine *symmetrische Matrix* handeln würde. Eine symmetrische Matrix verfügt stets über reale Eigenwerte und -vektoren. Eine allgemeine Matrix hat keine besondere Eigenschaft, wie z.B. Symmetrie oder dreieckige Struktur.



Ausgabe-Option bestimmt, was berechnet werden muß. Ausgabe-Option = 0 bedeutet, daß nur die Eigenwerte berechnet werden müssen. Ausgabe-Option = 1 bedeutet, daß Eigenwerte und -vektoren berechnet werden müssen. Die Berechnung sowohl von Eigenwerten als auch von Eigenvektoren ist sehr aufwendig. Deshalb ist es wichtig, daß Sie das Bedienelement Ausgabe-Option im Eigenwerte- und vektoren-VI mit äußerster Vorsicht einsetzen. Je nach Ihrer jeweiligen Anwendung sollten Sie nur die Eigenwerte oder aber sowohl Eigenwerte als auch Eigenvektoren berechnen. Außerdem erfordert eine symmetrische Matrix weniger Berechnungen als eine nichtsymmetrische. Gehen Sie also bei der Auswahl des Bedienelements Matrix-Typ vorsichtig vor.

In diesem Abschnitt haben Sie einige grundlegende Matrixoperationen und das Eigenwert-Eigenvektor-Problem kennengelernt. Das nächste Beispiel befaßt sich mit einigen VIs in der Analysebibliothek, die diese Operationen ausführen.

Matrixinverse und Lösen von linearen Gleichungssystemen

Die durch A^{-1} bezeichnete *Inverse* einer quadratischen Matrix *A* ist eine quadratische Matrix, so daß

$$A^{-1}A = AA^{-1} = I,$$

wobei *I* die Identitätsmatrix darstellt. Die Inverse einer Matrix existiert dann und nur dann, wenn die Determinante der Matrix ungleich Null ist (d.h., nichtsingulär ist). Im allgemeinen können Sie die Inverse nur von quadratischen Matrizen ermitteln. Sie können jedoch die *Pseudoinverse* einer rechteckigen Matrix berechnen, wie später im Abschnitt *Matrixfaktorisierung* noch beschrieben wird.

Lösungen von linearen Gleichungssystemen

Bei der Matrix-Vektor-Darstellung hat ein System von linearen Gleichungen die Form $Ax = b$, wobei *A* die $n \times n$-Matrix und *b* der vorgegebene *n*-Vektor ist. Das Ziel ist, *x*, den unbekannten *n*-Lösungsvektor zu bestimmen. Zwei wichtige Fragen geben Aufschluß über die Existenz einer solchen Lösung. Gibt es eine solche Lösung, und, wenn es sie gibt, ist sie eindeutig? Die Antwort auf diese beiden Fragen finden Sie, indem Sie bestimmen, ob die Matrix singulär oder nichtsingulär ist.

Wie bereits beschrieben, ist eine Matrix singulär, wenn sie eine der folgenden gleichwertigen Eigenschaften besitzt:

- Es gibt keine Inverse der Matrix.
- Die Determinante der Matrix beträgt Null.
- Die Zeilen (oder Spalten) von *A* sind linear abhängig.
- $Az = 0$ für einen Vektor $z \neq 0$.

Andernfalls ist die Matrix nichtsingulär. Ist die Matrix nichtsingulär, hat sie eine Inverse A^{-1}, und das System $Ax = b$ hat eine eindeutige Lösung: $x = A^{-1}b$, egal welchen Wert *b* hat. Auf der anderen Seite, wenn die Matrix singulär ist, dann wird die Anzahl der Lösungen vom Vektor *b* auf der rechten Seite bestimmt. Wenn *A* singulär ist und $Ax = b$, dann gilt $A(x + \Upsilon z) = b$ für jeden Skalar Υ, wobei der Vektor z der oben stehenden Definition entspricht. Wenn also ein singuläres System eine Lösung hat, dann kann diese Lösung nicht eindeutig sein.

Es wird nicht empfohlen, die Inverse einer Matrix ausdrücklich zu berechnen, da solch eine Berechnung numerischen Ungenauigkeiten unterliegt. Deshalb sollte ein lineares System von Gleichungen nicht gelöst werden, indem die Inverse der Matrix *A* mit dem bekannten rechten Vektor multipliziert wird. Um solch ein System von Gleichungen zu lösen, ist das Originalsystem in eines umzuformen, dessen Lösung der des ursprünglichen Systems entspricht, jedoch einfacher zu berechnen ist. Eine Methode dafür ist die Gaußsche Eliminierungsmethode. Weitere Informationen zu Matrixberechnungen können Sie Anhang A, *Analyse-Referenzen*, entnehmen. Die Gaußsche Eliminierungsmethode umfaßt drei grundlegende Schritte. Zuerst müssen Sie die Matrix *A* als folgendes Produkt ausdrücken:

$$A = LU,$$

wobei *L* eine untere Einheits-Dreiecksmatrix und *U* eine obere Einheits-Dreiecksmatrix ist. Eine solche Faktorisierung wird als LU- Faktorisierung bezeichnet. Auf dieser Basis kann das Linearsystem $Ax = b$ durch $LUx = b$ ausgedrückt werden. Dieses System kann dann gelöst werden, indem zunächst das untere Dreiecksystem $Ly = b$ für *y* durch *Vorwärts-Substitution* gelöst wird. Dies ist der zweite Schritt der Gaußschen Eliminierungsmethode. Wenn z.B.

$$l = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix} \qquad y = \begin{bmatrix} p \\ q \end{bmatrix} \qquad b = \begin{bmatrix} r \\ s \end{bmatrix},$$

dann gilt

$$p = \frac{r}{a}, \quad q = \frac{(s - bp)}{c}.$$

Das erste Element von *y* kann problemlos bestimmt werden, da die Matrix *L* ein unteres Dreiecksystem ist. Dieser Wert kann dann verwendet werden, um die übrigen Elemente des unbekannten Vektors der Reihenfolge nach zu bestimmen. Daher die Bezeichnung Vorwärts-Substitution. Der letzte Schritt besteht darin, das obere Dreiecksystem $Ux = y$ durch *Rückwärts-Substitution* zu berechnen. Wenn z.B.

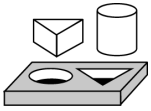
$$U = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix} \qquad x = \begin{bmatrix} m \\ n \end{bmatrix} \qquad y = \begin{bmatrix} p \\ q \end{bmatrix},$$

dann gilt

$$n = \frac{q}{c}, m = \frac{(p - bn)}{a}.$$

In diesem Fall kann das letzte Element von **x** problemlos bestimmt und dann dazu verwendet werden, die anderen Elemente der Reihenfolge nach zu berechnen. Daher der Name Rückwärts-Substitution. Bisher wurden in diesem Kapitel nur quadratische Matrizen behandelt. Da eine nichtquadratische Matrix zwangsläufig singulär ist, muß das Gleichungssystem entweder über keine Lösung oder eine nicht eindeutige Lösung verfügen. In solch einem Fall können Sie normalerweise eine eindeutige Lösung *x* finden, die den Anforderungen des Linearsystems annähernd entspricht.

Die Analysis-Bibliothek bietet VIs zur Berechnung der Inversen sowie der LU-Faktorisierung einer Matrix und zur Lösung eines Systems von Lineargleichungen. Es ist wichtig, die Eingangs-Matrix richtig zu erkennen, da so unnötige Berechnungen vermieden werden, wodurch wiederum numerische Ungenauigkeiten auf ein Mindestmaß reduziert werden. Die vier möglichen Matrixtypen sind allgemeine Matrizen, definit positive Matrizen und untere und obere Dreiecksmatrizen.Eine reale Matrix ist dann und nur dann definit positiv, wenn sie symmetrisch und die quadratische Form für alle Vektoren X ist. Wenn die Eingangs-Matrix quadratisch ist, aber keinen ganzen Rang hat (*eine Rang-defiziente Matrix*), dann ermittelt das VI die *kleinste quadratische* Lösung *x.* Die kleinste quadratische Lösung ist diejenige, die die Norm von $Ax - b$ auf ein Minimum reduziert. Das gleiche gilt für nichtquadratische Matrizen.



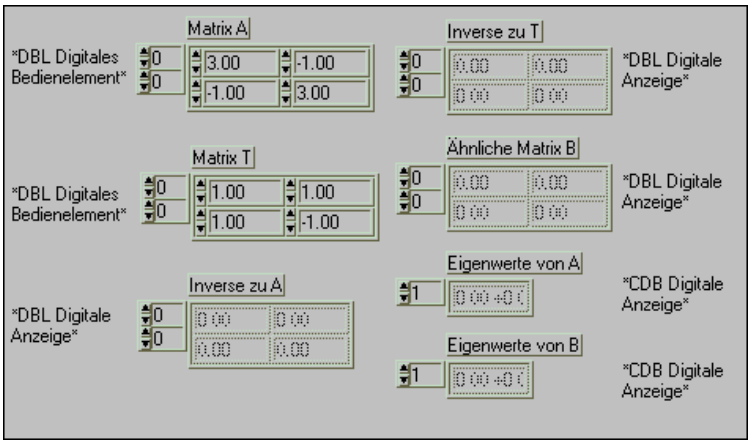
Übung 18-1. Die Inverse einer Matrix berechnen

Das Übungsziel ist, die Inverse einer Matrix zu berechnen.

Sie erstellen hier ein VI, das die Inverse einer Matrix *A* berechnet. Desweiteren berechnen Sie die Matrix *B*, die Matrix *A ähnelt.* Eine Matrix *B* ist einer Matrix *A* ähnlich, wenn es eine nichtsinguläre Matrix *T* gibt, so daß $B = T^{-1}AT$ gilt, so daß *A* und *B* die gleichen Eigenwerte haben. Überprüfen Sie diese Definition ähnlicher Matrizen.

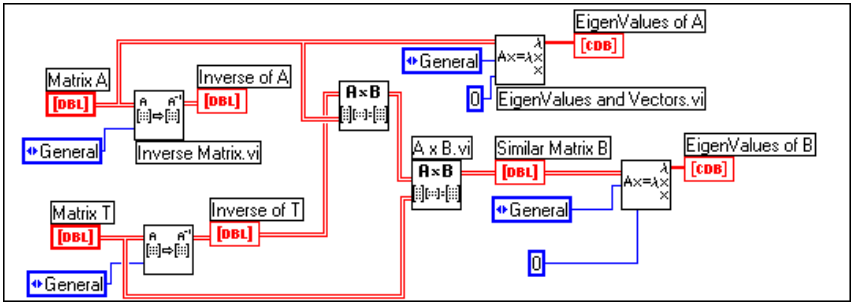
Frontpanel

1. Erstellen Sie das Frontpanel wie unten dargestellt. Matrix *A* ist eine reale 2×2 -Matrix. Matrix *T* ist eine nichtsinguläre 2×2-Matrix, die verwendet wird, um die ähnliche Matrix *B* zu erstellen.



Blockdiagramm

2. Öffnen Sie das Blockdiagramm, und ändern Sie es gemäß folgender Abbildung.

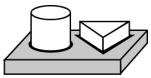


Inverse Matrix (Unterpalette **Analysis»Lineare Algebra**). Bei dieser Übung berechnet diese Funktion die Inverse der Eingangs-Matrix *A*.

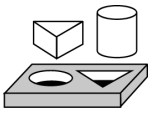
AxB-Funktion (Unterpalette **Analysis»Lineare Algebra**). Bei dieser Übung multipliziert diese Funktion zwei zweidimensionale Eingangs-Matrizen miteinander.

Eigenwerte und -vektoren (Unterpalette **Analysis»Lineare Algebra**). Bei dieser Übung berechnet diese Funktion die Eigenwerte und -vektoren der Eingangs-Matrix.

3. Speichern Sie das VI als `Matrix Inverse.vi` im Verzeichnis `LabVIEW\Activity`.
4. Kehren Sie zum Frontpanel zurück, und führen Sie das VI aus. Prüfen Sie, ob die Eigenwerte von *A* mit denen der ähnlichen Matrix *B* übereinstimmen.



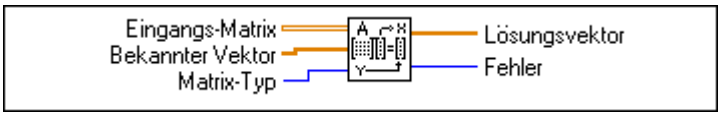
Ende der Übung 18-1.



Übung 18-2. Ein System von linearen Gleichungen lösen

Das Übungsziel ist, ein System von linearen Gleichungen zu lösen.

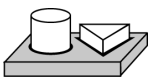
Viele praktische Anwendungen erfordern die Lösung eines Systems von linearen Gleichungen. Ein sehr wichtiger Einsatzbereich ist die militärische Verteidigung. Dazu gehört die Analyse der elektromagnetischen Streuung und Strahlung von großen Zielen, die Leistungsanalyse großer Radome und der Entwurf von Raumfahrzeugen mit kleinen Radarquerschnitten (Stealth-Technologie). Ein zweiter Anwendungsbereich ist der Entwurf und die Modellierung von drahtlosen Kommunikationssystemen, wie z.B. zellulare Handtelefone. Die Liste der Anwendungen ist lang, und aus diesem Grund ist es sehr wichtig, daß Sie genau verstehen, wie die VIs der Analysebibliothek zur Lösung von linearen Gleichungssystemen eingesetzt werden.



1. Verwenden Sie `Untergleichungen lösen.vi` in der Unterpalette **Analysis»Lineare Algebra**, um das Gleichungssystem $Ax = b$ zu lösen, wobei die Eingangs-Matrix *A* und der bekannte Vektor *b* wie folgt lauten:

$$A = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -1 & 7 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix}$$

   Wählen Sie für den Matrixtyp den allgemeinen Matrixtyp.
2. Verwenden Sie A x Vector.vi, um die Matrix *A* und den Vektor *x* (Ergebnis der Gleichung oben) zu multiplizieren, und prüfen Sie, ob das Ergebnis mit dem Vektor *b* oben übereinstimmt.
3. Speichern Sie das VI als `Linear System.vi` im Verzeichnis `LabVIEW\Activity`.



Ende der Übung 18-2.

Matrixfaktorisierung

Im vorangehenden Abschnitt wurde erörtert, wie ein lineares Gleichungssystem in ein System umgeformt werden kann, dessen Lösung einfacher zu berechnen ist. Der grundlegende Gedanke dabei war, die Eingangs-Matrix in die Multiplikation von mehreren, einfacheren Matrizen zu faktorisieren. Sie haben eine dieser Techniken kennengelernt, die *LU- Faktorisierungsmethode*, bei der die Eingangs-Matrix als Produkt der oberen und unteren Dreiecksmatrizen faktorisiert wird. Weitere oft verwendete Faktorisierungsmethoden sind die *Cholesky-*, die *QR-* und die *Einzelwertfaktorisierungsmethoden*. Sie können diese Methoden anwenden, um eine Reihe von Matrixproblemen zu lösen, wie z.B. die Lösung von linearen Gleichungssystemen, die Umkehrung einer Matrix und die Bestimmung der Determinanten einer Matrix.

Wenn die Eingangs-Matrix *A* symmetrisch und definit positiv ist, dann kann eine LU-Faktorisierung so berechnet werden, daß $A = U^T U$, wobei *U* eine obere Dreiecksmatrix ist. Dies wird als *Cholesky-Faktorisierung* bezeichnet. Verglichen mit der LU-Faktorisierung einer allgemeinen Matrix durch Gaußsche Eliminierung erfordert diese Methode nur die Hälfte an Arbeitsaufwand und Speicherplatz. Ob eine Matrix definit positiv ist, läßt sich einfach mit dem VI Testen auf definit positive Matrix in der Analysebibliothek bestimmen.

Eine Matrix *Q* ist *orthogonal*, wenn ihre Spalten *orthonormal* sind. Das heißt, wenn $Q^TQ = I$ die Identitätsmatrix ist. Die QR-Faktorisierungsmethode faktorisiert eine Matrix als Produkt einer orthogonalen Matrix *Q* und einer oberen Dreiecksmatrix *R*. Das heißt, $A = QR$. Die QR-Faktorisierung ist sowohl bei quadratischen als auch rechteckigen Matrizen hilfreich. Eine Reihe von Algorithmen können bei der QR-Faktorisierung eingesetzt werden, wie z.B. die *Householder-Transformation*, die *Givens-Transformation* und die *Fast-Givens-Transformation*.

Die Methode der Singulärwertzerlegung zerlegt eine Matrix in das Produkt von drei Matrizen: $A = USV^T$. *U* und *V* sind orthogonale Matrizen. *S* ist eine diagonale Matrix, deren diagonale Werte als *singuläre Werte* von *A* bezeichnet werden. Die singulären Werte von *A* sind die nichtnegativen Quadratwurzeln der Eigenwerte von A^TA, und die Spalten von *U* und *V*, die linke und rechte singuläre Vektoren genannt werden, sind orthonormale Eigenvektoren von AA^T bzw. A^TA. Die Einzelwertfaktorisierung ist bei der Lösung von Analyseproblemen, wie z. B. der Berechnung von Rang, Norm, Bedingungszahl und Pseudoinversen von Matrizen, hilfreich. Im folgenden Abschnitt wird die letztere Anwendung beschrieben.

Pseudoinverse

Die Pseudoinverse eines Skalars σ ist als $1/\sigma$ definiert, wenn $\sigma \neq 0$ und andernfalls gleich Null. Im Fall von Skalaren entspricht die Pseudoinverse der Inversen. Sie können jetzt die Pseudoinverse einer diagonalen Matrix definieren, indem Sie die Matrix transponieren und anschließend die skalare Pseudoinverse jedes Elements berechnen. Die Pseudoinverse einer allgemeinen realen $m \times n$-Matrix *A*, die mit $A^\dagger$ bezeichnet ist, wird dann folgendermaßen definiert:

$$A^\dagger = VS^\dagger U^T$$

Beachten Sie, daß die Pseudoinverse unabhängig davon existiert, ob die Matrix quadratisch oder rechteckig ist. Wenn *A* quadratisch und nichtsingulär ist, dann stimmt die Pseudoinverse mit der normalen Matrixinversen überein. Die Analysebibliothek enthält ein VI zur Berechnung der Pseudoinversen von realen und komplexen Matrizen.

Zusammenfassung

- Eine Matrix kann als zweidimensionales Array von *m* Zeilen und *n* Spalten betrachtet werden. Determinante, Rang und Bedingungszahl stellen einige wichtige Attribute einer Matrix dar.
- Die Bedingungszahl einer Matrix beeinflußt die Genauigkeit der endgültigen Lösung.
- Die Determinante einer diagonalen Matrix, einer oberen oder einer unteren Dreiecksmatrix ist das Produkt ihrer diagonalen Elemente.
- Zwei Matrizen können nur dann miteinander multipliziert werden, wenn die Anzahl der Spalten der ersten Matrix mit der Anzahl der Zeilen der zweiten Matrix übereinstimmt.
- Ein Eigenvektor einer Matrix ist ein von Null abweichender Vektor, der sich nicht dreht, wenn die Matrix darauf angewendet wird. Ähnliche Matrizen haben die gleichen Eigenwerte.
- Die Existenz einer eindeutigen Lösung für ein Gleichungssystem hängt davon ab, ob die Matrix singulär oder nichtsingulär ist.

19

Wahrscheinlichkeit und Statistik

In diesem Kapitel werden einige grundlegende Konzepte der Wahrscheinlichkeit und Statistik beschrieben. Darüber hinaus wird gezeigt, wie diese Konzepte angewandt werden, um reale Probleme zu lösen. Beispiele zur Verwendung der VIs zur Wahrscheinlichkeit und Statistik entnehmen Sie der Datei `examples\analysis\statxmpl.llb`.

Wahrscheinlichkeit und Statistik

Wir leben im Informationszeitalter, in dem Fakten und Zahlen einen wichtigen Teil des Lebens ausmachen. Häufig hören wir Aussagen wie “Die Gewitterwahrscheinlichkeit liegt bei 60%”, “Johannes befand sich unter den fünf Klassenbesten.”, “Michael Jordan erzielte in dieser Saison im Durchschnitt 30 Punkte pro Spiel.”, etc. Diese Aussagen vermitteln eine Menge von Informationen, aber selten überlegen wir, wie diese Informationen ermittelt wurden. Waren dazu viele Daten erforderlich? Wenn ja, wie wurden sie zu einer einzigen Zahl wie *60% Wahrscheinlichkeit* und *durchschnittlich 30 Punkte* oder Begriffe wie *fünf Besten* zusammengefaßt? Die Antwort auf alle diese Fragen ist im äußerst interessanten Bereich der Statistik zu finden.

Überlegen Sie zunächst, wie Informationen (Daten) erzeugt werden. Betrachten Sie einmal die Basketballsaison 1997. Michael Jordan von den Chicago Bulls spielte in 51 Spielen und erzielte dabei insgesamt 1568 Punkte. Dazu gehörten die 45 Punkte, einschließlich des spielentscheidenden 3-Punkte-Wurfs, die er bei dem 103-100 Sieg über die Charlotte Hornets erzielte, seine 36 Punkte bei dem 88-84 Sieg über die Portland Trail Blazers; die höchste Anzahl der Saison von 51 Punkten bei dem 88-87 Sieg über die New York Knicks; 45 Punkte, 7 Rebounds, 5 Vorlagen und 3 Steals bei dem 102-97 Sieg über die Cleveland Cavaliers und seine 40 Punkte, 6 Rebounds und 6 Vorlagen bei dem 107-104 Sieg über die Milwaukee Bucks. Es geht hier nicht darum, daß Jordan ein großartiger Spieler ist, sondern daß ein einziger Spieler in einer Saison viele Daten erzeugen kann. Es geht noch vielmehr darum, wie diese Daten zusammengefaßt werden können, so daß alle wichtigen Informationen vermittelt werden und sie zugleich einfach zu behalten sind. Hier kommt nun die *Statistik* zum Einsatz.

Damit alle Daten zusammengefaßt werden können, müssen sie durch einzelne Zahlen verständlicher gemacht werden, die dabei helfen, nützliche Folgerungen zu ziehen. Betrachten Sie die Punktezahl, die Jordan in verschiedenen Spielen erzielte. Es ist schwierig, sich zu merken, wieviele Punkte er in jedem Spiel erzielte. Wenn Sie aber die Gesamtpunktzahl, die Jordan erzielte (1568), durch die Anzahl der Spiele teilen, in denen er eingesetzt wurde (51), erhalten Sie eine einzige Zahl, 30,7, die die *Durchschnittspunktzahl* pro Spiel angibt.

Nehmen Sie einmal an, Sie wollen Jordans Freiwurf-Fähigkeiten bewerten. Dies könnte schwierig sein, wenn Sie seine Leistung in jedem einzelnen Spiel betrachten. Sie können jedoch die Anzahl aller Freiwürfe, die in allen Spielen in Punkten resultierten, durch die Anzahl der Freiwürfe teilen, die ihm zugesprochen wurden. Daraus ergibt sich, daß er einen *Freiwurfprozentsatz* von 84 hat. Diese Zahl können Sie für alle NBA-Spieler ermitteln und diese anschließend einstufen. Sie können also die Informationen für alle Spieler in einzelne Zahlen fassen, die den Freiwurfprozentsatz, die Punkte pro Spiel und den Durchschnitt aus 3-Punkte-Würfen wiedergibt. Auf der Grundlage dieser Informationen können Sie Spieler in verschiedene Kategorien einteilen. Die verschiedenen Zahlen können dann noch weiter gewichtet und zu einer einzigen Zahl für jeden Spieler zusammengefaßt werden. Diese Einzelzahlen können dann dazu beitragen, den wertvollsten Spieler (MVP- Most Valuable Player) der Saison zu ermitteln. Im weiteren Sinne impliziert der Begriff Statistik verschiedene Arten der Datenzusammenfassung, um nützliche und wichtige Informationen daraus abzuleiten.

Die nächste Frage lautet, was ist Wahrscheinlichkeit? Sie haben soeben Methoden kennengelernt, mit denen viele Daten zu einzelnen Zahlen zusammengefaßt werden können. Diese Zahlen helfen dann dabei, Schlußfolgerungen für die Gegenwart zu ziehen. Z.B. trug Jordans Statistik aus dem Jahr 1996 dazu bei, ihn als wertvollsten Spieler der Saison zu bestimmen. Können Sie nun aber etwas über die Zukunft voraussagen? Können Sie den Genauigkeitsgrad der Folgerung messen und sie für zukünftige Entscheidungen verwenden? Die Antwort liegt in der Wahrscheinlichkeitstheorie. Während man als Laie sagen würde, daß es *wahrscheinlich* ist, daß Jordan in den nächsten Jahren weiterhin der beste Spieler sein wird, können Sie in der Wahrscheinlichkeit verschiedene Konzepte einsetzen, um quantitativere Aussagen zu machen. Diese Konzepte werden später in diesem Kapitel beschrieben.

Ein ganz anderes Beispiel sind bestimmte Versuche, deren Ergebnisse zwar nicht vorherbestimmt werden können, bei denen bestimmte Ergebnisse jedoch wahrscheinlicher sind. Hier taucht erneut der Begriff

Wahrscheinlichkeit auf. Wenn Sie z.B. eine Münze in die Luft werfen, wie groß ist dann die Chance, daß sie mit dem Kopf nach oben fällt? Die Chance bzw. die Wahrscheinlichkeit beträgt 50%. Das bedeutet, wenn Sie die Münze mehrmals in die Luft werfen, landet sie die Hälfte der Würfe mit dem Kopf nach oben. Bedeutet das, daß bei 10 Würfen genau fünfmal der Kopf nach oben zeigen wird? Wird der Kopf bei 100 Würfen genau 50 mal nach oben zeigen? Wahrscheinlich nicht, aber auf lange Sicht gesehen wird die Wahrscheinlichkeit 0,5 betragen.

Während die Statistik Ihnen ermöglicht, Daten zusammenzufassen und daraus eine Schlußfolgerung für die Gegenwart zu ziehen, ermöglicht Ihnen die Wahrscheinlichkeit, den Genauigkeitsgrad dieser Schlußfolgerungen zu messen und diese für die Zukunft zu verwenden.

Statistik

In diesem Abschnitt werden Sie verschiedene Konzepte und Begriffe kennenlernen, die normalerweise in der Statistik verwendet werden. Darüber hinaus sehen Sie, wie die G Analyse-VIs bei verschiedenen Anwendungen eingesetzt werden.

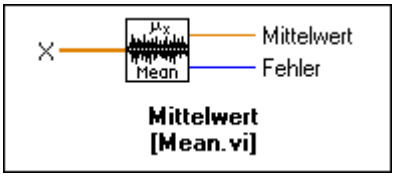
Mittelwert

Betrachten Sie einen Datensatz X, der aus n Stichproben besteht, x_0, x_1, x_2, x_3, … , x_{n-1}. Der Mittelwert (bzw. Durchschnittswert) wird durch $\bar{x}$ dargestellt und ist durch folgende Formel definiert:

$$\bar{x} = \frac{1}{n}(x_0 + x_1 + x_2 + x_3 + \ldots + x_{n-1})$$

Mit anderen Worten ist er die Summe aller Stichprobenwerte geteilt durch die Anzahl der Stichproben. Bei dem Beispiel von Michael Jordan bestand der Datensatz aus 51 Stichproben. Jede Stichprobe entsprach der Anzahl der Punkte, die Jordan in jedem Spiel erzielte. Die Gesamtsumme aller Punkte betrug 1568. Geteilt durch die Anzahl der Stichproben (51), erhält man einen Durchschnittswert von 30,7.

Die Ein- und Ausgabeverbindungen für das Mittelwert-VI sind unten dargestellt.



Zentralwert

$S = \{s_0, s_1, s_2, \ldots, s_{n-1}\}$ soll die sortierte Sequenz des Datensatzes X darstellen. Die Sequenz kann entweder in aufsteigender oder absteigender Reihenfolge sortiert werden. Der Zentralwert der Sequenz wird mit $x_{Zentralwert}$ bezeichnet und durch die folgende Formel ermittelt:

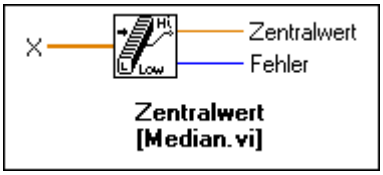
$$x_{Zentralwert} = \begin{cases} s_i & n \text{ ist ungerade} \\ 0,5(s_{k-1} + s_k) & n \text{ ist gerade} \end{cases}$$

wobei

$$i = \frac{n-1}{2} \text{ und } k = \frac{n}{2}.$$

In Worten ausgedrückt ist der Zentralwert einer Datensequenz der *Mittelpunktwert* in der sortierten Version der Sequenz. Betrachten Sie beispielsweise die Sequenz $\{5, 4, 3, 2, 1\}$, die aus fünf (ungerade) Stichproben besteht. Diese Sequenz wurde bereits in absteigender Reihenfolge sortiert. In diesem Fall ist der Zentralwert der Mittelpunktwert 3. Betrachten Sie eine weitere Sequenz, $\{1, 2, 3, 4\}$, die aus vier (gerade) Stichproben besteht. Diese Sequenz wurde bereits in aufsteigender Reihenfolge sortiert. In diesem Fall gibt es zwei Mittelpunktwerte, 2 und 3. Gemäß der oben stehenden Formel beträgt der Zentralwert $0,5 \times (2 + 3) = 2,5$. Wenn ein Student X in einem Test 4,5 Punkte und ein anderer Student Y im gleichen Test 1 Punkt erzielt, ist der Zentralwert eine nützliche Größe, um qualitative Aussagen zu machen, wie z.B. “X befindet sich in der oberen Hälfte der Klasse” oder “Y befindet sich in der unteren Hälfte der Klasse”.

Die Ein- und Ausgabeverbindungen für das **Zentralwert**-VI sind unten dargestellt.



Stichprobenvarianz

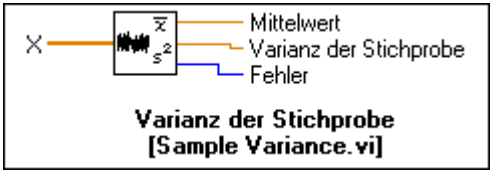
Die Stichprobenvarianz des Datensatzes *X*, der aus *n* Stichproben besteht, wird mit s^2 bezeichnet und ist durch folgende Formel definiert:

$$s^2 = \frac{1}{n-1}[(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2],$$

wobei $\bar{x}$ den Mittelwert des Datensatzes bezeichnet. Folglich ist die Stichprobenvarianz gleich der Summe der Quadrate der Abweichungen der Stichprobenwerte vom Mittelwert geteilt durch *n-1*.

Hinweis ***Die Formel oben gilt nicht für n=1. Die Berechnung der Stichprobenvarianz hat jedoch keine Aussagekraft, wenn der Datensatz nur eine Stichprobe enthält.***

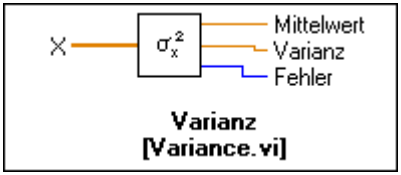
Die Ein-Ausgabeverbindungen für das **Stichprobenvarianz**-VI sind unten dargestellt.



Mit anderen Worten mißt die Stichprobenvarianz die Ausbreitung bzw. die Streuung der Stichprobenwerte.Wenn der Datensatz aus den Punkten besteht, die ein Spieler in verschiedenen Spielen erzielte, kann die Stichprobenvarianz als Maß für die Beständigkeit eines Spielers verwendet werden. Sie ist stets positiv, es sei denn, alle Stichprobenwerte stimmen miteinander und folglich mit dem Mittelwert überein.

Es gibt eine weitere Art der Varianz namens Populationsvarianz. Die Formel zur Berechnung der Populationsvarianz ähnelt der oben stehenden Formel für die Stichprobenvarianz, mit der Ausnahme, daß (*n–1*) im Nenner durch *n* ersetzt wird.

Die Ein-Ausgabeverbindungen für das **Varianz**-VI sind unten dargestellt.



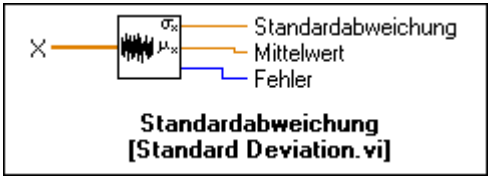
Das Stichprobenvarianz-VI berechnet die Stichprobenvarianz, während das Varianz-VI die Populationsvarianz berechnet. Statistiker und Mathematiker ziehen das letztere, Ingenieure das erstere VI vor. Für große Werte von n, wie z.B. $n \geq 30$, spielt die Wahl des VIs jedoch keine Rolle.

Hinweis ***Verwenden Sie das für Ihre Anwendung geeignete VI.***

Standardabweichung

Die positive Quadratwurzel der Stichprobenvarianz s^2 wird durch s dargestellt und mit Standardabweichung der Stichprobe bezeichnet.

Die Ein-Ausgabeverbindungen für das Standardabweichung-VI sind unten dargestellt.



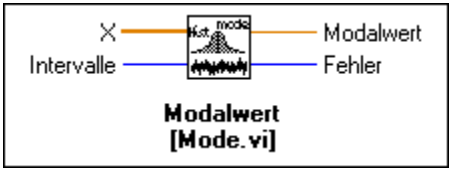
Modalwert

Der Modalwert einer Stichprobe ist der Stichprobenwert, der am häufigsten in der Stichprobe auftritt. Wenn beispielsweise die Eingabesequenz X

$$X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 7\}$$

lautet, dann ist der Modalwert von X 4, da dieser Wert am häufigsten in X vorkommt.

Die Ein-Ausgabeverbindungen für das **Modalwert**-VI sind unten dargestellt.



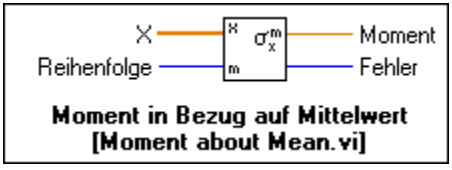
Moment in Bezug auf den Mittelwert

Wenn *X* die Eingabesequenz mit der darin enthaltenen Anzahl *n* von Elementen darstellt und $\bar{x}$ der Mittelwert dieser Sequenz ist, dann kann das Moment m^{ter} Ordnung anhand der folgenden Formel bestimmt werden:

$$\sigma_x^{\,m} = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^m$$

Das Moment in Bezug auf den Mittelwert ist also ein Maß der Abweichung der Sequenzelemente vom Mittelwert. Beachten Sie, daß bei $m = 2$ das Moment in Bezug auf den Mittelwert mit der Populationsvarianz übereinstimmt.

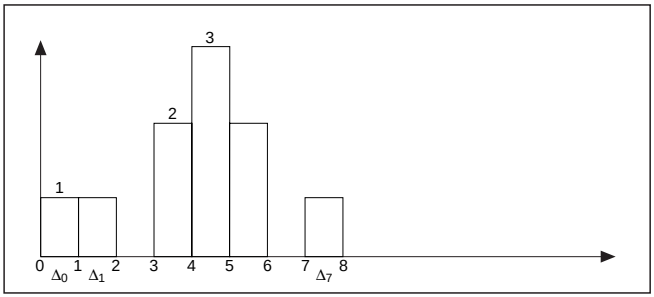
Die Ein-Ausgabeverbindungen für das VI **Moment in Bezug auf den Mittelwert** sind unten dargestellt.



Histogramm

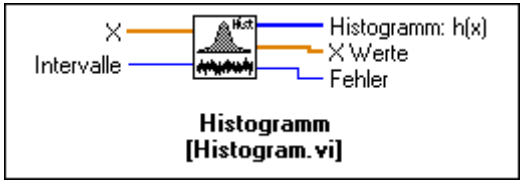
Bisher wurden in diesem Kapitel verschiedene Methoden zur Extraktion von wichtigen Eigenschaften eines Datensatzes erörtert. Die Daten werden normalerweise in Tabellenform gespeichert, was oftmals schwierig zu deuten ist. Die visuelle Darstellung von Daten trägt dazu bei, Schlüsse aus ihnen zu ziehen. Das Histogramm ist eine graphische Methode zur Darstellung von Daten und zur Zusammenfassung von wichtigen Informationen. Betrachten Sie die Datensequenz $X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 8\}$. Teilen Sie den gesamten Wertebereich in 8 Intervalle ein. Die Intervalle

sind 0 – 1, 1 – 2, 2 – 3, ..., 7 – 8. Das Histogramm der Sequenz *X* zeichnet die Anzahl der Datenstichproben auf, die in diesem Intervall liegen, wobei die obere Grenze nicht miteinbezogen wird.



Die obige Abbildung verdeutlicht, daß jeweils eine Datenstichprobe im Bereich 0 – 1 bzw. 1 – 2 liegt. Es gibt jedoch keine Stichprobe im Intervall 2 – 3. Genauso liegen zwei Stichproben im Intervall 3 – 4 sowie drei Stichproben im Bereich 4 – 5. Untersuchen Sie die Datensequenz *X* oben, und vergewissern Sie sich, daß Sie das Konzept genau verstehen.

Für die Berechnung von Daten für ein Histogramm stehen verschiedene Methoden zur Verfügung. Als nächstes sehen Sie, wie diese Berechnung im **Histogramm**-VI mit der Sequenz *X* durchgeführt wird.



Wie oben dargestellt, stellen die Eingabesequenz *X* und die Anzahl der Intervalle *m* die Eingaben in dieses VI dar. Das VI erhält **Histogramm:h(x)**, indem es *X* abtastet, um den Bereich der darin enthaltenen Werte zu bestimmen. Dann bestimmt das VI die Intervallbreite Δx gemäß dem festgelegten Wert von *m*.

$$\Delta x = \frac{max - min}{m},$$

wobei *max* für den in *X* gefundenen Höchstwert und *min* für den in *X* gefundenen Mindestwert und *m* für die festgelegte Anzahl von Intervallen steht.

Für

$$m = 8$$

gilt z.B.

$$\Delta x = \frac{8-0}{8} = 1$$

Dabei steht χ für die *X*-Werte der Ausgabesequenz. Das Histogramm ist eine Funktion von *X*. Dieses VI wertet die Elemente von χ folgendermaßen aus:

$$\chi_i = min + 0,5\Delta x + i\Delta x \qquad für \qquad i = 0, 1, 2, \ldots, m-1$$

Bei diesem Beispiel gilt

$$\chi_0 = 0,5, \chi_1 = 1,5, \ldots, \chi_7 = 7,5 .$$

Das VI definiert anschließend, daß das i^{te} Intervall innerhalb des Wertebereichs von $\chi_i(-0,5\Delta x)$ bis ausschließlich $\chi_i + 0,5\Delta x$ liegt,

$$\Delta_i = [(\chi_i - (0,5\Delta x)), (\chi_i + 0.5\Delta x)], für \qquad i = 0, 1, 2, \ldots, m-1$$

und definiert die Funktion $y_i(x) = 1$, wenn *x* zu Δ_i gehört; andernfalls ist sie Null. Die Funktion hat den Wert Eins, wenn der Wert von *x* innerhalb der festgelegten Intervalle, ausschließlich der Grenzen, liegt. Andernfalls ist sie Null. Beachten Sie, daß das Intervall um χ_i zentriert ist und seine Breite Δ_x beträgt. Wenn ein Wert dem Maximalwert entspricht, wird er zum letzten Intervall zugehörig gezählt.

Für unser Beispiel gilt

$$\Delta_0 = [0, 1], \Delta_1 = [1, 2], \ldots, \Delta_7 = [7, 8]$$

und beispielsweise

$$y_0(0) = 1$$

und

$$y_0(1) = y_0(3) = y_0(4) = y_0(5) = y_0(8) = 0 .$$

Abschließend bewertet das VI die Histogrammsequenz *H* unter Verwendung von

$$h_i = \sum_{j=0}^{n-1} y_i(x_j) \qquad für \qquad i = 0, 1, 2, \ldots, m-1$$

wobei h_i die Elemente der Ausgabesequenz *Histogramm: h(X)* und *n* die Anzahl der Elemente in der Eingabesequenz *X* darstellen. Für dieses Beispiel gilt $h_0 = 1, h_4 = 3, \ldots, h_7 = 1$.

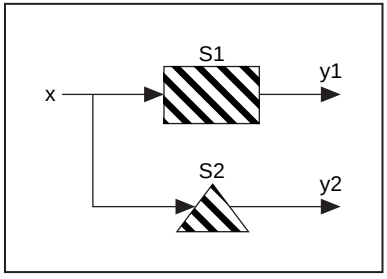
Die G Analysis-Bibliothek verfügt auch über ein Allgemeines Histogramm-VI, das fortschrittlicher als das Histogramm-VI ist. Nähere Informationen dazu entnehmen Sie bitte der *Analyse Online-Referenz.*

Mittlerer quadratischer Fehler (MSE)

Wenn X und Y zwei Eingabesequenzen darstellen, dann ist der mittlere quadratische Fehler der Durchschnitt aus der Summe des Quadrates der Differenz zwischen den entsprechenden Elementen der beiden Eingabesequenzen. Für die Ermittlung des MSE wird die folgende Formel verwendet.

$$mse = \frac{1}{n}\sum_{i=0}^{n-1} (x_i - y_i)^2$$

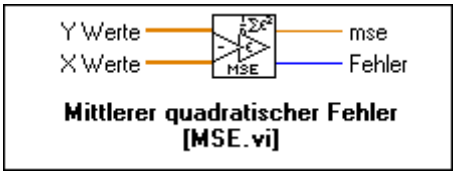
wobei *n* für die Anzahl der Datenpunkte steht.



Betrachten Sie ein digitales Signal x, das einem System S1 zugeführt wird. Der Ausgang des Systems ist y1. Jetzt ermitteln Sie ein neues System S2, das theoretisch das gleiche Ergebnis wie S1 erzeugen soll, jedoch eine doppelt so schnelle Reaktionszeit hat. Bevor Sie das alte System ersetzen, wollen Sie absolut sicher sein, daß der Ausgang beider Systeme genau

übereinstimmt. Wenn die Sequenzen y1 und y2 besonders groß sind, ist es schwierig, jedes Element der Sequenzen miteinander zu vergleichen. In diesem Fall können Sie das MSE-VI verwenden, um den mittleren quadratischen Fehler der beiden Sequenzen y1 und y2 zu berechnen. Wenn der mittlere quadratische Fehler kleiner als die akzeptable Toleranz ist, kann das System S1 zuverlässig durch das neue System S2 ersetzt werden.

Die Ein-Ausgabeverbindungen für das **MSE**-VI sind unten dargestellt.



Quadratischer Mittelwert (RMS)

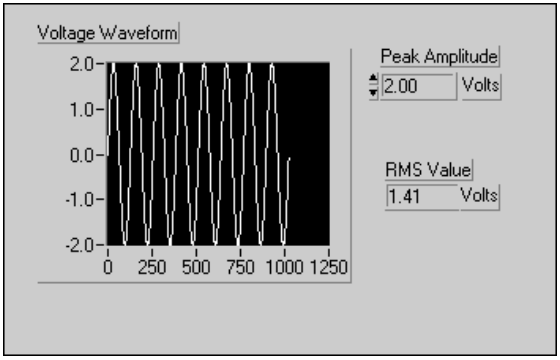
Der quadratische Mittelwert Ψ_x einer Sequenz X ist die positive Quadratwurzel des Mittelwertes des Quadrates der Eingabesequenz. Sie können also die Eingabesequenz zum Quadrat erheben, den Mittelwert dieser neuen zum Quadrat erhobenen Sequenz ermitteln und anschließend die Quadratwurzel dieser Größe bestimmen. Die Formel zur Berechnung des RMS-Wertes lautet

$$\Psi_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^{\,2}}$$

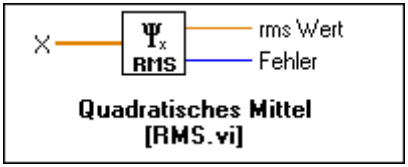
wobei *n* für die Anzahl der Elemente in *X* steht.

Der quadratische Mittelwert ist eine häufig verwendete Größe bei Analogsignalen. Wenn bei einer Spannungs-Sinuskurve V_p die Spitzenamplitude des Signals darstellt, dann wird der quadratische Mittelwert der Spannung V_{rms} durch $\frac{V_p}{\sqrt{2}}$ definiert.

In der folgenden Abbildung wird eine Spannungskurve mit der Spitzenamplitude = 2 V und dem RMS-Wert von $\sqrt{2} \approx 1,41$ V mit Hilfe der Analyse-Bibliothek berechnet.



Die Ein-Ausgabeverbindungen für das **RMS**-VI sind unten dargestellt.



Wahrscheinlichkeit

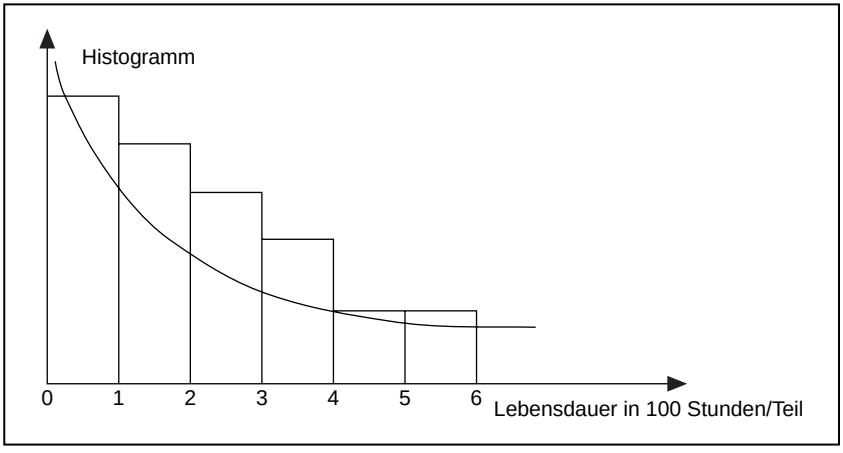
Bei jedem Zufallsexperiment gibt es immer die Möglichkeit, daß ein bestimmtes Ereignis eintreten bzw. nicht eintreten wird. Eine Zahl zwischen 0 und 1 wird bestimmt, um diese Möglichkeit bzw. Wahrscheinlichkeit, daß ein bestimmtes Ereignis eintritt, zu messen. Wenn Sie vollkommen sicher sind, daß das Ereignis eintreten wird, beträgt seine Wahrscheinlichkeit 100% oder 1,0. Sind Sie sich sicher, daß das Ereignis nicht eintreten wird, ist seine Wahrscheinlichkeit 0.

Betrachten Sie ein einfaches Beispiel. Wenn Sie einen einzigen, unverzerrten Würfel rollen, können sechs mögliche Ereignisse eintreten. Das Ergebnis kann 1, 2, 3, 4, 5 oder 6 lauten. Mit welcher Wahrscheinlichkeit wird das Ergebnis 2 lauten? Die Wahrscheinlichkeit beträgt eins zu sechs oder 0,16666. Die Wahrscheinlichkeit kann in einfachen Worten folgendermaßen beschrieben werden: Die Wahrscheinlichkeit für das Eintreten eines Ereignisses A ist das Verhältnis der Anzahl der für A positiven Ereignisse zur Gesamtanzahl der ebenso möglichen Ereignisse.

Zufallsvariablen

Viele Experimente haben Ergebnisse, die in reale Zahlen ausgedrückt interpretiert werden können. Beispiele sind die Anzahl von Fahrzeugen, die täglich an einem Stopp-Zeichen vorbeifahren, die Anzahl der Wähler, die Kandidat A bevorzugen und die Anzahl der Unfälle an einer bestimmten Kreuzung. Die Werte der numerischen Ergebnisse dieses Experiments können von Experiment zu Experiment verschieden ausfallen und werden Zufallsvariablen genannt. Zufallsvariablen können diskret (wenn sie nur eine endliche Anzahl von möglichen Werten annehmen können) oder kontinuierlich sein. Ein Beispiel für eine kontinuierliche Zufallsvariable: Das Gewicht von Patienten, die in ein Krankenhaus eingeliefert werden, kann z.B. irgendwo zwischen 80 und 300 Pfund liegen. Solche Zufallsvariablen können in einem Intervall von realen Zahlen irgendeinen Wert annehmen. Nehmen Sie einmal an, daß Sie bei dem vorgegebenen Beispiel die Wahrscheinlichkeit ermitteln wollen, einen Patienten zu finden, der genau 172,39 Pfund wiegt. Anhand eines Beispiels werden Sie sehen, wie diese Wahrscheinlichkeit berechnet werden kann.

Betrachten Sie z.B. ein Experiment, in dem die Lebensdauer *x* von 50 Batterien eines bestimmten Typs gemessen werden soll. Diese Batterien werden aus einer größeren Population von Batterien ausgewählt. Das Histogramm für die beobachteten Daten ist unten dargestellt.



Diese Abbildung verdeutlicht, daß die Lebensdauer der meisten Batterien zwischen Null und 100 Stunden liegt, und daß die Histogrammwerte bei längeren Lebensdauern gleichmäßig abflachen.

Sie können das oben dargestellte Histogramm durch eine exponentiell abfallende Kurve nähern. Sie könnten diese Funktion als mathematisches Modell für das Verhalten der Datenstichprobe betrachten. Wenn Sie die Wahrscheinlichkeit bestimmen wollen, daß eine willkürlich gewählte Batterie eine Lebensdauer von mehr als vierhundert Stunden hat, kann dieser Wert durch den Bereich unterhalb der Kurve rechts vom Wert 4 genähert werden. Solch eine Funktion, die das Histogramm einer Zufallsvariablen modelliert, wird mit *Wahrscheinlichkeitsdichtefunktion* bezeichnet.

Die vorausgegangenen Informationen können folgendermaßen zu einer Definition zusammengefaßt werden: eine Zufallsvariable *X* wird mit *kontinuierlich* bezeichnet, wenn sie eine unendliche Anzahl von möglichen Werten annehmen kann, die mit den Intervallen von realen Zahlen assoziiert werden, und wenn es eine Funktion *f(x)* gibt, die mit *Wahrscheinlichkeitsdichtefunktion* bezeichnet wird, so daß folgendes gilt:

1. $f(x) \geq 0$ *für alle x*

2. $\int_{-\infty}^{\infty} f(x)dx = 1$

3. $P(a \leq X \leq b) = \int_{a}^{b} f(x)dx$

Beachten Sie bei Gleichung (3) oben, daß für den bestimmten Wert der kontinuierlichen Zufallsvariablen folgendes gilt:

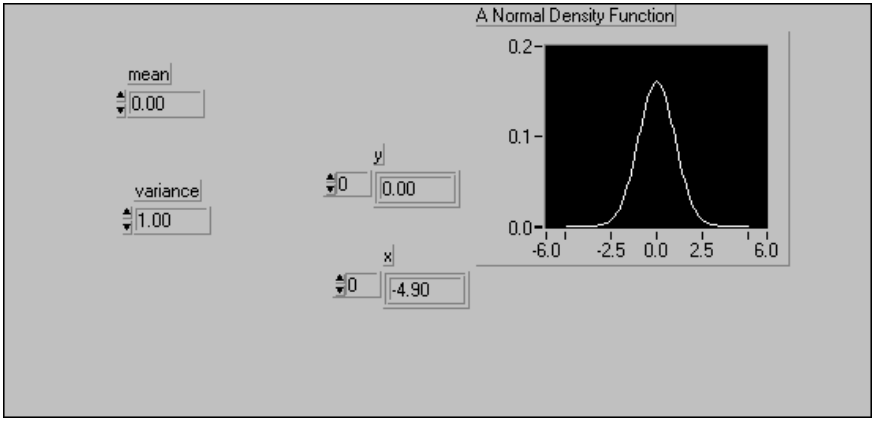
$$X=a,\ P(X = a) = \int_{a}^{a} f(x)dx = 0\ .$$

Es sollte nicht überraschen, daß Sie irgendeinem bestimmten Wert eine Wahrscheinlichkeit von Null zuweisen, da es für die Zufallsvariable eine unendliche Anzahl von möglichen Werten gibt. Folglich ist die Wahrscheinlichkeit, daß sie einen bestimmten Wert $X = a$ annehmen wird, äußerst gering.

Das vorangehende Beispiel verwendet das Exponentialfunktionsmodell für die Wahrscheinlichkeitsdichtefunktion. Für diese Funktion gibt es eine Reihe von verschiedenen Möglichkeiten. Eine davon ist die Normalverteilung, die im folgenden beschrieben wird.

Normalverteilung

Die Normalverteilung ist eine der am häufigsten eingesetzten kontinuierlichen Wahrscheinlichkeitsverteilungen. Diese Verteilungsfunktion hat die Form einer symmetrischen Glocke, wie in der folgenden Abbildung dargestellt.



Der Mittelpunkt der Kurve liegt beim Mittelwert $\bar{x} = 0$, und ihre Ausdehnung wird durch die Varianz $s^2 = 1$ gemessen. Diese zwei Parameter bestimmen die genaue Form und Position der Normaldichtefunktion, deren Funktionsform wie folgt definiert ist:

$$f(x) = \frac{1}{\sqrt{2\pi}s} e^{-(x-\bar{x})^2/(2s^2)}$$

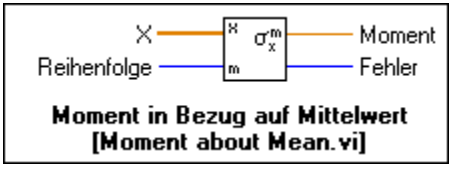
Nehmen Sie einmal an, daß eine Zufallsvariable Z eine Normalverteilung hat, deren Mittelwert Null und deren Varianz Eins entspricht. Diese Zufallsvariable hat dann eine *Standardnormalverteilung*.

Das G Analysis Normalverteilung-VI berechnet die einseitige Wahrscheinlichkeit *p* der normalverteilten Zufallsvariablen *x*

$$p = Prob(X \leq x)$$

wobei *X* die Standardnormalverteilung mit einem Mittelwert gleich Null und einer Varianz gleich Eins, *p* die Wahrscheinlichkeit und *x* der Wert ist.

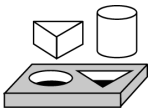
Nehmen Sie an, daß Sie ein Experiment durchführen, bei dem Sie die Größe von erwachsenen Männern messen. Sie führen diesen Versuch an 1000 willkürlich ausgewählten Männern aus und erhalten den Datensatz S. Die Histogrammverteilung zeigt viele Messungen, die dicht zusammen um eine Mittelwertgröße liegen, mit relativ wenigen sehr kleinen und sehr großen Männern in der Population. Deshalb kann das Histogramm durch eine Normalverteilung dicht genähert werden. Nehmen Sie anschließend an, daß Sie bei einem anderen Satz von 1000 willkürlich ausgewählten Männern die Wahrscheinlichkeit ermitteln wollen, daß die Größe eines Mannes größer oder gleich 170 cm ist. Um diese Wahrscheinlichkeit zu ermitteln, können Sie das Normalverteilung-VI verwenden. Legen Sie als Eingabe $x = 170$ fest. Die Wahl der Wahrscheinlichkeitsdichtefunktion ist also für den Erhalt eines richtigen Wahrscheinlichkeitswerts von größter Bedeutung.



Das Inverse Normalverteilung-VI führt genau die entgegengesetzte Funktion aus. Bei einer gegebenen Wahrscheinlichkeit *p* ermittelt es die Werte *x*, die der Wahrscheinlichkeit nach in einer normalverteilten Stichprobe enthalten sind. Sie wollen z.B. die Größen feststellen, die mit 60%iger Wahrscheinlichkeit in einem willkürlich gewählten Datensatz enthalten sind.

Wie bereits erwähnt, gibt es verschiedene Möglichkeiten für die Wahrscheinlichkeitsdichtefunktion. Die bekanntesten und am häufigsten angewandten sind die Chi-Quadrat-Verteilung, die F-Verteilung und die T-Verteilung. Weitere Informationen über diese Verteilungen können Sie Anhang A, *Analyse-Referenzen* entnehmen. Die G-Analyse-Bibliothek enthält VIs, die die einseitige Wahrscheinlichkeit für diese verschiedenen

Verteilungstypen berechnen. Darüber hinaus enthält sie auch VIs, die die umgekehrte Operation durchführen.



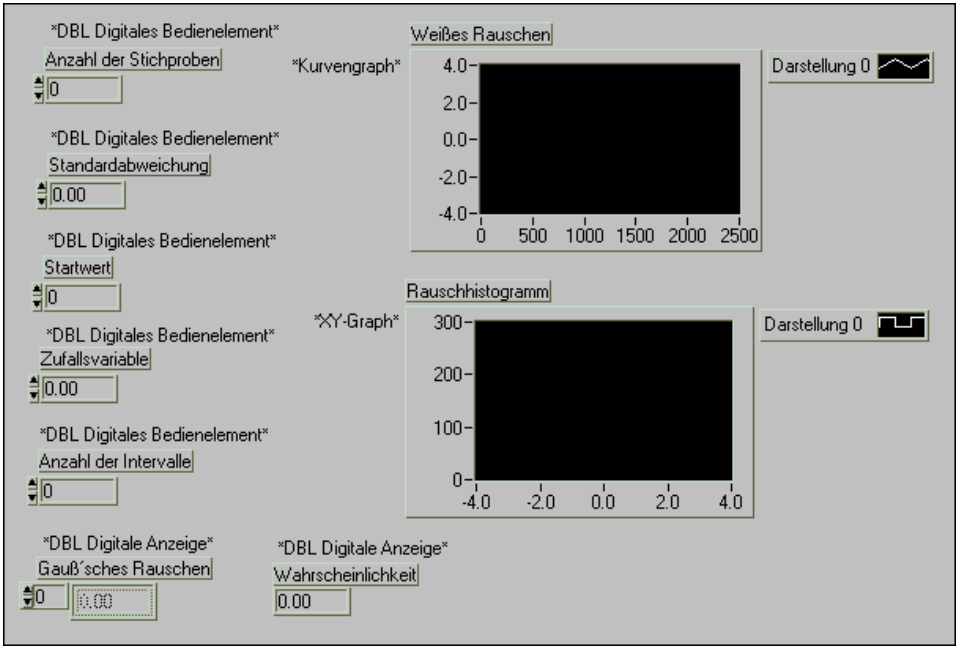
Übung 19-1. Das Normalverteilung-VI verwenden

Das Übungsziel ist, die wichtigsten Wahrscheinlichkeitskonzepte zu verstehen.

Bei dieser Übung werden Sie zunächst eine Datenstichprobe mit Standardnormalverteilung erstellen und dann die Wahrscheinlichkeit einer Zufallsvariablen *x* mit dem Normalverteilung-VI überprüfen.

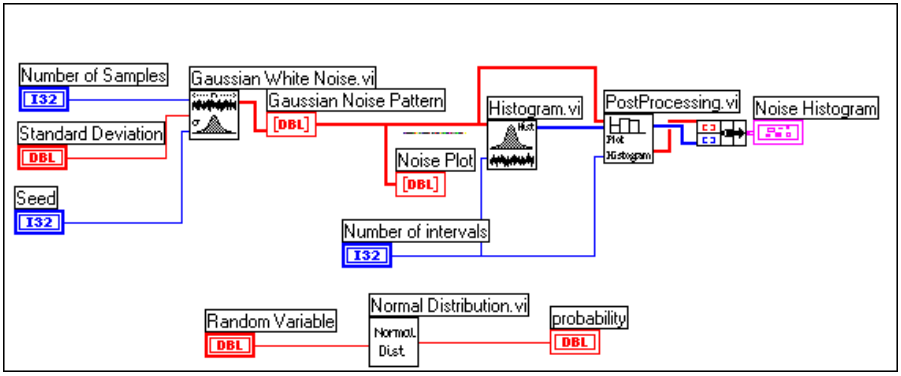
Frontpanel

1. Erstellen Sie das Frontpanel wie in der folgenden Abbildung dargestellt. Weißes Rauschen ist ein Kurvengraph, während Rauschhistogramm ein XY-Graph ist.



Blockdiagramm

2. Erstellen Sie das Blockdiagramm wie in der folgenden Abbildung dargestellt. Das Gaußsche weiße Rauschen erzeugt ein nach Gauß verteiltes Muster mit einem Mittelwert von Null und einer Standardabweichung, die vom Anwender mit Hilfe der Eingabestandardabweichung gesetzt wird. Stichproben ist die Anzahl der Stichproben des Gaußschen Rauschens. Startwert ist der Ausgangswert, der für die Erzeugung des Zufallsrauschens verwendet wird. Verbinden Sie das Gaußsche Rauschen mit dem Kurvengraphen Weißes Rauschen.



Gaußsches weißes Rauschen (Unterpalette **Analysis»Signalerzeugung**). Bei dieser Übung erzeugt diese Funktion ein Gaußsches weißes Rauschen-Muster.

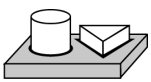
Histogramm (Unterpalette **Analysis»Wahrscheinlichkeit und Statistik**). Bei dieser Übung berechnet diese Funktion das Histogramm des Gaußschen Rauschens.

Normalverteilung (Unterpalette **Analysis»Wahrscheinlichkeit und Statistik**). Bei dieser Übung berechnet diese Funktion die einseitige Wahrscheinlichkeit der normalverteilten Zufallsvariablen **Zufallsvariable**.

3. Berechnen Sie das Histogramm des Gaußschen Rauschens mit dem Histogramm-VI, das in der vorangegangenen Übung verwendet wurde.
4. Wie bereits erörtert, sollten Sie eine Nachverarbeitung durchführen, um das Histogramm auf eine andere Art zu zeichnen. Wählen Sie das Nachverarbeitungs-VI aus dem Verzeichnis `LabVIEW\Activity`.
5. Bündeln Sie den Ausgang dieses VIs, und verbinden Sie ihn mit dem Rauschen-Histogramm.
6. Wählen Sie das Normalverteilung-VI. Verbinden Sie das Bedienelement Zufallsvariable mit dem Eingangsanschluß, und verbinden Sie den Ausgang mit dem Anzeigeelement Wahrscheinlichkeit.
7. Kehren Sie zum Frontpanel zurück. Setzen Sie die Anzahl der Stichproben auf 2,048, die Standardabweichung auf 1, Startwert auf 2 und Anzahl der Intervalle auf 10. Führen Sie das VI aus.
8. Sie sehen das Gaußsche weiße Rauschen im Weißes Rauschen-Graphen. Es ist schwierig, aus diesem Graphen Schlüsse zu ziehen. Die Histogrammkurve des gleichen Rauschens bietet jedoch eine Fülle von Informationen. Sie zeigt, daß die meisten Stichproben um den Mittelwert Null zentriert sind. Sie können dieses Rauschen von diesem Histogramm durch eine Normalverteilungsfunktion (Gaußsche Verteilung) nähern. Da der Mittelwert Null beträgt und Sie die Standardabweichung auf Eins festlegen, stellt die Wahrscheinlichkeitsdichtefunktion tatsächlich eine Standardnormalverteilung dar.

Hinweis ***Es ist sehr wichtig, daß Sie den geeigneten Verteilungsfunktionstyp wählen, um Ihre Daten zu nähern. Bei diesem Beispiel haben Sie das Histogramm tatsächlich eingezeichnet, um eine Entscheidung zu treffen. Auf der Basis von Kenntnissen über das Verhalten und die Charakteristiken einer Datenstichprobe können Sie oft eine kluge Entscheidung treffen.***

9. Kehren Sie zum Frontpanel zurück, und geben Sie einen Wert für die Zufallsvariable ein. Dieses VI berechnet die einseitige Wahrscheinlichkeit dieser normalverteilten Zufallsvariablen. Bedenken Sie, daß Sie nach einem Blick auf das Histogramm angenommen haben, daß die Variable normalverteilt ist.
10. Speichern Sie das VI unter `Probability.vi` im Verzeichnis `LabVIEW\Activity`.



Ende der Übung 19-1.

Zusammenfassung

- Verschiedene Konzepte in der Statistik und Wahrscheinlichkeit helfen dabei, Informationen und Daten zu entschlüsseln, um kluge Entscheidungen treffen zu können.
- Mittelwert, Zentralwert, Stichprobenvarianz und Modalwert sind einige der Statistikmethoden, die Ihnen helfen, aus einer Stichprobe Rückschlüsse über eine Population zu ziehen.
- Histogramme werden häufig als einfache aber informative Methode der Datendarstellung eingesetzt.
- Mit Hilfe der Wahrscheinlichkeitstheorie können Sie aus einer Stichprobe Rückschlüsse über eine Population ziehen und anschließend den Genauigkeitsgrad dieser Rückschlüsse messen.

Teil IV

Kommunikation im Netzwerk und zwischen den Anwendungen

Dieser Teil des Handbuchs enthält grundsätzliche Informationen zur Kommunikation im Netzwerk und zwischen den Anwendungen.

Teil IV, *Kommunikation im Netzwerk und zwischen den Anwendungen*, enthält die folgenden Kapitel:

- Kapitel 20, *Einführung in die Kommunikation*, bietet eine Einführung in die Art und Weise, wie LabVIEW die Kommunikation im Netzwerk und zwischen den Anwendungen bewältigt.
- Kapitel 21, *TCP und UDP*, beschreibt die Protokolle Internet Protocol (IP), User Datagram Protocol (UPD) und Transmission Control Protocol (TCP) sowie Internet-Adressen und Beispiele für TCP- Client/Server-Anwendungen.
- Kapitel 22, *ActiveX-Unterstützung*, erläutert, wie LabVIEW als ActiveX-Server und -Client verwendet wird. ActiveX entspricht der OLE Automation-Kommunikation.
- Kapitel 23, *DDE verwenden*, beschreibt die LabVIEW-VIs für den dynamischen Datenaustausch DDE (Dynamic Data Exchange) für Windows 3.1, Windows 95 und Windows NT. Diese VIs führen DDE-Funktionen zur Freigabe von Daten für andere Anwendungen aus, die DDE-Verbindungen akzeptieren.
- Kapitel 24, *AppleEvents*, beschreibt AppleEvents, eine Form der IAC (Interapplication Communication)-Kommunikation, die nur für den Macintosh geeignet ist und die die Kommunikation zwischen Anwendungen auf dem Macintosh ermöglicht.
- Kapitel 25, *Programm-zu-Programm-Kommunikation*, beschreibt die Programm-zu-Programm-Kommunikation (PPC/Program-to-Program Communication), eine Form der Apple IAC-Kommunikation auf niedriger Stufe, die von Anwendungen auf dem Macintosh zum Senden und Empfangen von Datenblöcken verwendet wird.

20

Einführung in die Kommunikation

Dieses Kapitel enthält eine Einführung in die Art und Weise, mit der LabVIEW Datenübertragungen im Netzwerk und zwischen verschiedenen Anwendungen handhabt.

Überblick über die Kommunikation mit LabVIEW

In diesem Kapitel bezieht sich der Begriff *Networking* auf die Kommunikation zwischen verschiedenen Prozessen, die auf unterschiedlichen Rechnern ablaufen können. Diese Art der Kommunikation geschieht normalerweise über ein Hardware-Netzwerk, wie z.B. Ethernet oder LocalTalk.

Networking wird in Software-Anwendungen vor allem eingesetzt, wenn eine oder mehrere Anwendungen die Dienste einer anderen Anwendung verwenden sollen. Beispielsweise kann die Anwendung, welche die Dienste zur Verfügung stellt (der Server), entweder eine Datenerfassungsanwendung auf einem dafür bestimmten Rechner oder ein Datenbankprogramm sein, das anderen Anwendungen Informationen liefert.

In diesem Kapitel werden Sie die Terminologie des Networking und der Kommunikation sowie die Programmierung von vernetzten Anwendungen kennenlernen.

Einführung in Kommunikationsprotokolle

Damit eine Kommunikation zwischen Prozessen stattfinden kann, müssen diese eine gemeinsame Kommunikationssprache verwenden, die *Protokoll* genannt wird.

In einem Kommunikationsprotokoll können Sie Daten, die Sie senden oder empfangen möchten, sowie den Ziel- oder Quellort festlegen, ohne daß Sie sich Gedanken darüber machen müssen, wie die Daten dorthin gelangen. Das Protokoll übersetzt Ihre Befehle in Daten, die von Netzwerktreibern

gelesen werden können. Die Netzwerktreiber sorgen anschließend für die jeweils geeignete Übertragung der Daten im Netz.

Mittlerweile gibt es verschiedene Netzwerkprotokolle, die als Standardprotokolle für die Datenübertragung akzeptiert werden. Im allgemeinen sind Protokolle nicht miteinander kompatibel, und folglich müssen Sie vor dem Einsatz von Kommunikationsanwendungen zunächst entscheiden, welches Protokoll Sie verwenden wollen. Wenn Sie mit einer bereits bestehenden Anwendung kommunizieren möchten, müssen Sie die von dieser Anwendung unterstützten Protokolle verwenden.

Wenn Sie die Anwendung jedoch selbst schreiben, haben Sie bei der Wahl des Protokolls eine größere Flexibilität. Die Faktoren, von denen Ihre Protokollwahl abhängt, sind z.B. die Rechnertypen, auf denen die Prozesse ablaufen werden, die Art des verfügbaren Hardware-Netzwerkes sowie die Komplexität der Datenübertragung, die für Ihre Anwendung erforderlich ist.

LabVIEW verfügt über verschiedene bereits integrierte Protokolle, von denen manche speziell für eine Rechnerart bestimmt sind. LabVIEW verwendet die folgenden Protokolle für die Datenübertragung zwischen Rechnern:

- **TCP**—Auf allen Rechnern verfügbar
- **UDP**—Auf allen Rechnern verfügbar
- **DDE**—Auf PCs zur Kommunikation zwischen Windows-Anwendungen verfügbar
- **ActiveX**—Für den Einsatz mit Windows 95 und Windows NT
- **AppleEvents**—Auf Macintosh zum Senden von Informationen zwischen Macintosh-Anwendungen verfügbar
- **PPC**—Auf Macintosh zum Senden und Empfangen von Informationen zwischen Macintosh-Anwendungen verfügbar.

Jedes Protokoll ist anders, vor allem in der Art, wie es die Netzwerkadresse einer Fernanwendung anspricht. Die Protokolle sind nicht miteinander kompatibel. Wenn Sie also Daten zwischen einem Macintosh und einem PC übertragen wollen, müssen Sie ein Protokoll wie z.B. TCP verwenden, das mit beiden Rechnerarten kompatibel ist.

Weitere Kommunikationsoptionen in LabVIEW sind z.B.:

- **Systembefehl ausführen-VI**—Ermöglicht Ihnen, einen Befehl auf Systemebene auszuführen. Es gibt zwei Systembefehl ausführen-VIs, eins für alle Versionen von Windows und ein anderes für Sun und HP-UX.
- **Pipes**—Nur bei UNIX
- **HiQ**—Nur bei Macintosh und PC

Gemeinsame Dateinutzung contra Kommunikationsprotokolle

Bevor Sie sich näher mit Kommunikationsprotokollen befassen, überlegen Sie, ob eine andere Vorgehensweise für Ihre Anwendung nicht geeigneter wäre. Betrachten Sie beispielsweise eine Anwendung, bei der ein dafür bestimmtes System Daten erfaßt, die Sie auf einem anderen Rechner aufzeichnen wollen.

Sie könnten eine Anwendung schreiben, die Netzwerkprotokolle verwendet, um die Daten vom Erfassungsrechner an den Aufzeichnungsrechner zu übertragen, wo eine andere Anwendung die Daten erfaßt und auf der Festplatte speichert.

Es könnte einfacher sein, die Fähigkeiten zur gemeinsamen Dateinutzung zu verwenden, über die die meisten vernetzten Rechner verfügen. Bei der gemeinsamen Dateinutzung können Sie mit Hilfe von Treibern, die Teil des Betriebssystems sind, auf andere Rechner zugreifen. Der Festplattenspeicher des Fernrechners wird als Erweiterung der eigenen Festplatte behandelt. Nachdem zwei Systeme miteinander verbunden sind, wird diese Verbindung durch die gemeinsame Dateinutzung transparent, so daß jede Anwendung auf die Fernfestplatte schreiben kann, als ob es die rechnereigene wäre. Die gemeinsame Dateinutzung ist häufig die einfachste Methode der Datenübertragung zwischen Rechnern.

Client/Server-Modell

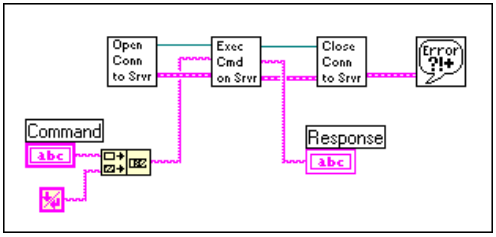
Das Client/Server-Modell ist ein gängiges Modell für vernetzte Anwendungen. Bei diesem Modell fordert ein Prozeßsatz (Client) Dienste von einem anderen Prozeßsatz (Server) an.

Sie könnten in Ihrer Anwendung beispielsweise einen bestimmten Rechner einrichten, um praktische Messungen zu erfassen. Der Rechner agiert als Server, wenn er Daten auf Abfrage an andere Rechner liefert. Er agiert als Client, wenn er eine andere Anwendung, wie z.B. ein Datenbankprogramm, auffordert, die von ihm erfaßten Daten aufzuzeichnen.

In LabVIEW können Sie Client-/Server-Anwendungen mit allen Protokollen außer Macintosh AppleEvents einsetzen. AppleEvents wird verwendet, um Befehle an andere Anwendungen zu senden. Sie können mit AppleEvents keinen Befehlsserver in LabVIEW einrichten. Wenn Sie Server-Fähigkeiten auf einem Macintosh-Rechner benötigen, müssen Sie TCP, UDP oder PPC verwenden.

Allgemeines Client-Modell

Das folgende Blockdiagramm zeigt, wie ein vereinfachtes Modell für einen Client in LabVIEW aussieht.



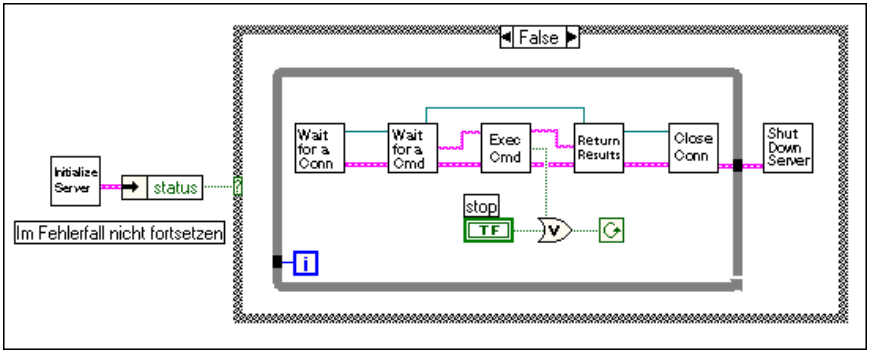
Im vorhergehenden Diagramm öffnet LabVIEW zunächst eine Verbindung zu einem Server. Anschließend sendet es einen Befehl an den Server, erhält eine Antwort von diesem und schließt dann die Verbindung wieder. Abschließend meldet es alle Fehler, die während der Datenübertragung aufgetreten sind.

Um eine bessere Leistung zu erzielen, können Sie, nachdem die Verbindung geöffnet wurde, mehrere Befehle gleichzeitig verarbeiten. Nachdem alle Befehle ausgeführt sind, können Sie die Verbindung wieder schließen.

Diese grundlegende Blockdiagramm-Struktur dient als Modell und wird an anderer Stelle in diesem Handbuch verwendet, um zu zeigen, wie ein gegebenes Protokoll in LabVIEW implementiert wird.

Allgemeines Server-Modell

Das folgende Blockdiagramm zeigt ein vereinfachtes Modell für einen Server in LabVIEW.

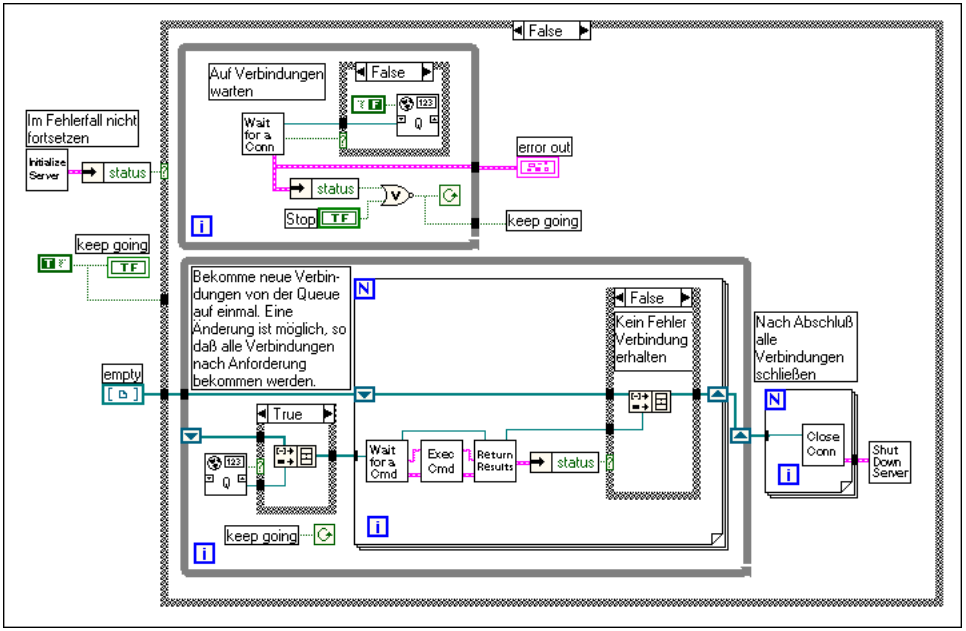


Im vorhergehenden Diagramm wird der Server zunächst von LabVIEW initialisiert. Bei einer erfolgreichen Initialisierung geht LabVIEW in eine Schleife und wartet auf eine Verbindung. Nachdem die Verbindung steht, wartet LabVIEW auf einen Befehl. LabVIEW führt den Befehl aus und gibt die Ergebnisse zurück. Die Verbindung wird anschließend geschlossen. LabVIEW wiederholt den gesamten Vorgang, bis es entweder lokal beendet wird, nachdem ein Anwender eine Stopp-Taste auf dem Frontpanel drückt, oder entfernt, wenn es einen Befehl erhält, das VI zu beenden.

Dieses VI meldet keine Fehler. Es kann eine Antwort zurücksenden, die besagt, daß der Befehl ungültig ist, aber es zeigt kein Dialogfeld an, wenn ein Fehler auftritt. Da ein Server unbeaufsichtigt ablaufen kann, sollten Sie genau überlegen, wie der Server mit Fehlern umgehen sollte. Es empfiehlt sich, kein Dialogfeld anzuzeigen, da dies eine Interaktion mit dem Anwender am Server erfordert. (Jemand müßte auf die Taste OK drücken). Vielleicht wollen Sie jedoch, daß LabVIEW ein Protokoll aller Transaktionen und Fehler in einer Datei oder einem String wiedergibt.

Sie können die Leistung steigern, indem Sie dafür sorgen, daß die Verbindung offen bleibt, so daß Sie mehrere Befehle erhalten können. Dieser Vorgang blockiert jedoch die Verbindung für andere Clients, bis der gegenwärtig verbundene Client von ihr getrennt wird. Wenn das Protokoll mehrere Verbindungen gleichzeitig unterstützt, können Sie LabVIEW

neustrukturieren, so daß es mehrere Clients gleichzeitig handhaben kann, wie im folgenden Diagramm dargestellt.



Das vorangehende Diagramm verwendet LabVIEW Multitasking-Fähigkeiten, um zwei Schleifen gleichzeitig auszuführen. Eine Schleife wartet fortwährend auf eine Verbindung. Nach Erhalt einer Verbindung wird sie zur Warteschleife hinzugefügt. Die andere Schleife prüft alle offenen Verbindungen und führt alle empfangenen Befehle aus. Wenn ein Fehler bei einer der Verbindungen auftritt, wird diese getrennt. Wenn der Anwender den Server beendet, werden alle offenen Verbindungen geschlossen. Diese grundlegende Blockdiagramm-Struktur dient als Modell und wird an anderer Stelle in diesem Handbuch verwendet, um zu zeigen, wie ein gegebenes Protokoll in LabVIEW implementiert wird.

21

TCP und UDP

In diesem Kapitel werden das Internet Protocol (IP), das User Datagram Protocol (UPD), das Transmission Control Protocol (TCP), Internetadressen und Beispiele für TCP-Client-/Server-Anwendungen beschrieben.

Überblick

TCP/IP ist eine Folge von Kommunikationsprotokollen, die ursprünglich für die amerikanische Defense Advanced Research Projects Agency (DARPA) entwickelt wurde. Seitdem ist TCP/IP zu einem weitgehend akzeptierten Standard geworden und steht auf einer Reihe von Rechnersystemen zur Verfügung.

Der Name TCP/IP stammt von den zwei am besten bekannten Protokollen der Folge, dem Transmission Control Protocol (Übertragungssteuerungsprotokoll) (TCP) und dem Internet Protocol (IP). TCP, IP und das User Datagram Protocol (Anwenderdatagrammprotokoll) (UDP) sind grundlegende Werkzeuge für die Netzwerkkommunikation.

TCP/IP ermöglicht die Datenübertragung über Einzelnetze oder mehrere miteinander verbundene Netzwerke (Internet). Die einzelnen Netzwerke können durch große geographische Entfernungen voneinander getrennt sein. TCP/IP leitet Daten von einem Netzwerk oder Internet-Rechner zu einem anderen. Da TCP/IP auf den meisten Rechnern zur Verfügung steht, ist es in der Lage, Informationen zwischen verschiedenen Systemen zu übertragen.

Das Internet Protocol (IP) überträgt Daten durch das Netzwerk. Dieses Low-Level-Protokoll sendet Daten von begrenzter Größe als *Datagramm* durch das Netzwerk. Da es nicht gewährleistet, daß die Daten am anderen Ende ankommen, wird IP selten direkt von Anwendungen verwendet. Wenn Sie mehrere Datagramme versenden, kommen sie außerdem manchmal in der falschen Reihenfolge an oder werden mehrfach übertragen, je nachdem, wie die Netzwerkübertragung abläuft. UDP, das auf IP aufbaut, weist ähnliche Probleme auf.

TCP ist ein Protokoll höherer Ebene, das IP für die Datenübertragung einsetzt. TCP teilt die Daten in Komponenten auf, die von IP verarbeitet werden können. Es bietet zudem eine Fehlererkennung und gewährleistet, daß die Daten in der richtigen Reihenfolge und ohne Duplikate ankommen. Aus diesem Grund ist TCP normalerweise die beste Wahl für Netzwerkanwendungen.

LabVIEW und TCP/IP

Sie können die TCP/IP-Protokollfolge auf allen Plattformen mit LabVIEW einsetzen. LabVIEW verfügt über einen Satz von TCP- und UDP-VIs, den Sie verwenden können, um Client- oder Server-VIs zu erstellen.

Internetadressen

Jeder Host in einem IP-Netzwerk hat eine eindeutige 32-Bit Internetadresse. Diese Adresse identifiziert das Netzwerk im Internet, an das der Host angeschlossen ist, und den bestimmten Rechner in diesem Netzwerk. Sie verwenden diese Adresse, um den Absender oder Empfänger der Daten zu identifizieren. IP plaziert die Adresse in die Kopfzeile des Datagramms, damit jedes Datagramm richtig geleitet wird.

Eine Methode, um diese 32-Bit-Adresse zu beschreiben, ist die IP-Dezimalpunktdarstellung, bei der die 32-Bit-Adresse in vier 8-Bit-Zahlen eingeteilt wird. Die Adresse wird in Form der vier Integer geschrieben, die durch Dezimalpunkte getrennt sind. Die folgende 32-Bit-Adresse wird in der Dezimalpunktdarstellung mit `132.13.2.30` umschrieben:

```
10000100     00001101     00000010     00011110
```

Die 32-Bit-Adresse kann außerdem in Form von Namen verwendet werden, die der IP-Adresse zugewiesen sind. Netzwerktreiber führen diese Zuweisung normalerweise durch, indem Sie entweder auf eine lokale *Hostdatei* zugreifen, die den Namen zu den Adressenzuweisungen enthält, oder auf eine größere Domain-Name-System-Datenbank, um andere Rechnersysteme nach der Adresse eines bestimmten Namens abzufragen. Ihre Netzwerkkonfiguration bestimmt den genauen Ablauf dieses Vorgangs, der auch mit *Hostnamenauflösung* bezeichnet wird.

Internet Protocol (IP)

Das Internet Protocol (IP) führt den Low-Level-Dienst der Übertragung von Daten zwischen Systemen aus. IP verpackt Daten in Komponenten namens *Datagramme*. Ein Datagramm enthält u. a. die Daten sowie eine Kopfzeile mit der Quelle und der Zieladresse. IP bestimmt den korrekten Pfad, den das Datagramm durch das Netzwerk oder das Internet nehmen muß und sendet die Daten an das angegebene Ziel.

Es ist möglich, daß der ursprüngliche Host nicht den vollständigen Datenpfad kennt. Mit Hilfe der Kopfzeile kann jeder Host im Netz die Daten an das Ziel leiten. Dies geschieht entweder direkt oder indem er sie an einen anderen Host weiterleitet. Da einige Systeme unterschiedliche Übertragungsfähigkeiten haben, kann IP Datagramme, falls erforderlich, in kleinere Segmente zerlegen. Nachdem die Daten am Ziel angekommen sind, werden sie vom IP automatisch wieder zu ihrem ursprünglichen Format zusammengesetzt.

Das IP versucht nach Möglichkeit, die Daten am Ziel abzuliefern, kann dies jedoch nicht garantieren. Die Datagramme können in der falschen Reihenfolge ankommen, da jedes einzeln vom IP geleitet wird. Es kann sogar vorkommen, daß das IP ein Einzelpaket mehrmals abliefert, wenn dieses während der Übertragung dupliziert wurde. Das IP bestimmt nicht die Reihenfolge der Pakete. Diese wird von Protokollen bestimmt, die sich auf einer höheren Ebene über dem IP befinden und die auch für eine zuverlässige Lieferung sorgen. Das IP wird selten direkt verwendet, da die Programme stattdessen TCP oder UDP verwenden.

User Datagram Protocol (UDP)

Das UDP sorgt für eine einfache Low-Level-Kommunikation zwischen den Prozessen, die auf Rechnern ablaufen. Prozesse kommunizieren miteinander, indem sie Datagramme an einen Zielrechner oder -anschluß senden. Das IP handhabt die Lieferung von Rechner zu Rechner. Nachdem die Daten in einem Rechner angekommen sind, werden sie vom UDP an den richtigen Zielanschluß geleitet. Wenn der Zielanschluß nicht mit einem Empfangsprozeß verbunden ist, wird das Datagramm verworfen. Das UDP weist die gleichen Lieferungsprobleme wie das IP auf.

Normalerweise wird UDP in Anwendungen verwendet, bei denen es nicht auf die Zuverlässigkeit ankommt. Beispielsweise kann eine Anwendung häufig informative Daten an ein Ziel übertragen, so daß verlorene Datensegmente nicht weiter tragisch sind.

Verwenden von UDP

UDP ist kein verbindungsbasiertes Protokoll, wie z.B. TCP. Das bedeutet, daß es vor dem Senden oder Empfangen von Daten nicht erforderlich ist, eine Verbindung mit dem Ziel herzustellen. Stattdessen wird das Ziel für die Daten bestimmt, wenn jedes einzelne Datagramm gesendet wird. Das System meldet keine Übertragungsfehler.

Sie können das UDP öffnen-VI zum Öffnen eines Anschlusses verwenden. Ein Anschluß ist der Ort, an den die Daten gesendet werden. Die Anzahl der gleichzeitig offenen UDP-Anschlüsse hängt vom System ab. UDP öffnen gibt eine Netzwerkverbindung-Refnum zurück, ein Token, das in allen nachfolgenden mit dem Anschluß in Zusammenhang stehenden Operationen verwendet wird.

Sie können das UDP schreiben-VI verwenden, um Daten an ein Ziel zu senden und das UDP lesen-VI, um sie zu lesen. Jeder Schreibvorgang erfordert eine Zieladresse und einen Zielanschluß. Jeder Lesevorgang umfaßt die Quelladresse und den Quellanschluß. Paketbereichsgrenzen werden bewahrt. Das heißt, ein Lesevorgang enthält niemals Daten, die in zwei separaten Schreibvorgängen gesendet wurden.

Theoretisch sollten Sie in der Lage sein, Pakete von jeder Größe zu senden. Falls erforderlich, wird ein Paket in kleinere Teile zerlegt gesendet. Am Ziel angelangt, werden die Teile zusammengesetzt, und das Paket wird an den Abfrageprozeß übergeben. In der Praxis reservieren Rechnersysteme jedoch nur eine bestimmte Speichergröße für die Zusammensetzung eines Pakets. Ein Paket, das nicht wieder zusammengesetzt werden kann, wird verworfen. Die maximale Größe eines Pakets, das ohne Zerlegung gesendet werden kann, hängt von Ihrer Netzwerk-Hardware ab.

Nachdem LabVIEW alle Datenübertragungen beendet hat, können Sie Systemressourcen freigeben, indem Sie das UDP schließen-VI aufrufen.

Transmission Control Protocol (TCP)

Das Transmission Control Protocol (TCP) gewährleistet eine zuverlässige Übertragung von Daten in der richtigen Reihenfolge, ohne Fehler, Verluste oder Duplizierung im Netz. Wenn Sie Daten an das TCP übergeben, hängt es zusätzliche Informationen an und übergibt die Daten anschließend an das IP. Das IP überträgt die Daten in Form eines Datagramms an den Empfänger. Am Zielort läuft dieser Vorgang in umgekehrter Reihenfolge ab, wobei das TCP prüft, ob die Daten fehlerlos und in der richtigen Reihenfolge empfangen wurden und anschließend die erfolgreiche

Übertragung meldet. Wenn das Sender-TCP keine Rückmeldung erhält, überträgt es das Datensegment erneut.

Verwenden von TCP

TCP ist ein verbindungsbasiertes Protokoll. Das bedeutet, daß vor dem Senden und Empfangen von Daten eine Verbindung hergestellt werden muß. TCP ermöglicht mehrere Verbindungen gleichzeitig.

Die Verbindung wird entweder durch Warten auf eine eingehende Verbindung oder durch eine aktive Herstellung einer Verbindung mit einer bestimmten Adresse initiiert. Beim Aufbau von TCP-Verbindungen müssen Sie sowohl die Adresse als auch einen Anschluß an dieser Adresse angeben. Ein Anschluß wird von einer Zahl zwischen 0 und 65535 dargestellt. Bei UNIX-Systemen sind Anschlußnummern unter 1024 für besondere Anwendungen reserviert. Mehrere Anschlüsse an der Adresse kennzeichnen verschiedene Dienste unter dieser Adresse und vereinfachen den Umgang mit mehreren gleichzeitigen Verbindungen.

Sie können eine Verbindung mit einem bestimmten Anschluß und einer bestimmten Adresse aktiv herstellen, indem Sie die Funktion TCP Verbindung öffnen verwenden. Mit dieser Funktion geben Sie die Adresse und den Anschluß an, mit dem Sie kommunizieren wollen. Ist die Verbindung erfolgreich, gibt die Funktion eine Verbindungs-ID zurück, welche die Verbindung eindeutig bezeichnet. Verwenden Sie diese Verbindungs-ID, um die Verbindung in nachfolgenden Funktionsaufrufen anzusprechen.

Es gibt zwei Methoden, um auf eine eingehende Verbindung zu warten:

- Bei der ersten Methode verwenden Sie das TCP hören-VI, um einen Listener zu erstellen und auf eine angenommene TCP-Verbindung an einem festgelegten Anschluß zu warten. Wenn die Verbindung erfolgreich ist, gibt das VI eine Verbindungs-ID sowie die Adresse und den Anschluß des Remote-TCP zurück.
- Bei der zweiten Methode verwenden Sie die Funktion TCP Listener erstellen, um einen Listener zu erstellen, und verwenden dann die Funktion Auf Listener warten, um auf neue Verbindungen zu warten und diese anzunehmen. Auf Listener warten gibt dieselbe Listener-ID zurück, die an die Funktion übergeben wurde, sowie die Verbindungs-ID für eine Verbindung. Wenn Sie nicht mehr auf neue Verbindungen warten, können Sie TCP schließen verwenden, um den Listener zu schließen. Daten von einem Listener können weder gelesen noch zu diesem geschrieben werden.

Der Vorteil der zweiten Methode liegt darin, daß Sie eine Listen-Funktion durch Aufrufen von TCP schließen abbrechen können. Dies ist nützlich, wenn Sie ohne die Verwendung einer Zeitabschaltung auf eine Verbindung hören möchten, aber die Hören-Funktion abbrechen wollen, wenn eine andere Bedingung eintritt (z.B., wenn der Anwender eine Taste drückt).

Nachdem eine Verbindung hergestellt ist, können Sie Daten mit den Funktionen TCP lesen und TCP schreiben zur Fernanwendung schreiben oder von dieser lesen.

Verwenden Sie abschließend die Funktion TCP Verbindung schließen, um die Verbindung zur Fernanwendung zu schließen. Wenn es noch ungelesene Daten gibt und die Verbindung geschlossen wird, können diese Daten verloren gehen. Die miteinander verbundenen Parteien sollten ein Protokoll höherer Ebene verwenden, damit sie bestimmen können, wann eine Verbindung geschlossen werden kann. Nachdem eine Verbindung geschlossen wurde, können Sie nicht mehr von ihr lesen oder zu ihr schreiben.

TCP contra UDP

Wenn Sie sowohl den Client als auch den Server entwickeln und ihr System TCP/IP verwenden kann, ist TCP wahrscheinlich das beste Protokoll, da es zuverlässig und verbindungsbasiert ist. UDP ist ein verbindungsloses Protokoll mit einer höheren Leistungsstärke. Es gewährleistet jedoch keine zuverlässige Datenübertragung.

Beispiel für einen TCP-Client

Dieser Abschnitt enthält eine allgemeine Beschreibung zur Verwendung der Komponenten des Client-Blockdiagrammodells mit dem TCP-Protokoll.

Verwenden Sie die Funktion TCP Verbindung öffnen, um eine Verbindung zu einem Server zu öffnen. Sie müssen die Internetadresse des Servers sowie den *Anschluß* für den Server angeben. Die Adresse identifiziert einen Rechner im Netz. Der Anschluß ist eine zusätzliche Nummer, die für einen Kommunikationskanal des Rechners steht, den der Server verwendet, um auf Datenübertragungsanfragen zu warten. Wenn Sie einen TCP-Server erstellen, müssen Sie den Anschluß festlegen, den der Server als Kommunikationsanschluß verwenden soll.

Zum Ausführen eines Befehls auf dem Server verwenden Sie die Funktion TCP schreiben, um den Befehl an den Server zu senden. Sie setzen dann die Funktion TCP lesen ein, um die Ergebnisse vom Server zurückzulesen. Bei der Funktion TCP lesen müssen Sie die Anzahl der Zeichen festlegen, die Sie lesen wollen. Da die Länge der Antwort variieren kann, ist dies nicht immer günstig. Das gleiche Problem kann für den Server beim Befehl auftreten, da auch seine Länge variieren kann.

Mit den folgenden Methoden können Sie Befehle von verschiedener Länge adressieren:

- Setzen Sie einen Parameter von fester Länge vor den Befehl oder das Ergebnis, der die Größe des Befehls bzw. Ergebnisses bestimmt. In diesem Fall lesen Sie den Größenparameter und anschließend die Anzahl der durch die Größe festgelegten Zeichen. Diese Option ist effizient und flexibel.
- Legen Sie für jeden Befehl und jedes Ergebnis eine feste Größe fest. Wenn ein Befehl kleiner als die festgelegte Größe ist, kann er bis zur festgelegten Größe aufgefüllt werden.
- Setzen Sie nach jedem Befehl und jedem Ergebnis ein bestimmtes Abschlußzeichen. Die Daten müssen in diesem Fall in kleineren Segmenten bis zum Abschlußzeichen gelesen werden.

Verwenden Sie die Funktion TCP Verbindung schließen, um die Verbindung zum Server zu schließen.

Zeitbegrenzungen und Fehler

Im vorangehenden Abschnitt wurden die Kommunikationsprotokolle für den Server behandelt. Beim Entwurf einer Netzanwendung sollten Sie genau überlegen, was bei einem Fehler geschehen soll. Wie sollen die Client-VIs z.B. mit dem Absturz eines Servers umgehen?

Sie könnten z.B. dafür sorgen, daß jedes VI eine Zeitbegrenzung hat. Wenn dann ein Ergebnis aufgrund von Fehlern nicht eintritt, fährt der Client nach einer bestimmten Zeitspanne mit der Ausführung fort. Dabei kann er versuchen, die Ausführung zu wiederholen oder eine Fehlermeldung liefern. Falls erforderlich, kann er auch einfach die Client-Anwendung beenden.

Beispiel für einen TCP-Server

In diesem Abschnitt wird beschrieben, wie Sie das TCP einsetzen können, um jede Komponente des allgemeinen Servermodells zu erfüllen.

Bei TCP ist keine Initialisierung erforderlich, und Sie können diesen Schritt auslassen.

Verwenden Sie das TCP hören-VI, um auf eine Verbindung zu hören. Sie müssen den Anschluß bestimmen, der für die Datenübertragung verwendet werden soll. Dieser Anschluß muß mit dem Anschluß übereinstimmen, mit dem der Client eine Verbindung aufzubauen versucht. Weitere Informationen entnehmen Sie bitte dem Abschnitt *Beispiel für einen TCP-Client* in diesem Kapitel.

Nachdem eine Verbindung aufgebaut wurde, lesen Sie von diesem Anschluß, um einen Befehl wiederzugewinnen. Wie im TCP-Client-Beispiel beschrieben, müssen Sie das Format der Befehle festlegen. Wenn den Befehlen ein Längenfeld vorangestellt ist, lesen Sie zunächst das Längenfeld und anschließend die Datenanzahl, die durch das Längenfeld festgelegt ist.

Die Ausführung eines Befehls sollte unabhängig vom Protokoll ablaufen, da sie auf dem lokalen Rechner durchgeführt wird. Nachdem der Vorgang abgeschossen ist, werden die Daten an die nächste Stufe übergeben, von wo sie an den Client übertragen werden.

Verwenden Sie die Funktion TCP schreiben, um Ergebnisse zurückzugeben. Wie im Abschnitt *Beispiel für einen TCP-Client* beschrieben, müssen die Daten ein Format haben, das vom Client akzeptiert wird.

Verwenden Sie die Funktion TCP Verbindung schließen, um die Verbindung zu schließen.

Dieser Schritt kann bei TCP ausgelassen werden, da nach dem Schließen der Verbindung alles beendet ist.

TCP-Server mit Mehrfachverbindungen

TCP handhabt problemlos Mehrfachverbindungen. Sie können die im vorangehenden Abschnitt beschriebenen Methoden einsetzen, um die

Komponenten eines Servers mit Mehrfachverbindungen zu implementieren.

Einstellungen

Vor dem Einsatz von TCP/IP müssen Sie sicherstellen, daß Sie die richtigen Einstellungen vornehmen, die je nach verwendetem Rechner verschieden sind.

UNIX

Eine TCP/IP-Unterstützung ist bereits integriert. Wenn Ihr Netzwerk ordnungsgemäß konfiguriert wurde, sind keine zusätzlichen Einstellungen für LabVIEW notwendig.

Macintosh

TCP/IP ist bereits im Macintosh-Betriebssystem 7.5 und höher integriert. Damit Sie TCP/IP mit früheren Systemen verwenden können, müssen Sie die MacTCP-Treibersoftware installieren, die von der Apple Programmer Developer Association (APDA) erhältlich ist. (Wenden Sie sich dazu an European Developer Relations / Apple Computer, Inc. über die Rufnummer 089-99640 0.) Sie erhalten dort Informationen zur Lizenzgebung des MacTCP-Treibers. LabVIEW arbeitet auch mit Open Transport.

Windows 3.*x*

Für die Verwendung von TCP/IP müssen Sie eine Ethernetkarte mit dem dazugehörigen Low-Level-Treiber installieren. Darüber hinaus müssen Sie TCP/IP-Software kaufen und installieren, die eine Windows Sockets (WinSocks) DLL enthält, die dem Standard 1.1 entspricht. WinSock ist eine Standardschnittstelle, die die Kommunikation von Anwendungen mit einer Reihe von Netzwerktreibern ermöglicht. Netzwerksoftware, die WinSock DLL enthält, wird von verschiedenen Herstellern angeboten. Installieren Sie die Ethernetkarte, die Kartentreiber und die WinSock DLL gemäß den Anweisungen des Software-Herstellers.

Verschiedene Hersteller liefern WinSock-Treiber für den Einsatz mit einer Reihe von Karten. Wenden Sie sich an den Hersteller Ihrer Karte, um herauszufinden, ob er eine WinSock DLL anbietet, die Sie damit verwenden können. Installieren Sie die WinSock DLL gemäß den Anweisungen des Software-Herstellers.

National Instruments empfiehlt, die WinSock DLL zu verwenden, die im Lieferumfang von Windows für Arbeitsgruppen von Microsoft enthalten ist. Vor der Freigabe dieser WinSock DLL hat National Instruments eine Reihe von DLLs getestet. Zum Zeitpunkt der Tests entsprachen viele DLLs nicht vollkommen dem Standard. Aus diesem Grund empfiehlt es sich, vor dem Kauf der kompletten Version eine Demo-Version auszuprobieren. Normalerweise sind Demo-Versionen vom Hersteller erhältlich. Die meisten Demo-Versionen sind voll funktionsfähig, laufen jedoch nach einer bestimmten Zeitspanne aus.

Windows 95 und Windows NT

Eine TCP-Unterstützung ist bereits in Windows 95 und NT integriert. Es ist nicht notwendig, eine DLL von einem anderen Hersteller zu verwenden, um mit Hilfe von TCP kommunizieren zu können.

22

ActiveX-Unterstützung

In diesem Kapitel wird beschrieben, wie LabVIEW als ActiveX-Server und -Client eingesetzt wird.

Mit der ActiveX-Automation können Sie auf Eigenschaften und Methoden einer Windows-Anwendung zugreifen, die normalerweise in Objekten gruppiert sind und diese in einer anderen Windows-Anwendung einsetzen. LabVIEW für Windows 3.*x* unterstützt keine ActiveX-Automation. ActiveX-Automation wird nur von Windows 95 und NT unterstützt.

Eine Anwendung unterstützt ActiveX-Automation entweder als Server oder als Client. Anwendungen, die Objekte exponieren und Methoden zur Änderung dieser Objekte liefern, stellen ActiveX-Automation-Server dar. Anwendungen, welche die von einer anderen Anwendung exponierten Methoden verwenden, repräsentieren ActiveX-Automation-Clients.

LabVIEW kann sowohl als ActiveX-Server als auch als ActiveX-Client arbeiten. Es ist darüber hinaus in der Lage, ein ActiveX-Objekt mit Hilfe des ActiveX-Containers auf dem Frontpanel darzustellen. Weitere Informationen über ActiveX entnehmen Sie bitte Kapitel 16, *ActiveX-Bedienelemente*, im *Referenzhandbuch zur Programmierung in G*.

☞ **Hinweis** ***In diesem Dokument bezieht sich der Begriff ActiveX auf die ActiveX-Technologie der Microsoft Corporation sowie auf die OLE-Technologie.***

Allgemeine Informationen zu ActiveX entnehmen Sie bitte der Dokumentation des Microsoft Developers Network und *Inside OLE*, ***zweite Ausgabe von Kraig Brockschmidt.***

Funktionalität des ActiveX-Automation-Servers

Da Sie LabVIEW als ActiveX-Automation-Server einsetzen können, können andere ActiveX-fähige Anwendungen (wie Microsoft Excel) Eigenschaften und Methoden von LabVIEW und einzelnen VIs abfragen.

Zum Aktivieren von LabVIEW als ActiveX-Server wählen Sie **Bearbeiten»Voreinstellungen»Server: Konfiguration**. Das folgende Dialogfeld wird angezeigt:

Abbildung 22-1. Dialogfeld Voreinstellungen, Serverkonfiguration

Wählen Sie das Protokoll **ActiveX**. LabVIEW exportiert eine erstellbare Klasse namens Anwendung und eine Dispatch-Klasse namens Virtuelles Instrument an ActiveX. Die progId `LabVIEW.Application` oder `LabVIEW.Application.5` erstellt ein Anwendungsobjekt. Die Anwendungsmethode GetVIReference erstellt einen Zeiger und gibt diesen an das Objekt Virtuelles Instrument zurück.

Eigenschaften und Methoden von ActiveX-Servern

Siehe LabVIEW *Online-Referenz* für detaillierte Beschreibungen von Eigenschaften und Methoden der exportierten Klassen.

Ein Beispiel zur Verwendung von Eigenschaften und Methoden finden Sie im Verzeichnis `examples\comm` in `\freqresp.xls`. Dieses Beispiel verwendet ein Visual Basic Skriptmakro, um ein VI auszuführen und die Ergebnisse tabellarisch darzustellen.

Funktionalität des ActiveX-Automation-Client

LabVIEW kann als ActiveX-Automation-Client agieren und andere ActiveX-Automation-Server steuern. LabVIEW kann Eigenschaften einstellen und lesen und Methoden ausführen, die von ActiveX-Servern zur Verfügung gestellt werden. Ein Server exportiert Informationen über seine Objekte, Methoden und Eigenschaften mit Hilfe einer Typenbibliothek-Datei. Eine Typenbibliothek wird normalerweise von der Umgebung erstellt, in der die Server erstellt wurden. Weitere Informationen entnehmen Sie bitte der Dokumentation der jeweiligen Serveranwendung.

Die Tabelle 22-1 enthält und beschreibt die Funktionen, die Sie für einen ActiveX-Automation-Client verwenden können.

Tabelle 22-1. Funktionen zur Unterstützung von ActiveX-Automation-Clients

Funktion	Beschreibung
Automation öffnen	Wählt eine zu öffnende Automation-Klasse
Methodenknoten	Führt Funktionen einer Klasse aus
Eigenschaftsknoten	Stellt die Eigenschaften einer Klasse ein oder liest diese
Automation schließen	Schließt eine Automation-Refnum

Mit den folgenden Schritten können Sie eine Client-Anwendung in C erstellen:

1. Holen Sie das IDispatch-Interface des Objekts, auf dessen Methoden Sie zugreifen wollen.
2. Holen Sie die DispatchID der Objektmethode.
3. Rufen Sie die Methode mit Hilfe der Aufruffunktionen des IDispatch-Interface auf, und tragen Sie alle Parameter in die Parameterliste ein.

Zum Erstellen einer Client-Anwendung in LabVIEW führen Sie die folgenden Schritte aus:

1. Verwenden Sie die Funktion Automation öffnen, um eine Automation-Refnum zu erhalten, die das IDispatch-Interface eindeutig beschreibt.
2. Verwenden Sie den Methodenknoten, um eine zum Objekt gehörende Methode auszuführen. LabVIEW konvertiert seine Datentypen in eine ActiveX-Varianz, falls die Server-Anwendung Daten in diesem Format voraussetzt.

Die Beispiele im nächsten Abschnitt, *Beispiele für ActiveX-Clients*, zeigen die Verwendung dieser Knoten.

Beispiele für ActiveX-Clients

Die folgenden Beispiele zeigen, wie die neuen, oben aufgezählten ActiveX-Funktionen verwendet werden.

Konvertieren vom Datentyp ActiveX in G-Daten

Das erste, in Abbildung 22-2 dargestellte Beispiel zeigt, wie die ActiveX-Daten in G-Daten konvertiert werden. Bei jeder ActiveX-Anwendung müssen Sie zu Beginn eine ActiveX-Automation-Refnum öffnen und diese am Ende wieder schließen. Bei diesem Beispiel öffnen Sie das Anwendungsobjekt von Microsoft Excel und stellen die sichtbare Eigenschaft mit der Funktion Eigenschaftsknoten dar. Die sichtbare Eigenschaft hat das ActiveX-Datenformat. Die G-Datenfunktion muß die Eigenschaftsinformation in ein Format konvertieren, das von LabVIEW unterstützt wird.

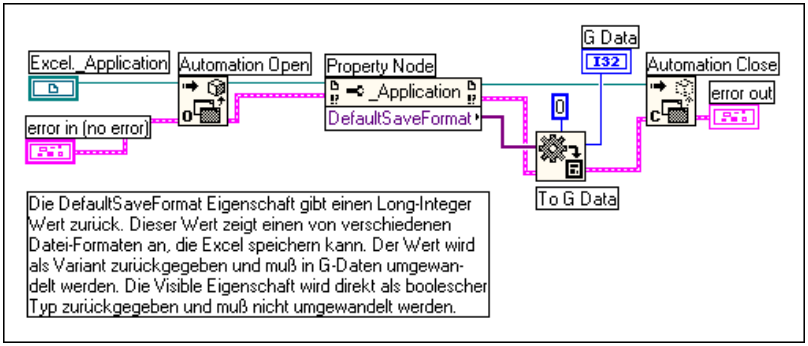


Abbildung 22-2. Blockdiagramm zur Darstellung der Konvertierung vom ActiveX-Datentyp in G-Daten

Weitere Informationen über die Eigenschaftsinformation anderer Anwendungen entnehmen Sie bitte der Online-Hilfe der jeweiligen Anwendung. Die *Online-Referenz* von LabVIEW enthält keine Eigenschaftsinformationen für andere ActiveX-fähige Anwendungen.

Hinzufügen eines Workbook in Microsoft Excel von LabVIEW aus

Das zweite in Abbildung 22-3 dargestellte Beispiel fügt von LabVIEW aus ein Workbook in Microsoft Excel hinzu. Wie bereits zuvor in diesem Abschnitt erwähnt, müssen Sie jede ActiveX-Anwendungs-Refnum mit der Funktion Automation Refnum öffnen öffnen und die ActiveX-Anwendung mit der Funktion Automation Refnum schließen wieder schließen. Zum Hinzufügen eines weiteren Workbook benötigen Sie eine Refnum zum Workbook. Bei diesem Beispiel öffnen Sie die Excel Refnum mit Automation Refnum öffnen und greifen mit dem Eigenschaftsknoten auf die Workbook-Refnum zu. Nachdem Sie ein Workbook in Excel hinzugefügt haben, wird eine Refnum mit Bezug zu diesem Workbook an LabVIEW zurückgegeben. Wenn Excel nicht länger geöffnet bleiben muß, schließen Sie die Excel und Workbook Refnums.

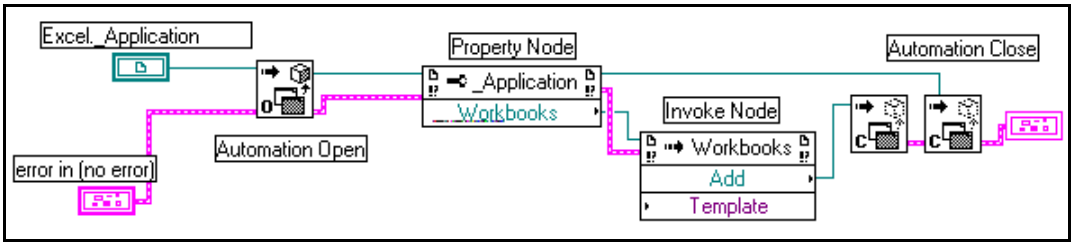


Abbildung 22-3. Hinzufügen eines Workbook in Microsoft Excel

23

DDE verwenden

In diesem Kapitel werden die LabVIEW-VIs für den dynamischen Datenaustausch DDE (Dynamic Data Exchange) in Windows 3.1, Windows 95 und Windows NT beschrieben. Diese VIs führen DDE-Funktionen zur gemeinsamen Nutzung von Daten durch andere Anwendungen aus, die DDE-Verbindungen akzeptieren.

DDE-Überblick

Der dynamische Datenaustausch oder "Dynamic Data Exchange" (DDE) ist ein Protokoll für den Austausch von Daten zwischen Windows-Anwendungen.

Bei der TCP/IP-Kommunikation öffnen die Anwendungen eine Kommunikationsverbindung und übertragen dann Rohdaten. DDE arbeitet dagegen auf einer höheren Ebene, auf der die Anwendungen einander Nachrichten senden, um Informationen auszutauschen. Durch eine einfache Nachricht kann z.B. der Auftrag erteilt werden, einen Befehl an eine andere Anwendung zu senden. Die meisten anderen Nachrichten haben mit der Übertragung von Daten zu tun, wobei auf die Daten durch Namen verwiesen wird.

Beide Anwendungen müssen laufen, und beide müssen Windows ihre Adresse für die Rückruffunktion mitteilen, bevor die DDE-Kommunikation beginnen kann. Die Rückruffunktion akzeptiert alle DDE-Nachrichten, die Windows an die Anwendung sendet.

Ein DDE-Client leitet den Datenaustausch mit einer anderen Anwendung (einem DDE-Server) durch Senden einer Verbindungsnachricht ein. Nachdem eine Verbindung hergestellt wurde, kann der Client Befehle an den Server senden und Daten ändern oder den Wert von Daten abrufen, die vom Server verwaltet werden.

Ein Client kann Daten direkt anfordern oder um wiederholte Benachrichtigung (DDE Advise) bitten. Durch eine Datenanforderung ermittelt der Client den aktuellen Wert der Daten. Wenn ein Wert über einen bestimmten Zeitraum überwacht werden soll, muß der Client darum bitten, daß Änderungen mitgeteilt werden. Durch diese Aufforderung zur

wiederholten Benachrichtigung (d.h. um “Advise”) über die Datenwerte stellt der Client eine Verbindung zum Server her, über die der Server den Client informiert, wenn sich die Daten ändern. Wenn die Überwachung der Daten beendet werden soll, kann der Client dem Server mitteilen, daß die Advise-Verbindung abgebrochen werden soll.

Wenn der Datenaustausch einer DDE-Kommunikation vollständig durchgeführt wurde, sendet der Client eine Nachricht zum Schließen der Verbindung an den Server.

DDE eignet sich am besten für die Kommunikation mit frei erhältlichen Standardanwendungen wie z.B. Microsoft Excel.

Mit LabVIEW können Sie VIs erstellen, die als Clients für andere Anwendungen eingesetzt werden können (d.h., sie fordern Daten von anderen Anwendungen an oder senden Daten an andere Anwendungen). Sie können auch VIs erstellen, die als Server dienen und genau bezeichnete Informationen für den Zugriff durch andere Anwendungen bereithalten. Beim Einsatz als Server führt LabVIEW die Kommunikation nicht auf *Verbindungsbasis* durch. Stattdessen geben Sie genau bezeichnete Informationen für andere Anwendungen an, die die in diesen Informationen enthaltenen Werte anhand ihrer Bezeichnung lesen oder einstellen können.

Services, Themen und Datenelemente

Bei Verwendung von TCP/IP identifizieren Sie den Prozeß, mit dem Sie kommunizieren möchten, durch seine Computeradresse und eine Portnummer. Im DDE-Protokoll geben Sie dagegen die Anwendung, mit der Sie kommunizieren möchten, durch den Namen eines Service und ein Thema an. Der Server legt beliebige Service- und Themennamen fest. Ein Server verwendet normalerweise (jedoch nicht in jedem Fall) den Namen seiner Anwendung für den Service. Ein solcher Server kann mehrere Themen anbieten, für die die Kommunikation möglich ist. Bei Excel kann als Thema z.B. der Name einer Kalkulationstabelle angegeben werden.

Wenn Sie mit einem Server kommunizieren wollen, sollten Sie zuerst die Namen für den Service und das Thema ermitteln, über die Sie kommunizieren möchten. Beginnen Sie dann mit dem Datenaustausch, indem Sie diese beiden Namen zur Identifikation des Servers verwenden.

Abgesehen von Fällen, in denen Sie einen Befehl an den Server senden, arbeiten Sie normalerweise mit Datenelementen, zu deren Bearbeitung der Server bereit ist. Sie können diese Datenelemente als eine Liste der *Variablen* behandeln, deren Manipulation der Server Ihnen erlaubt. Sie können den Namen einer Variablen ändern, indem Sie der Variablen einen neuen Wert geben. Ebenso können Sie auch die Werte der Variablen über deren Namen abrufen.

Beispiele für die Client-Kommunikation mit Excel

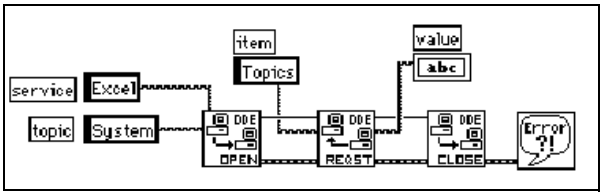
Jede Anwendung, die DDE unterstützt, verfügt über unterschiedliche Dienste, Themen und Datenelemente, über die Informationen ausgetauscht werden können. Zwei verschiedene Tabellenkalkulationsprogramme können z.B. vollkommen unterschiedliche Methoden zur Kennzeichnung von Tabellenzellen verwenden. Im Dokumentationsmaterial, das mit der Anwendung geliefert wird, finden Sie ausführliche Informationen über die von der jeweiligen Anwendung unterstützten Funktionen.

Microsoft Excel, ein weit verbreitetes Tabellenkalkulationsprogramm für Windows, bietet DDE-Unterstützung. Mit Hilfe von DDE können Sie Befehle an Excel senden. Sie können außerdem die Daten in den Tabellen nach Namen manipulieren und lesen. Weitere Informationen über die Verwendung von DDE mit Excel finden Sie im *Microsoft Excel Benutzerhandbuch 2*.

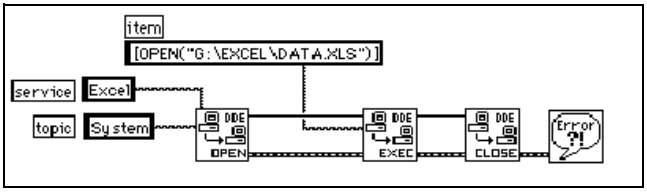
Der Servicename für Excel lautet `Excel`. Als Thema können Sie den Namen eines geöffneten Dokuments (z.B. eines Tabellendokuments) oder das Wort `System` verwenden.

Wenn Sie den Namen `System` verwenden, können Sie Informationen über den Status von Excel abrufen oder allgemeine Befehle (d.h. Befehle, die sich nicht auf eine spezielle Tabelle beziehen) an Excel senden. Bei dem Thema `System` kommuniziert Excel z.B. über solche Aspekte wie `Status`, der als `Busy` (Besetzt) angegeben wird, wenn Excel belegt ist, oder als `Ready` (Bereit), wenn Excel zum Ausführen von Befehlen bereit ist. `Topics` (Themen) ist ein weiteres nützliches Datenelement, das Sie mit dem Thema `System` verwenden können. Durch dieses Element erhalten Sie eine Liste der Themen, über die Excel kommunizieren kann, einschließlich aller geöffneten Tabellen und des Themas `System`.

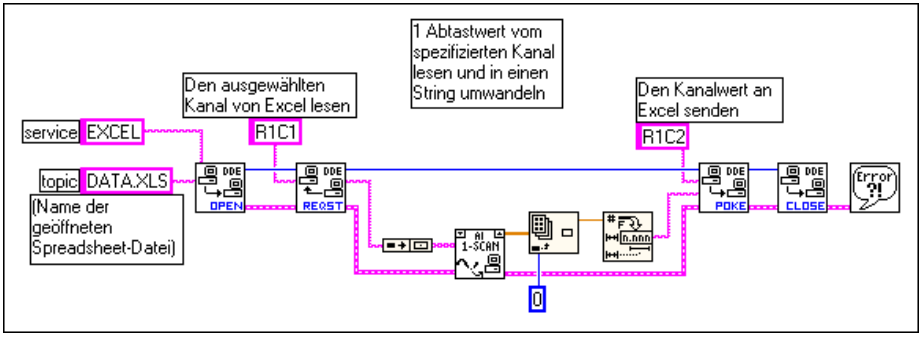
Das folgende VI illustriert, wie Sie den Befehl “Themen” in LabVIEW verwenden können. Als Wert wird ein String ausgegeben, der die Namen der geöffneten Tabellen und das Wort `System` enthält.



Eine andere Methode zur Verwendung des Themas `System` mit Excel besteht darin, Excel zum Öffnen eines speziellen Dokuments anzuweisen. Verwenden Sie das VI `DDE Execute.vi`, um ein Excel-Makro zu senden, durch das Excel angewiesen wird, das Dokument zu öffnen. Das folgende LabVIEW-Blockdiagramm verdeutlicht diese Methode.



Wenn Sie eine Tabellendatei geöffnet haben, können Sie Befehle an die Tabelle senden, die das Lesen der Zellenwerte bewirken. In diesem Fall entspricht das Thema dem Namen des Tabellendokuments. Hierbei wird der Name der Zelle, ein Bereich von Zellen oder ein genau bezeichneter Abschnitt der Tabelle als Element verwendet. Im folgenden Diagramm kann LabVIEW z.B. den Wert der Zelle in Reihe 1 und Spalte 1 abrufen. Danach erfaßt das Programm einen Beispielwert aus dem angegebenen Kanal und sendet das resultierende Beispielergebnis an Excel zurück.



LabVIEW-VIs als DDE-Server

Sie können LabVIEW-VIs erstellen, die als Server für Datenelemente eingesetzt werden können. Es wird dabei als allgemeines Konzept zugrundegelegt, daß ein LabVIEW-VI seine Bereitwilligkeit anzeigt, Informationen bezüglich eines speziellen Service- und Themenpaars zu liefern. LabVIEW kann einen beliebigen Service- und Themennamen verwenden. Es ist z.B. möglich, daß als Servicename der Name der Anwendung (`LabVIEW`) und als Themenname entweder der Name des Server-VIs oder eine allgemeine Klassifikation für die ausgegebenen Daten (z.B. `Lab Data`) verwendet wird.

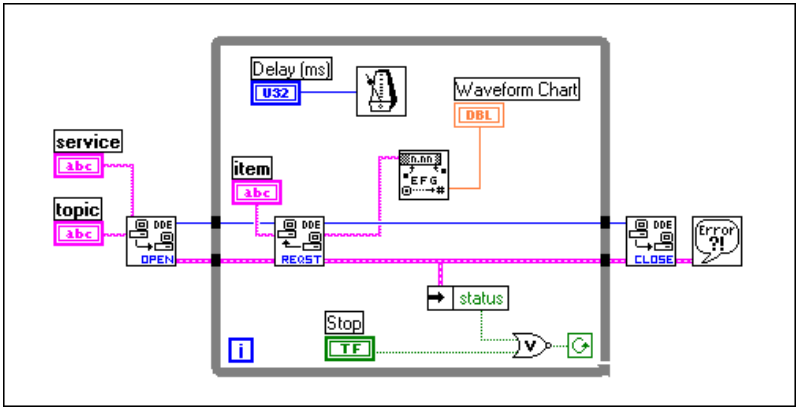
Das Server-VI registriert dann Datenelemente für den jeweiligen Service, auf den sich die Kommunikation bezieht. LabVIEW speichert die Datennamen und ihre Werte und wickelt die Kommunikation mit anderen Anwendungen bezüglich der Daten ab. Wenn das Server-VI den Wert der Daten ändert, die für die DDE-Kommunikation registriert sind, benachrichtigt LabVIEW alle Client-Anwendungen, die eine Benachrichtigung bezüglich dieser Daten angefordert haben. Auf dieselbe Weise ändert LabVIEW diesen Wert, wenn eine andere Anwendung eine `Poke`-Nachricht sendet, um den Wert eines Datenelements zu ändern.

Sie können den Befehl “DDE ausführen” nicht im Zusammenhang mit einem LabVIEW-VI verwenden, das als Server eingesetzt wird. Wenn Sie einen Befehl an ein VI senden möchten, müssen Sie den Befehl mit Hilfe von Datenelementen senden.

Beachten Sie auch, daß LabVIEW zum gegenwärtigen Zeitpunkt über kein Thema verfügt, das mit dem von Excel angebotenen Thema “System” vergleichbar ist. Die LabVIEW-Anwendung entspricht keinem eigentlichen DDE-Server, an den Sie Befehle senden oder von dem Sie Statusinformationen abrufen können. Sie können LabVIEW-VIs jedoch dazu verwenden, einen DDE-Server zu erstellen.

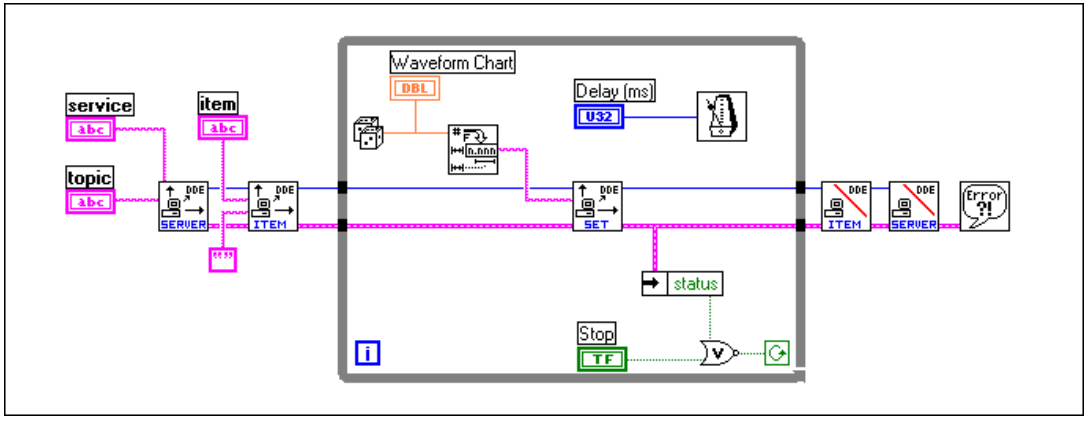
Das folgende Beispiel zeigt, wie ein DDE-Server-VI erstellt wird, das Daten an andere Client-Anwendungen liefert. In diesem Fall bestehen die Daten aus einer Zufallszahl. Sie können die Zufallszahl leicht durch reale

Daten von Datenerfassungskarten oder -geräten ersetzen, die über GPIB-, VXI- oder serielle Anschlüsse mit dem Computer verbunden sind.



Das VI im obigen Diagramm registriert einen Server unter LabVIEW. Das VI registriert ein Element, zu dessen Weitergabe an Clients das VI bereit ist. In der Schleife stellt das VI den Wert des Elements periodisch ein. Wie oben bereits erwähnt, benachrichtigt LabVIEW andere Anwendungen darüber, daß Daten verfügbar sind. Wenn die Schleife abgeschlossen ist, beendet das VI den Vorgang dadurch, daß die Registrierung für das Element und den Server wieder aufgehoben wird.

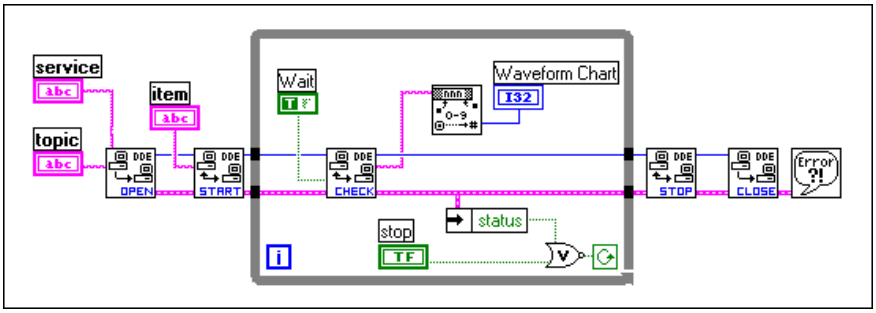
Als Client für dieses VI kommen alle Anwendungen in Frage, die DDE interpretieren können, einschließlich anderer LabVIEW-VIs. Das folgende Diagramm illustriert einen Client für das im vorherigen Diagramm gezeigte VI. Es ist dabei wichtig, daß die Dienst-, Themen- und Elementnamen mit den vom Server verwendeten Namen übereinstimmen.



Datenanforderungen gegenüber Datenbenachrichtigungen

Im vorherigen Client-Beispiel wird das DDE Anforderungs-VI in einer Schleife verwendet, um Daten abzurufen. Durch die DDE Anforderung werden die Daten sofort abgerufen; dabei spielt es keine Rolle, ob Sie die Daten zuvor bereits eingesehen haben oder nicht. Wenn der Server und der Client die Schleife nicht mit der gleichen Geschwindigkeit ausführen, besteht die Möglichkeit, daß Sie Daten verdoppeln oder verpassen.

Eine Möglichkeit, die Verdopplung von Daten zu vermeiden, besteht darin, DDE Advise-VIs zu verwenden, die zur Mitteilung über Änderungen im Wert eines Datenelements auffordern. Das folgende Diagramm illustriert die Anwendung dieses Schemas.



Im vorherigen Diagramm öffnet LabVIEW zuerst eine Verbindung. Danach fordert das Programm mit Hilfe des DDE Advise-Start-VIs zur Benachrichtigung über Änderungen im Wert eines Datenelements auf. Bei jedem Durchlauf der Schleife ruft LabVIEW das DDE Advise-Prüf-VI auf, das darauf wartet, daß ein Datenelement seinen Wert ändert. Wenn die Schleife abgeschlossen ist, beendet LabVIEW die Advise-Schleife durch Aufrufen des DDE Advise-Stopp-VIs und anschließendes Schließen der Verbindung.

Synchronisierung von Daten

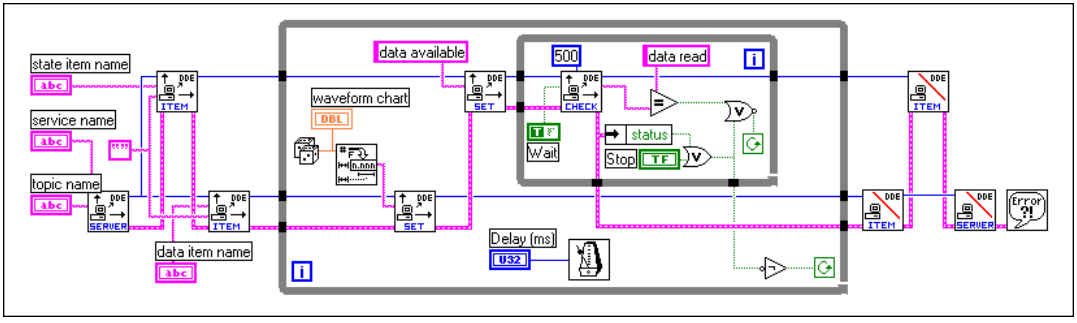
Die Client-Server-Beispiele im vorherigen Abschnitt eignen sich gut zum Überwachen von Daten. In diesen Beispielen gibt es jedoch keine Gewähr dafür, daß der Client alle Daten, die vom Server gesendet werden, auch tatsächlich erhält. Selbst bei Verwendung der DDE Advise-Schleife besteht die Möglichkeit, daß der Client einen vom Server ausgegebenen Datenwert verpaßt, wenn die Überprüfung auf Datenänderungen vom Client nicht häufig genug durchgeführt wird.

In einigen Anwendungen entstehen keine Probleme, wenn Daten verpaßt werden. Wenn Sie z.B. ein Datenerfassungssystem überwachen und dabei lediglich allgemeine Trends beobachten möchten, entstehen eventuell keinerlei Probleme, wenn nicht alle Daten registriert werden. In anderen Anwendungen müssen Sie unter Umständen allerdings darauf achten, daß keine Daten verlorengehen.

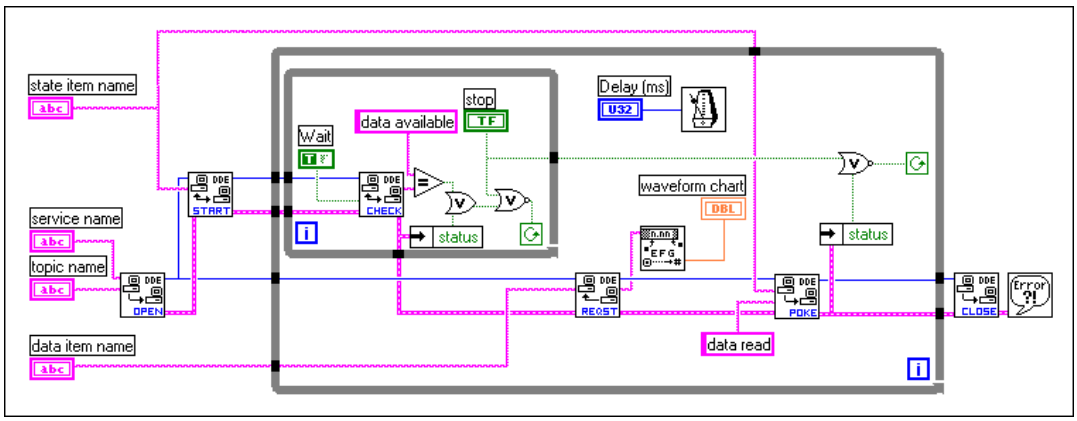
Ein wesentlicher Unterschied zwischen TCP und DDE besteht darin, daß TCP die Daten in eine Warteschlange einordnet, wodurch gewährt wird, daß Sie keine Daten verpassen und alle Daten in der richtigen Reihenfolge erhalten. DDE bietet keinen Service dieser Art.

Bei Verwendung von DDE können Sie ein separates Element einrichten, das der Client dazu verwendet, um den Empfang der neuesten Daten zu bestätigen. Sie können das erfaßte Datenelement dann so aktualisieren, daß es nur dann einen neuen Punkt enthält, wenn der Client den Empfang der vorherigen Daten bestätigt.

Sie können z.B. das in diesem Kapitel im Abschnitt *Datenanforderungen gegenüber Datenbenachrichtigungen* gezeigte Server-Beispiel so verändern, daß ein *Statuselement* auf einen bestimmten Wert eingestellt wird, nachdem das erfaßte Datenelement aktualisiert wurde. Der Server überwacht dann das *Statuselement*, bis der Client den Empfang von Daten bestätigt. Diese Veränderung ist im folgenden Blockdiagramm zu sehen.



Ein Client für den in der folgenden Abbildung gezeigten Server überwacht das Statuselement so lange, bis es als *Daten verfügbar* Status anzeigt. Zu diesem Zeitpunkt liest der Client dann die Daten im erfaßten Datenelement, das vom Server ausgegeben wurde, und aktualisiert das Statuselement auf den Wert *Daten gelesen*.



Auf diese Weise ist es möglich, den Datentransfer zwischen einem Server und einem einzelnen Client zu synchronisieren. Diese Methode ist jedoch nicht ohne Nachteile. Der erste Nachteil besteht darin, daß Sie lediglich einen Client verwenden können. Mehrere Clients können Konflikte untereinander verursachen. Es kann beispielsweise vorkommen, daß ein Client die Daten erhält und bestätigt, bevor der andere Client bemerkt, daß neue Daten verfügbar sind. Sie können kompliziertere DDE-Diagramme zusammenstellen, um dieses Problem zu beheben. Solche Diagramme werden jedoch schnell unübersichtlich.

Ein weiteres Problem dieser Methode zur Synchronisierung der Kommunikation besteht darin, daß die Geschwindigkeit der Datenerfassung durch die Übertragungsrate der Daten bestimmt wird. Dieses Problem läßt sich dadurch umgehen, daß Sie die Erfassung und die Übertragung in unterschiedliche Schleifen aufteilen. Die Erfassungsschleife reiht die Daten in eine Warteschlange ein, die die Übertragungsschleife sendet. Diese Struktur ist mit dem TCP-Server-Beispiel vergleichbar, in dem der Server mehrere Verbindungen bearbeitet.

Wenn für Ihre Anwendung eine zuverlässige Synchronisation des Datentransfers notwendig ist, sollten Sie stattdessen TCP/IP verwenden, da dadurch das Einreihen in eine Warteschlange, das Bestätigen des Datentransfers und das Unterstützen mehrerer Verbindungen auf der Treiberebene möglich ist.

DDE im Netzwerk

Sie können DDE dazu verwenden, mit Anwendungen auf demselben Computer oder auf anderen Computern im Netzwerk zu kommunizieren. Zur Verwendung von DDE über ein Netzwerk ist Windows für Workgroups 3.1 oder höher, Windows 95 oder Windows NT notwendig. Die Standardversion von Windows 3.1 unterstützt rein DDE in einer Netzwerkkonfiguration.

Jeder Computer unter Windows für Workgroups erhält einen Netzwerkcomputernamen. Sie können diesen Namen über die Netzwerksystemsteuerung einstellen.

Bei der Kommunikation über das Netzwerk ändert sich die Bedeutung der Dienste- und Themen-Strings. Durch den geänderten Servicenamen wird angezeigt, daß Sie DDE über ein Netzwerk verwenden möchten; der Servicename enthält außerdem den Namen des Computers, mit dem Sie kommunizieren möchten. Der Servicename weist folgende Form auf:

`\\`*`computer-name`*`\ndde$`

Sie können einen beliebigen Namen für das Thema verwenden. Anschließend bearbeiten Sie die Datei `SYSTEM.INI`, so daß der verwendete Themenname mit dem tatsächlichen Service und Thema assoziiert ist, der bzw. das auf dem Remote-Computer verwendet wird. Diese Konfiguration schließt auch Parameter ein, durch die die Netzwerkverbindung konfiguriert wird. Das folgende Beispiel zeigt, wie dieser Abschnitt danach aussehen könnte:

```
[DDE Shares]

topicname = appname, realtopic, ,31,,0,,0,0,0
```

Als `topicname` ist der Name einzusetzen, den das Client-VI für das Thema verwendet. `Appname` ist der Name der Remote-Anwendung. Bei Verwendung von DDE über ein Netzwerk muß dieser Name mit dem Servicenamen identisch sein. `Realtopic` ist das Thema, das auf dem Remote-Computer verwendet wird. Durch die restlichen Parameter wird festgelegt, wie DDE arbeitet. Verwenden Sie diese Parameter wie im vorherigen Beispiel aufgeführt. Microsoft stellt keine Erklärungen zur Bedeutung dieser Parameter zur Verfügung.

Wenn Sie beispielsweise zwei Computer, auf denen LabVIEW ausgeführt wird, so einrichten wollen, daß Sie über Netzwerk-DDE miteinander kommunizieren, muß der Server `LabVIEW` als Servicenamen und einen Namen wie z.B. `labdata` als Thema verwenden.

Falls als Computername beispielsweise `Lab` verwendet wird, versucht der Client eine Verbindung zu öffnen, indem er `\\Lab\ndde$` als Service verwendet. Als Thema kann der Client einen Namen von `remotelab` verwenden.

Damit diese Konfiguration ordnungsgemäß arbeiten kann, müssen Sie die Datei `SYSTEM.INI` auf dem Server-Computer so bearbeiten, daß die folgende Zeile im Abschnitt `[DDEShares]` vorhanden ist:

```
remotelab=LabVIEW,labdata,,31,,0,,0,0,0
```

Unter Windows NT müssen Sie die Datei `DDEShare.exe` starten; sie befindet sich im Verzeichnis `winnt/system 32`. Wählen Sie **Freigaben»DDE-Freigaben…**, und wählen Sie dann **Freigabe hinzufügen…**, um den Service- und Themennamen auf dem Server zu registrieren.

NetDDE verwenden

NetDDE ist in Windows für WorkGroups 3.11, Windows 95 und Windows NT integriert. Für Windows 3.1 kann NetDDE durch ein Zusatzpaket von WonderWare installiert werden. Wenn Sie Windows 3.1 mit dem WonderWare-Paket verwenden, sollten Sie sich in den Dokumentationsmaterialien von WonderWare über die Verwendung von NetDDE informieren.

Wenn Sie Windows für WorkGroups, Windows 95 oder Windows NT verwenden, können Sie sich an den folgenden Anleitungen orientieren:

Server-Computer

Windows für Workgroups

Ergänzen Sie für den Server (d.h. für die Anwendung, die DDE-Befehle erhält) die folgende Zeile im Abschnitt `[DDE Shares]` der Datei `system.ini`:

```
lvdemo = service_name,topic_name,,31,,0,,0,0,0
```

Dabei steht:

`lvdemo` für einen beliebigen Namen;

`service_name` in der Regel für den Namen der Anwendung, wie z.B. `excel`;

topic_name in der Regel für den speziellen Dateinamen, wie z.B. arbeitsblatt1.

Geben Sie die Kommas und die Zahlen exakt wie gezeigt ein.

Windows 95

☞ **Hinweis** ***NetDDE wird durch Windows 95 nicht automatisch gestartet. Sie müssen das Programm \WINDOWS\NETDDE.EXE ausführen. (Sie können dieses Programm in den Startordner aufnehmen, damit es bei jedem Systemstart aufgerufen wird.)***

Gehen Sie folgendermaßen vor, um einen NetDDE-Server unter Windows 95 einzurichten:

- Führen Sie \WINDOWS\REGEDIT.EXE aus.
- Öffnen Sie in der Baumstruktur den Ordner Arbeitsplatz\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetDDE\DDE Shares.
- Erstellen Sie eine neue DDE-Freigabe, indem Sie **Bearbeiten»Neu»Schlüssel** wählen, und geben Sie den Namen lvdemo ein.
- Achten Sie darauf, daß der lvdemo-Schlüssel markiert ist, und fügen Sie die im folgenden beschriebenen Werte hinzu, die für die Freigabe erforderlich sind. (Diese Schlüssel werden zwar lediglich aus dem CHAT$-Teil kopiert; REDEGIT erlaubt jedoch nicht das direkte Ausschneiden, Kopieren oder Einfügen von Schlüsseln oder Werten.) Verwenden Sie **Bearbeiten»Neu**, um neue Werte hinzuzufügen. Wenn Sie den Schlüssel erstellen, werden ein Standardwert mit der Bezeichnung (Default) und der Wert (value not set) verwendet. Lassen Sie diese Werte unverändert, und ergänzen Sie folgende Werte:

Tabelle 23-1. Ersatzwerte für den Standardwert

Wertetyp	Name	Wert
Binär	Zusätzliche Elementzählung	00 00 00 00
String	Anwendung	service_name
String	Element	service_name
String	Paßwort1	service_name
String	Paßwort2	service_name
Binär	Befugnis1	1f 00 00 00
Binär	Befugnis2	00 00 00 00
String	Thema	topic_name

- Schließen Sie `REGEDIT`.
- Starten Sie den Computer neu. (NetDDE muß neu gestartet werden, damit die Änderungen wirksam werden können.)

Windows NT

Rufen Sie die Datei `DDEShare.exe` auf, die sich im Verzeichnis `winnt\system32` befindet. Wählen Sie **Freigaben»DDE-Freigaben» Freigabe hinzufügen...**, um den Service- und Themennamen auf dem Server zu registrieren.

Client-Computer

Für den Client-Computer (d.h. für die Anwendung, die die DDE-Verbindung einleitet) sind keine Konfigurationsänderungen notwendig.

Verwenden Sie die folgenden Eingaben für das VI `DDE Open Conversation.vi`:

Dienst: `\\machine_name\ndde$`

Thema: `lvdemo`

Dabei steht:
machine_name für den Namen des Server-Computers;

lvdemo für den Namen, der im Abschnitt [DDE Shares] auf dem Server angegeben wird.

Sehen Sie sich die Beispiel-VIs Chart Client.vi und Chart Server.vi an, die Sie in examples\network\ddeexamp.llb finden. Wenn Sie diese VIs dazu verwenden möchten, mit Hilfe von NetDDE Informationen zwischen zwei Computern zu übertragen, sollten Sie folgendermaßen vorgehen:

Server-Computer:

1. Verändern Sie keine Werte auf dem Frontpanel.
2. Ergänzen Sie auf dem Server-Computer in der Datei system.ini die folgende Zeile im Abschnitt [DDEShares]:
   lvdemo = TestServer,Chart,,31,,0,,0,0,0

Client-Computer:

Stellen Sie die Bedienelemente auf dem Frontpanel folgendermaßen ein:
Dienst = \\machine_name\ndde$
Thema = lvdemo
Element = Random

24

AppleEvents

In diesem Kapitel wird das AppleEvents-Protokoll beschrieben, eine Form der Kommunikation zwischen Anwendungen (auch IAC für Interapplication Communication), die nur für den Macintosh verfügbar ist.

AppleEvents

Das AppleEvents-Protokoll wurde speziell für den Macintosh entwickelt, um die Kommunikation zwischen Anwendungen zu ermöglichen. Wie beim DDE-Protokoll verwenden die Anwendungen eine Nachricht, um Vorgänge von anderen Anwendungen anzufordern oder Informationen auszugeben. Eine Anwendung kann eine Nachricht an sich selbst, an eine andere Anwendung auf demselben Computer oder an eine Anwendung, die auf einem anderen Computer im Netzwerk ausgeführt wird, senden. Sie können AppleEvents dazu verwenden, Befehle (z.B. die Befehle zum Öffnen oder Drucken) oder Datenanforderungen (z.B. für Tabelleninformationen) an andere Anwendungen zu senden.

LabVIEW enthält VIs zum Senden einiger Befehle, die in den meisten Anwendungen verwendet werden. Die VIs lassen sich leicht verwenden und setzen keine detaillierten Kenntnisse der Funktionsweise von AppleEvents voraus. Die folgende Liste enthält einige Beispiele für den Einsatz von AppleEvents in LabVIEW-Anwendungen:

- Sie können LabVIEW anweisen, eine andere Anwendung (sogar eine Anwendung auf einem anderen Computer im Netzwerk) zum Ausführen eines bestimmten Vorgangs aufzufordern. LabVIEW kann beispielsweise ein Tabellenkalkulationsprogramm auffordern, einen Graphen zu erstellen. Detaillierte Hinweise finden Sie im Abschnitt *AppleEvents senden* in diesem Kapitel.
- Sie können ein AppleScript-Programm als Anfangsprogramm verwenden, durch das LabVIEW zum Ausführen spezieller VIs veranlaßt wird.

- Durch das Senden von Anleitungen für das Durchführen spezieller Operationen können Sie mit LabVIEW-Anwendungen auf anderen Computern im Netzwerk kommunizieren und diese Anwendungen steuern. Detaillierte Hinweise finden Sie im Abschnitt *AppleEvents senden* in diesem Kapitel.
- Sie können LabVIEW anweisen, Nachrichten an sich selbst zu senden, durch die einzelne VIs geladen, ausgeführt und entladen werden. In umfangreichen Anwendungen, die viel Speicherplatz beanspruchen, können Sie z.B. SubVI-Aufrufbefehle durch ein Utility-VI (das AESend Open, Run, Close-VI) ersetzen und die VIs dynamisch laden, ausführen und entladen.

Diese VIs verwenden das Low-Level-AESend-VI zum Senden von AppleEvents. Apple hat ein umfangreiches *Vokabular* für Nachrichten definiert, um die Standardisierung der AppleEvent-Kommunikation zu erleichtern. Sie können in diesem Vokabular *Wörter* (Words) kombinieren, um komplexe Nachrichten zusammenzustellen. Mit diesem VI können Sie arbiträre AppleEvents an andere Anwendungen senden. Das Erstellen und Senden von AppleEvents auf dieser Ebene ist jedoch kompliziert und erfordert ein detailliertes Verständnis von AppleEvents. Sie finden weitere Hinweise in *Inside Macintosh* und in *AppleEvent Registry.*

AppleEvents senden

Die Unterpalette **Kommunikation** in der Palette **Funktionen** enthält VIs zum Senden von AppleEvents. Mit diesen VIs können Sie eine Zielanwendung für ein AppleEvent auswählen, AppleEvents erstellen und die AppleEvents an die Zielanwendung senden.

Die Palette **AppleEvent VIs** in der Unterpalette **Kommunikation** enthält VIs, die spezielle AppleEvent-Nachrichten senden. Mit diesen VIs können Sie mehrere Standard-AppleEvents (Open Document, Print Document und Close Application) und alle an LabVIEW angepaßten AppleEvents senden. Für diese High-Level-VIs sind nur geringe Kenntnisse der AppleEvent-Programmierung notwendig. Die Diagramme für diese AppleEvents stellen gute Beispiele für das Erstellen und Senden von AppleEvents dar.

Sie können das Low-Level-AESend-VI verwenden, um ein AppleEvent zu senden, für das LabVIEW kein VI bereitstellt. Die Palette **AppleEvent-VIs** in der Subpalette **Kommunikation** enthält ebenfalls VIs, mit denen Sie ein AppleEvent erstellen können. Für das Erstellen und Senden eines AppleEvents auf dieser Ebene sind jedoch detaillierte Kenntnisse der AppleEvent-Programmierung notwendig. Sie finden Hinweise in *Inside Macintosh, Volume VI* und in *AppleEvent Registry.*

Client-Server-Modell

Sie können die AppleEvent-VIs nicht dazu verwenden, LabVIEW-Diagramme zu erstellen, die als Server dienen können. Die VIs werden zum Senden von Nachrichten an andere Anwendungen verwendet. Wenn Sie Server-Fähigkeiten auf Diagrammbasis benötigen, müssen Sie die Protokolle TCP oder PPC verwenden.

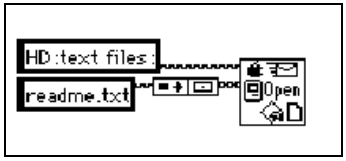
LabVIEW kann eine Reihe von AppleEvents verstehen und darauf reagieren; damit verhält sich das Programm gewissermaßen wie ein AppleEvent-Server. Insbesondere können Sie LabView mit Hilfe von AppleEvents anweisen, VIs zu öffnen, zu drucken, auszuführen und zu schließen. Sie können von LabVIEW Informationen darüber erhalten, ob ein bestimmtes VI gerade ausgeführt wird, und Sie können LabVIEW durch ein AppleEvent auch beenden.

Durch Verwendung dieser Server-Fähigkeiten sind Sie in der Lage, Anweisungen zum Ausführen von VIs an andere LabVIEW-Anwendungen zu senden, und Sie können LabVIEW über eine Remote-Verbindung steuern. Sie können LabVIEW auch dazu auffordern, Nachrichten an sich selbst zu senden, die Anweisungen zum Laden spezieller VIs enthalten. In umfangreichen Anwendungen, die über begrenzte Speicherkapazitäten verfügen, können Sie z.B. SubVI-Aufrufbefehle dadurch ersetzen, daß Sie das AESend Open, Run, Close-VI aufrufen, durch das VIs je nach Bedarf geladen und ausgeführt werden. Sie werden bemerken, daß sich das Frontpanel für das VI öffnet, das Sie auf diese Weise ausführen—gerade so, als hätten Sie **Datei»Öffnen...** gewählt.

Beispiele für AppleEvent-Clients

Andere Anwendungen starten

An eine andere Anwendung kann nur dann eine Nachricht gesendet werden, wenn diese Anwendung läuft. Mit Hilfe des AESend Finder Open-VIs ist es möglich, eine andere Anwendung zu starten. Dieses VI sendet eine Nachricht an den Finder. Der Finder ist selbst eine Anwendung, die eine begrenzte Anzahl von AppleEvents versteht. Das folgende einfache Beispiel zeigt, wie mit Hilfe von AppleEvents eine bestimmte Textdatei in TeachText geöffnet werden kann.



Wenn sich die Anwendung auf einem Remote-Computer befindet, müssen Sie den Standort des Computers angeben. Durch Eingaben für das AESend Finder Open-VI können Sie die Netzwerkzone und den Servernamen des Computers, mit dem Sie kommunizieren möchten, festlegen. Falls die Netzwerkzone und der Servername wie in der vorherigen Anwendung nicht angegeben werden, werden standardmäßig die Angaben für den aktuellen Computer verwendet.

Wenn Sie versuchen, Nachrichten an einen anderen Computer zu senden, werden Sie automatisch aufgefordert, sich bei diesem Computer anzumelden. Diese Eingabeaufforderung kann nicht umgangen werden, da sie in das Betriebssystem integriert ist. Dadurch können Probleme entstehen, wenn Sie beabsichtigen, Ihre Anwendung auf einem nicht überwachten Computersystem auszuführen.

Events an andere Anwendungen senden

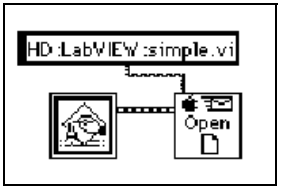
Sobald eine Anwendung läuft, können Sie andere AppleEvents verwenden, um Nachrichten an diese Anwendung zu senden. Nicht alle Anwendungen unterstützen AppleEvents, und selbst die Anwendungen, die geeignet sind, unterstützen eventuell nicht alle veröffentlichten AppleEvents. Sehen Sie in der mit Ihrer Anwendung gelieferten Dokumentation nach, welche AppleEvents unterstützt werden.

Falls die Anwendung AppleEvents versteht, können Sie ein AppleEvent mit der Ziel-ID für die Anwendung aufrufen. Eine Ziel-ID kennzeichnet ein Cluster, das einen Zielstandort im Netzwerk (Zone, Server und unterstützende Anwendung) beschreibt. Sie müssen sich nicht um die genaue Struktur dieses Clusters kümmern, da LabVIEW VIs zur Verfügung stellt, mit deren Hilfe Sie eine Ziel-ID generieren können.

Es gibt zwei Methoden, eine Ziel-ID zu erstellen. Bei der ersten Methode können Sie mit dem Get Target ID-VI eine Ziel-ID durch das Programm aufgrund des Anwendungsnamens und des Netzwerkstandorts erstellen. Bei der zweiten Methode verwenden Sie das PPC Browser-VI, das in einem Dialogfeld die Netzwerkanwendungen anzeigt, die AppleEvents verstehen. In dieser Liste können Sie interaktiv Ihre Auswahl treffen, um eine Ziel-ID zu erstellen.

Mit dem PPC Browser-VI können Sie außerdem feststellen, ob eine andere Anwendung AppleEvents verwendet. Wenn Sie das VI ausführen und den Computer auswählen, auf dem die Anwendung ausgeführt wird, wird die Anwendung im Dialogfeld angezeigt, falls sie AppleEvents verarbeiten kann.

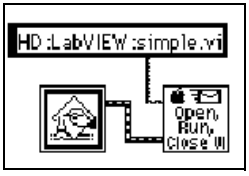
Das folgende Diagramm zeigt, wie LabVIEW eine AppleEvents-fähige Anwendung im Netzwerk interaktiv auswählt und die Anwendung zum Öffnen eines Dokuments anweist. Im vorliegenden Fall bewirkt LabVIEW, daß die Anwendung ein VI öffnet.



VIs dynamisch laden und ausführen

Das AESend Open, Run, Close-VI sendet Nachrichten, durch die LabVIEW zum Ausführen eines VIs aufgefordert wird. Zuerst wird eine Nachricht zum Öffnen eines Dokuments gesendet, und LabVIEW öffnet ein VI. Danach sendet das Open, Run, Close-VI die Nachricht zum Ausführen eines VIs an LabVIEW, und LabVIEW führt das angegebene VI aus. Das Open, Run, Close-VI sendet nun die Nachricht “VI Active?”, und LabVIEW gibt Statusinformationen zum angegebenen VI aus, solange das VI läuft. Schließlich sendet das VI die Nachricht zum Schließen des VIs.

Sofern sich LabVIEW, d.h. die Zielanwendung, auf einem anderen Computer befindet, können Sie das folgende Diagramm direkt übernehmen, um das VI zu laden und auszuführen. Falls Sie das VI an die aktuelle LabVIEW-Anwendung senden, benötigen Sie das PPC Browser-VI nicht.



25

Programm-zu-Programm-Kommunikation

In diesem Kapitel wird die Programm-zu-Programm-Kommunikation PPC (Program-to-Program Communication) beschrieben. Es handelt sich dabei um eine Low-Level-Version der Kommunikation zwischen Apple-Anwendungen (Interapplication Communication/IAC), die Anwendungen auf dem Macintosh zum Senden und Empfangen von Datenblöcken verwenden.

Einführung in PPC

Bei der Programm-zu-Programm-Kommunikation PPC handelt es sich um ein Protokoll für den Macintosh zum Übertragen von Datenblöcken zwischen Anwendungen. Sie können mit PPC VIs erstellen, die als Clients oder Server eingesetzt werden können. Die Programm-zu-Programm-Kommunikation wird zwar von allen Macintosh-Computern unterstützt, auf denen System 7.*x* ausgeführt wird; von den meisten Anwendungen für Macintosh wird sie jedoch nicht verwendet. Stattdessen kommunizieren die meisten Anwendungen auf Macintosh-Computern mit Hilfe von AppleEvents, einem High-Level-Protokoll zum Senden von Befehlen zwischen Anwendungen.

Obwohl PPC nicht dieselbe breite Unterstützung erhält wie AppleEvents, bietet es einige Vorteile. Da PPC auf einer niedrigeren Stufe arbeitet, können damit bessere Leistungen als mit AppleEvents erzielt werden. Außerdem können Sie in LabVIEW VIs erstellen, die als Clients oder Server fungieren, indem Sie PPC verwenden. Sie können dagegen keine Diagramme erstellen, die als AppleEvent-Server einsetzbar sind.

LabVIEW-VIs können PPC verwenden, um große Datenmengen zwischen Anwendungen auf demselben Computer oder verschiedenen Computern im Netzwerk auszutauschen. Damit zwei Anwendungen über PPC miteinander kommunizieren können, müssen beide Anwendungen laufen und zum Senden oder Empfangen von Daten bereit sein.

In Bezug auf Server- und Client-Anwendungen haben PPC und TCP eine ähnliche Struktur. Die PPC-Methode zur Kennzeichnung einer Remote-Anwendung unterscheidet sich von der TCP-Methode. Davon abgesehen bieten beide Protokolle ähnliche Leistungen und Funktionen. Beide Protokolle können Warteschlangen verwalten und Daten auf zuverlässige Weise übertragen. Sie können beide Protokolle für mehrere offene Verbindungen verwenden.

Bei der Entscheidung zwischen TCP und PPC sollten Sie bedenken, auf welcher Plattform Sie Ihre VIs auszuführen beabsichtigen und mit welchen Plattformen Sie kommunizieren werden. Wenn die Anwendung nur auf Macintosh-Computern eingesetzt wird, kann es von Vorteil sein, PPC zu verwenden, da dieses Protokoll ins Betriebssystem integriert ist. TCP ist in die Macintosh-Betriebssysteme der Version 7.5 und höher integriert. Wenn Sie TCP mit einem niedrigeren System verwenden möchten, müssen Sie einen separaten TCP/IP-Treiber von Apple erwerben. Wenn der Kauf des separaten Treibers problemlos möglich ist, ist die Verwendung von TCP zu empfehlen, da die TCP-Benutzeroberfläche einfacher zu benutzen ist als die Benutzeroberfläche von PPC. PPC verwendet nämlich einige recht komplizierte Datenstrukturen zum Beschreiben von Adressen.

Wenn Ihre Anwendung mit anderen Plattformen kommunizieren oder auf anderen Plattformen laufen muß, sollten Sie TCP/IP verwenden.

Ports, Ziel-IDs und Sessions

Für die Kommunikation über PPC müssen sowohl der Client als auch der Server Ports öffnen, die für die anschließende Kommunikation verwendet werden. Das Open Port-VI öffnet den Port mit Hilfe eines Clusters, das unter anderem den Namen enthält, den Sie für den Port verwenden möchten. Über die Ports ist es möglich, zwischen verschiedenen Services zu unterscheiden, die eine Anwendung bietet. In jeder Anwendung können mehrere Ports gleichzeitig geöffnet sein.

Jeder Port kann mehrere simultane Sessions oder Verbindungen unterstützen. Zum Öffnen einer Session verwendet ein Client eine Ziel-ID, durch die auf den Standort des Servers verwiesen wird. PPC verwendet dieselbe Art von Ziel-IDs, die auch von AppleEvent-VIs benutzt werden. Sie können die Ziel-ID für die Remote-Anwendung entweder mit dem PPC Browser-VI oder mit dem Get Target ID-VI generieren.

Ein Server wartet darauf, daß Clients versuchen, eine Session mit dem PPC Inform Session-VI zu öffnen. Der Server kann mit Hilfe des PPC Accept Session-VIs die Session akzeptieren oder ablehnen. Ein Client

kann auch versuchen, mit dem PPC Start Session-VI eine Session mit einem Server zu öffnen.

Nach dem Start der Session können Sie das PPC Read- und das PPC Write-VI zum Übertragen von Daten verwenden. Zum Schließen einer Session können Sie PPC End Session verwenden. Einen Port können Sie mit dem PPC Close Port-VI schließen.

Beispiel für einen PPC-Client

Im folgenden Abschnitt wird erläutert, wie Sie mit PPC jede einzelne Komponente des allgemeinen Client-Modells erfüllen können.

Öffnen Sie eine Verbindung zum Server mit dem PPC Open Connection-VI und dem PPC Open Session-VI. Dabei müssen Sie die Ziel-ID des Servers angeben, die Sie entweder mit dem PPC Browser-VI oder dem Target ID-VI ermitteln können. Als Endresultat erhalten Sie eine Port-Refnum und eine Session-Refnum, die beide zur Kommunikation mit dem Server verwendet werden.

Senden Sie einen Befehl, den Sie auf dem Server ausführen möchten, mit Hilfe des PPC Write-VIs an den Server. Verwenden Sie dann das PPC Read-VI, um die vom Server gesendeten Ergebnisse zu lesen. Für das PPC Read-VI müssen Sie die Anzahl der Zeichen angeben, die gelesen werden sollen. Wie bei TCP besteht auch hier eine mögliche Komplikation darin, daß die Länge der Antwort variieren kann. Ein ähnliches Problem ergibt sich auch für den Server, da die Länge eines Befehls ebenfalls variieren kann.

Die im folgenden beschriebenen Ansätze gehen mit dem Problem, das sich aus der wechselnden Länge der Befehle ergibt, auf unterschiedliche Weise um. Diese Methoden eignen sich ebenso für TCP.

- Stellen Sie dem Befehl und dem Ergebnis einen Größenparameter voran, durch den die Größe des Befehls oder Ergebnisses festgelegt wird. Lesen Sie in diesem Fall zuerst die Größenangabe des Parameters und dann die Anzahl an Zeichen, die durch die Größe angegeben wird. Diese Option ist effizient und flexibel.
- Benutzen Sie nur Befehle und Ergebnisse mit einer festen Größe. Sie können Befehle, die zu kurz sind, durch Füllzeichen verlängern.
- Hängen Sie an jeden Befehl und jedes Ergebnis ein spezielles Abschlußzeichen an. Sie können die Daten dann in kleinen Mengen lesen, bis Sie auf das Abschlußzeichen stoßen.

Schließen Sie die Verbindung zum Server mit dem PPC Close Session-VI und dem PPC Close Connection-VI.

Beispiel für einen PPC-Server

Im folgenden Abschnitt wird erläutert, wie Sie mit PPC jede einzelne Komponente des allgemeinen Server-Modells erfüllen können.

Verwenden Sie das PPC Open Port-VI während der Initialisierungsphase, um einen Kommunikationsport zu öffnen.

Verwenden Sie das PPC Inform Session-VI, um auf eine Verbindung zu warten. Bei der Programm-zu-Programm-Kommunikation können Sie entweder automatisch eingehende Verbindungsanfragen akzeptieren, oder Sie können mit dem PPC Accept Session-VI entscheiden, ob Sie die Sitzung akzeptieren oder ablehnen möchten. Auf diese Weise haben Sie die Möglichkeit, unerwünschte Verbindungen herauszufiltern, indem Sie ihnen die Bestätigung verweigern.

Nach dem Herstellen einer Verbindung haben Sie Lesezugang zu dieser Sitzung, und Sie können einen Befehl abrufen. Wie bereits im Abschnitt *Beispiel für einen PPC-Client* besprochen wurde, müssen Sie über das Format für die Befehle entscheiden. Wenn den Befehlen ein Längenfeld vorangeht, müssen Sie zuerst das Längenfeld und dann die darin angegebene Datenmenge lesen.

Da die Ausführung eines Befehls auf dem lokalen Computer stattfindet, sollte dieser Vorgang protokollunabhängig sein. Nach der Befehlsausführung leiten Sie die Ergebnisse an die nächste Stufe weiter, von wo sie an den Client übertragen werden.

Verwenden Sie das PPC Write-VI, um das Ergebnis auszugeben. Wie bereits im Abschnitt *Beispiel für einen PPC-Client* besprochen wurde, müssen die Daten in einem Format vorliegen, das der Client akzeptiert.

Schließen Sie die Verbindung mit dem PPC Close Session-VI.

Verwenden Sie das PPC Close Port-VI, um den Port, den Sie während der Initialisierungsphase geöffnet haben, nach Ende der Bearbeitung durch den Server zu schließen.

PPC-Server mit mehreren Verbindungen

PPC kann mühelos mehrere Sessions und Ports verwalten. Die im vorherigen Abschnitt beschriebenen Methoden zur Implementierung der einzelnen Komponenten eines Servers eignen sich ebenso gut für Server mit mehreren Verbindungen. Abbildung 25-1 illustriert die Reihenfolge, in der die PPC-VIs verwendet werden.

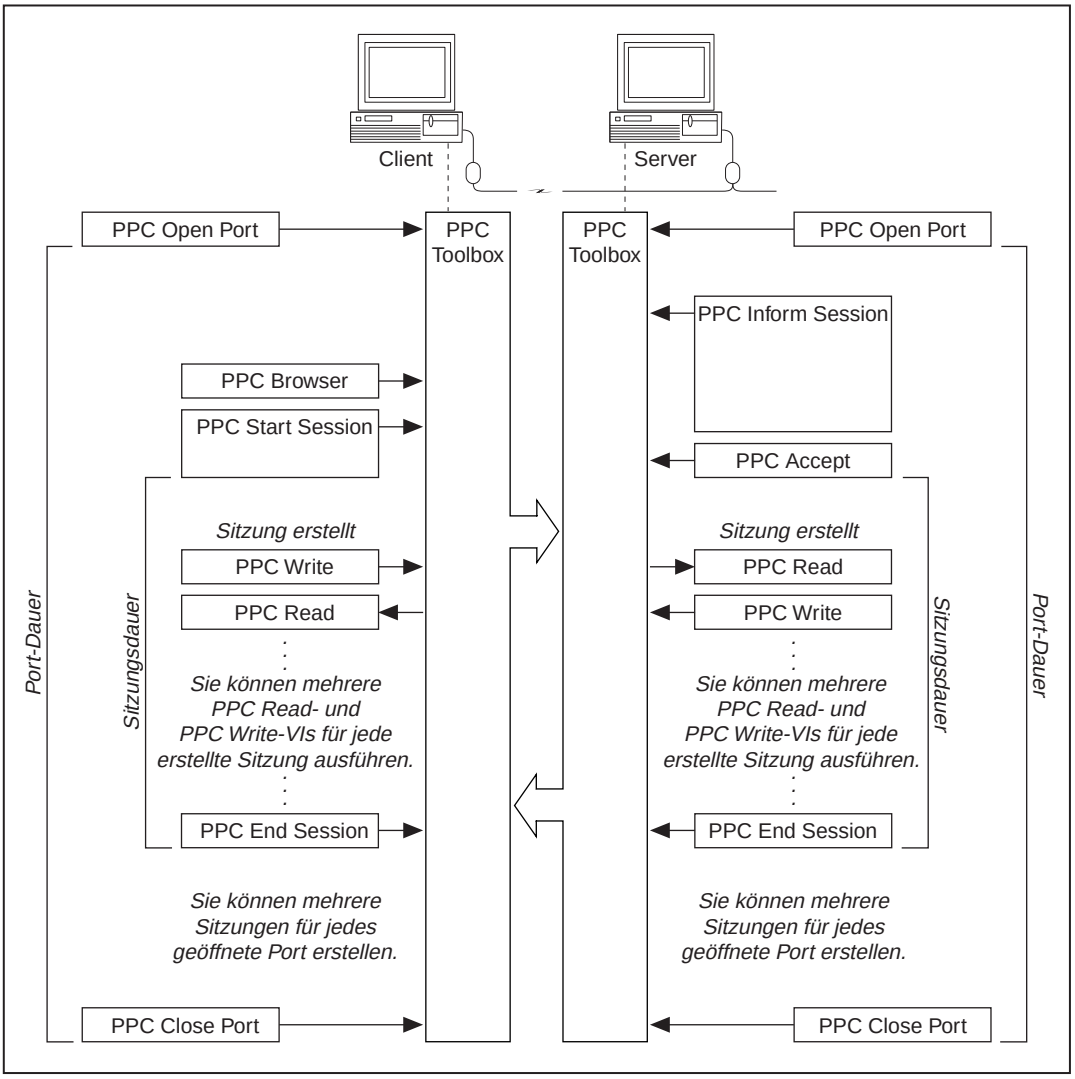


Abbildung 25-1. Ausführreihenfolge der PPC-VIs (Mit freundlicher Genehmigung von Apple Computer, Inc.)

Teil V

Fortgeschrittenes Programmieren in G

Dieser Teil des Handbuchs enthält Informationen über das Anpassen von VIs, das Konfigurieren von Frontpanel-Objekten durch das Programm und das Design komplexer Anwendungen.

Teil V, *Fortgeschrittenes Programmieren in G*, enthält die folgenden Kapitel.

- Kapitel 26, *VIs anpassen*, erläutert das Anpassen von VIs. Außerdem enthält dieses Kapitel Übungen, durch die illustriert wird, wie Sie die Optionen **VI-Einstellungen…** und **Einstellungen SubVI Knoten…** verwenden können, um das Aussehen und das Ausführverhalten eines laufenden VIs anzupassen.
- Kapitel 27, *Attribute der Frontpanel-Objekte*, beschreibt die als Attributknoten bezeichneten Objekte, die spezielle Blockdiagrammknoten zur Steuerung des Aussehens und der funktionalen Charakteristiken von Bedien- und Anzeigeelementen darstellen.
- Kapitel 28, *Programmdesign*, erläutert Methoden, die zum Erstellen von Programmen verwendet werden können, und beschreibt Richtlinien für den Programmierstil.
- Kapitel 29, *Weiteres Vorgehen*, bietet Informationen über Ressourcen, die Ihnen das Erstellen gelungener Anwendungen erleichtern können.

☞ **Hinweis** **(Windows 3.1)** ***Sie müssen die VIs, die Sie in Teil V erstellen, in VI-Bibliotheken speichern. In VI-Bibliotheken können Sie Dateinamen verwenden, die mehr als 8 Zeichen umfassen. Darüber hinaus befinden sich auch die VIs, die für die Übungen in Teil V benötigt werden, in der VI-Bibliothek*** `LabVIEW\Activity\Activity.llb`***. Weitere Informationen zu VI-Bibliotheken finden Sie im Abschnitt*** *Speichern von VIs* ***in Kapitel 2,*** *Bearbeiten von VIs,* ***im*** *Referenzhandbuch zur Programmierung in G****.***

Nicht nur Frontpanel-Objekte lassen sich durch das Programm konfigurieren; sondern auch VIs und selbst LabVIEW können durch das Programm und, das Ausführverhalten eines VIs und die Druckeinstellungen für LabVIEW ändern und alle VIs bestimmen, die in den Arbeitsspeicher geladen werden. Einige der Optionen, die Sie interaktiv über **Voreinstellungen** und **VI-Einstellungen…** einstellen können, können auch durch das Programm eingestellt werden. Die über **Voreinstellungen** verfügbaren Optionen stellen Anwendungseinstellungen dar, da Sie für alle VIs in LabVIEW gelten. Die Optionen unter **VI-Einstellungen…** sind VI-Einstellungen, da sie sich nur auf ein einzelnes VI und alle in einem SubVI eingesetzten Instanzen dieses VIs auswirken. Sie können nicht nur die Konfiguration eines VIs und von LabVIEW ändern, sondern auch einzelne Aktionen ausführen und verschiedene Methoden anwenden. So können Sie z.B. alle Frontpanel-Objekte auf ihre Standardwerte zurücksetzen.

Das dynamische Laden und Aufrufen eines VIs stellt ein weiteres Beispiel für die Steuerung durch das Programm dar. Wenn Sie mit einer großen Anwendung arbeiten, in der viele SubVIs vorhanden sind, werden alle SubVIs in den Arbeitsspeicher geladen, wenn Sie das Top-Level-VI laden. Bei großen Anwendungen ist es jedoch unter Umständen unter praktischen Gesichtspunkten nicht empfehlenswert, alle SubVIs in den Arbeitsspeicher zu laden. Sie können die SubVIs stattdessen auch dynamisch aufrufen, indem Sie den Knoten “Aufruf über Referenz” verwenden.

Weitere Informationen zum programmgestützten Konfigurieren und Steuern von VIs und von LabVIEW finden Sie in Kapitel 21, *VI-Server*, im *Referenzhandbuch zur Programmierung in G* und in der *Online-Referenz* für LabVIEW.

Ebenso wie auf einem lokalen Computer können Sie VIs und LabVIEW auch auf einem Remote-Computer über ein TCP/IP-Netzwerk konfigurieren und steuern. Weitere Informationen finden Sie in Kapitel 7, *Anpassen Ihrer Umgebung*, im *Referenzhandbuch zur Programmierung in G*. Außerdem können Sie VIs und LabVIEW auch über eine andere ActiveX-Anwendung konfigurieren und steuern. Weitere Informationen zur Unterstützung von ActiveX in LabVIEW finden Sie in Kapitel 22, *ActiveX-Unterstützung*, in diesem Handbuch.

26

VIs anpassen

In diesem Kapitel wird erklärt, wie Sie VIs Ihren Anforderungen anpassen können. Außerdem enthält dieses Kapitel Übungen, durch die das Ausführen der folgenden Aufgaben illustriert wird:

- Die Option **VI-Einstellungen…** verwenden
- Die Option **Einstellungen SubVI Knoten…** verwenden

Beispiele für angepaßte VIs finden Sie in `Examples\General\viopts.llb`.

Wie wird ein VI angepaßt?

Für das Ausführen eines VIs werden mehrere Konfigurationsmöglichkeiten angeboten. Sie können auf diese Optionen zugreifen, indem Sie das Popup-Menü für das Icon-Feld in der oberen rechten Ecke des Frontpanels aufrufen und **VI-Einstellungen…** wählen.

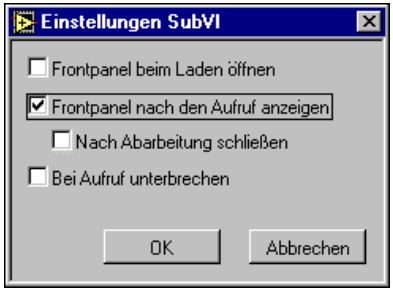
Daraufhin wird ein Dialogfeld für die VI-Einstellungen eingeblendet, in dem Sie Optionen für das Ausführen des VIs, das Aussehen des Panels, die Dokumentation und das Aussehen der Runtime-Menüleiste einstellen können. In Übung 26-1 erfahren Sie, wie Sie mit diesen Optionen umgehen können. Weitere detaillierte Informationen finden Sie in Kapitel 6, *VIs und SubVIs einrichten*, im *Referenzhandbuch zur Programmierung in G*.

Fensteroptionen einstellen

Verwenden Sie die Menüoption **Fensteroptionen**, um das Aussehen eines laufenden VIs zu steuern. Klicken Sie auf den nach unten zeigenden Pfeil in der Menüleiste, wenn Sie von **Ausführungsoptionen** zu **Fensteroptionen** wechseln möchten.

Einstellungen für SubVI-Knoten

Sie können auch Konfigurationseinstellungen für das Ausführen eines SubVIs vornehmen. Diese Konfigurationsoptionen stehen über das Popup-Menü für das SubVI-Icon im Blockdiagramm des aufrufenden VIs zur Verfügung. Wählen Sie dort die Menüoption **Einstellungen SubVI Knoten…**. Die folgende Abbildung zeigt das Dialogfeld “Einstellungen SubVI Knoten”.



☞ **Hinweis** ***Optionen, die Sie im Dialogfeld “VI-Einstellungen” für ein VI wählen, gelten für alle Instanzen dieses VIs. Wenn Sie dagegen eine Option im Dialogfeld “Einstellungen SubVI Knoten” wählen, gilt die Einstellung nur für den betreffenden Knoten.***

Übung 26-1. Einstellungsoptionen für ein SubVI verwenden

Übungsziel ist das Anfertigen eines VIs, das den Benutzer zur Eingabe von Informationen auffordert.

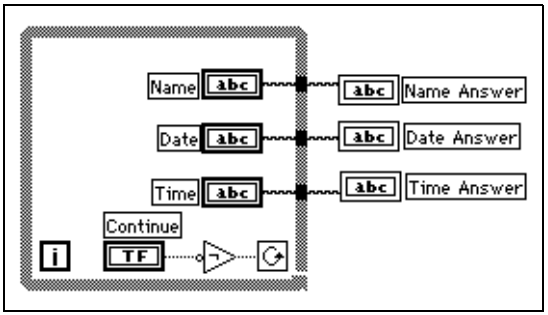
In dieser Übung erstellen Sie ein VI, das ein Dialogfeld öffnet, in dem der Benutzer bei der Ausführung zur Eingabe bestimmter Informationen aufgefordert wird. Wenn der Benutzer nach der Eingabe auf eine Schaltfläche klickt, wird das Dialogfeld wieder ausgeblendet.

Frontpanel

1. Öffnen Sie ein neues Frontpanel, und fügen Sie die String-Steuerelemente und die Schaltfläche hinzu, die in der folgenden Abbildung gezeigt werden.

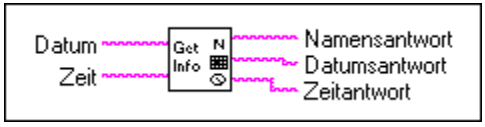
Blockdiagramm

2. Stellen Sie das Blockdiagramm zusammen, das in der folgenden Abbildung zu sehen ist.



3. Erstellen Sie das links neben diesem Text abgebildete Icon für das VI. Sie können auf den Icon-Editor zugreifen, indem Sie das Popup-Menü für das Icon-Feld im Frontpanel aufrufen und **Icon bearbeiten** wählen.
4. Wechseln Sie zum Anschlußfeld, indem Sie das Popup-Menü für das Icon-Feld aufrufen und **Anschluß anzeigen** wählen.

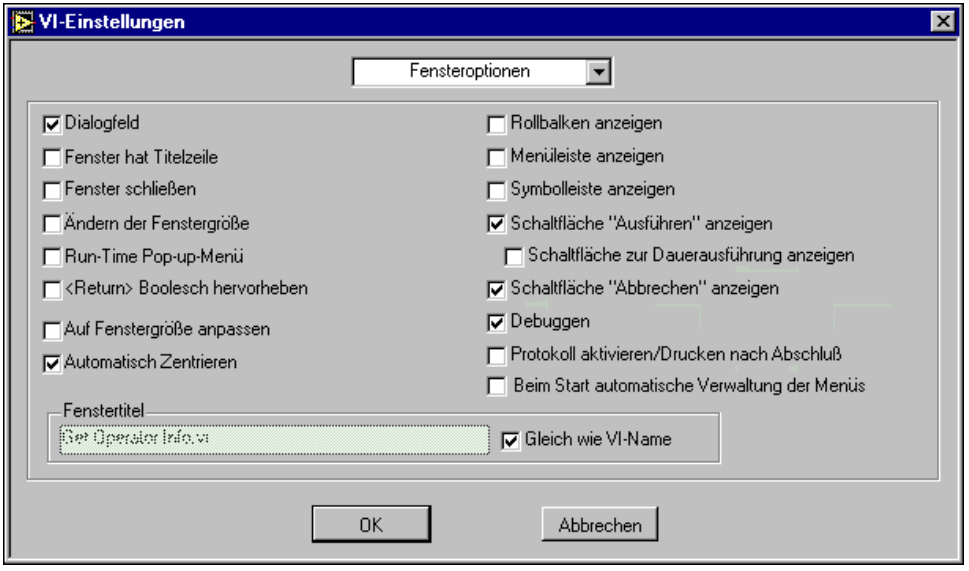
5. Erstellen Sie den Anschluß. Beachten Sie dabei, daß das Standard-Anschlußfeld anders aussieht als das links neben diesem Text abgebildete Anschlußfeld. Wählen Sie **Pattern** im Popup-Menü für den Anschluß, um das richtige Anschlußfeld zu erstellen. Wählen Sie das Muster mit drei Eingängen und zwei Ausgängen. Wählen Sie dann **Horizontal kippen**. Sie können nun entsprechend der folgenden Abbildung die Bedienelemente **Datum** und **Zeit** mit den beiden Anschlüssen auf der linken Seite des Icons und die Anzeigeelemente **Name-Antwort**, **Datum-Antwort** und **Zeit-Antwort** mit den drei Anschlüssen auf der rechten Seite des Icons verbinden. Kehren Sie zur Icon-Anzeige zurück, wenn Sie den Anschluß fertiggestellt haben.



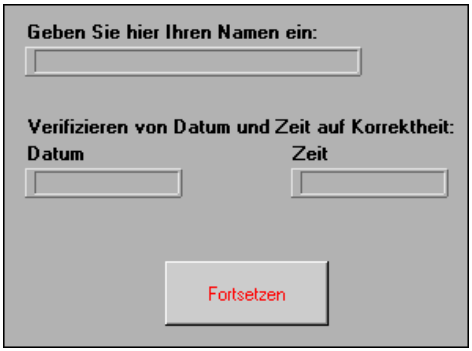
6. Speichern Sie das VI unter der Bezeichnung `Benutzer-Info.vi` im Verzeichnis `LabVIEW\Activity`.
7. Sie können nun damit beginnen, das VI mit Hilfe der Optionen unter **VI-Einstellungen** so zu gestalten, daß es wie ein Dialogfeld aussieht.

a. Rufen Sie das Popup-Menü für das Icon auf, und wählen Sie **VI-Einstellungen**. Konfigurieren Sie die **Ausführungsoptionen** entsprechend der folgenden Abbildung.

b. Wählen Sie **Fensteroptionen**, und wählen Sie die in der folgenden Abbildung gezeigten Einstellungen.



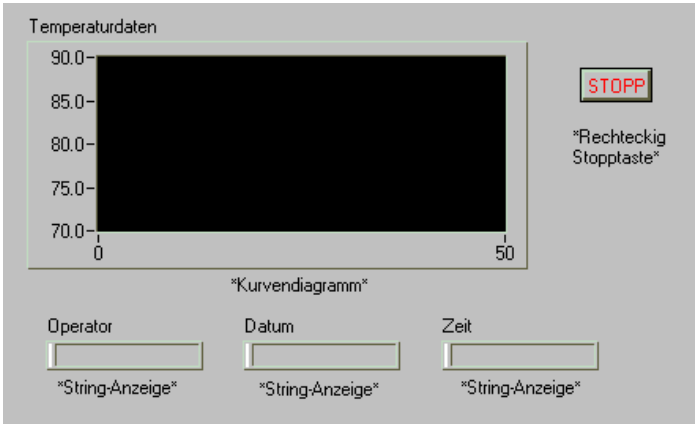
8. Verändern Sie die Größe des Frontpanels, nachdem Sie die Auswahl der Optionen unter **VI-Einstellungen** abgeschlossen haben. Führen Sie die Änderungen entsprechend der folgenden Abbildung durch, so daß die drei String-Anzeigen nicht mehr sichtbar sind.



9. Speichern Sie das VI, und schließen Sie es. Sie können dieses VI nun als SubVI verwenden.

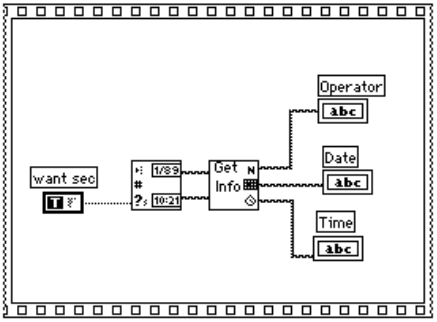
Frontpanel

10. Öffnen Sie ein neues Frontpanel.
11. Plazieren Sie ein Kurvendiagramm (**Elemente»Graph**) auf dem Frontpanel, und geben Sie ihm die Bezeichnung `Temperaturdaten`.
12. Ändern Sie die Skalierung für das Diagramm, so daß als Obergrenze der Wert `90,0` und als Untergrenze der Wert `70,0` verwendet wird. Rufen Sie das Popup-Menü für das Diagramm auf, und wählen Sie **Anzeigen»Legende**, um die Legende auszublenden. Rufen Sie das Popup-Menü für das Diagramm noch einmal auf, und wählen Sie **Anzeigen»Palette**, um die Palette auszublenden.
13. Stellen Sie das Frontpanel entsprechend der folgenden Abbildung fertig.



Blockdiagramm

14. Erstellen Sie eine Sequenzstruktur, und fügen Sie die Objekte, die in der folgenden Abbildung gezeigt werden, in Rahmen 0 ein.



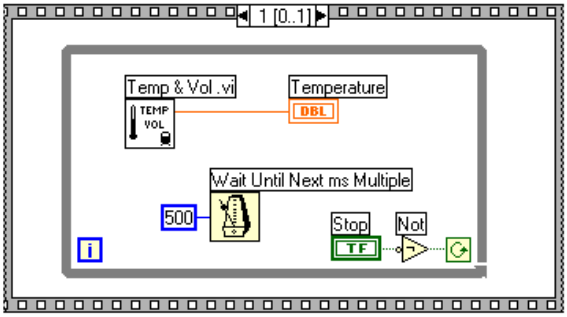
Datum-/Zeit-String bekommen (**Funktionen»Zeit & Dialog**)—Gibt die aktuelle Zeit und das aktuelle Datum aus.

VI Benutzer Infos bekommen (**Funktionen»VI auswählen…** im Verzeichnis `LabVIEW\Activity`)—Öffnet das Frontpanel, und fordert den Benutzer zur Eingabe eines Namens, des Datums und der Uhrzeit auf.

Boolesche Konstante (**Funktionen»Boolesch**)—Kontrolliert, ob TRUE als Wert für die Eingabedatum- und Eingabezeit-Strings vorhanden ist. Klicken Sie mit dem Bedienwerkzeug auf die Konstante, um diese Option auf TRUE einzustellen.

15. Rufen Sie das Popup-Menü für die Sequenzstruktur auf, und wählen Sie im Popup-Menü die Option **Rahmen danach einfügen**.

16. Plazieren Sie eine While-Schleife in den Rahmen 1 der Sequenzstruktur.
17. Fügen Sie die in der folgenden Abbildung gezeigten Objekte ein.



Temp & Vol-VI (**Funktionen»VI auswählen…** im Verzeichnis `LabVIEW\Activity`)—Gibt eine einzelne Temperaturmessung eines simulierten Temperatursensors aus.

Wartet bis zum nächsten Vielfachen von ms (**Funktionen» Zeit & Dialog**)—Bewirkt, daß die While-Schleife in ms ausgeführt wird.

Numerische Konstante (**Funktionen»Numerisch**)—Sie können auch das Popup-Menü für die Funktion “Wartet bis zum nächsten Vielfachen von ms” aufrufen und **Konstante erstellen** wählen, um die numerische Konstante automatisch zu erstellen und zu verbinden. Durch die numerische Konstante wird die Ausführung der Schleife um 500 ms (0,5 Sekunden) verzögert.

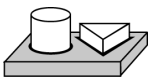
Nicht (**Funktionen»Boolesch**)—Kehrt den Wert der **STOPP**-Schaltfläche um, so daß die While-Schleife so lange ausgeführt wird, bis Sie auf **STOPP** klicken.

18. Speichern Sie das VI unter der Bezeichnung `Popup-Feld-Demo.vi` im Verzeichnis `LabVIEW\Activity`.

19. Führen Sie das VI aus. Das Frontpanel des Benutzer-Info-VIs wird eingeblendet, und Sie werden aufgefordert, Ihren Namen, das Datum und die Uhrzeit einzugeben. Klicken Sie auf die Schaltfläche **Fortsetzen**, um zum aufrufenden VI zurückzukehren. Danach werden Temperaturdaten so lange erfaßt, bis Sie auf die Schalfläche **STOPP** klicken.

☞ **Hinweis** ***Das Frontpanel des VIs "Benutzer-Infos bekommen" wird aufgrund der Optionen eingeblendet, die Sie im Dialogfeld "VI-Einstellungen" gewählt haben. Versuchen Sie nicht, das Frontpanel des SubVIs über das Blockdiagramm des Popup-Feld-Demo-VIs zu öffnen.***

20. Schließen Sie alle Fenster.



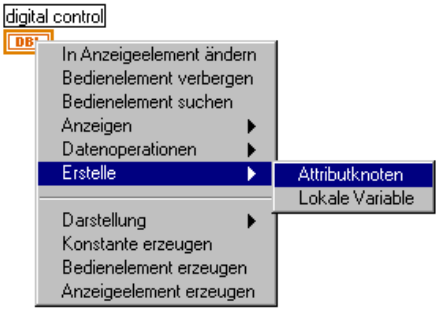
Ende der Übung 26-1.

27

Attribute der Frontpanel-Objekte

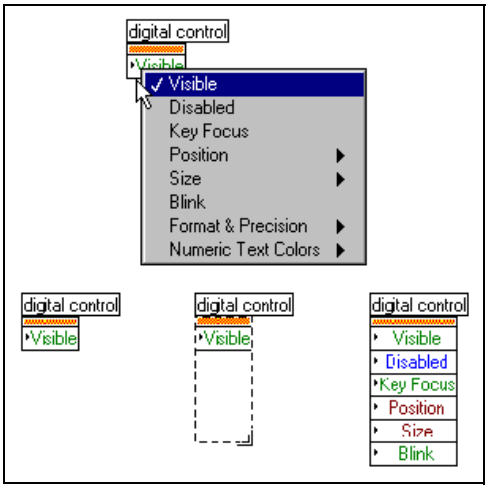
In diesem Kapitel werden Objekte beschrieben, die als Attributknoten bezeichnet werden. Sie stellen spezielle Blockdiagrammknoten dar, durch die das Erscheinungsbild und die funktionalen Charakteristiken von Bedien- und Anzeigeelementen gesteuert werden.

Mit Hilfe dieser Attributknoten können Sie eine Reihe von Attributen einstellen, z.B. die Anzeigefarben, den Ein- bzw. Ausblendestatus, die Position, Blinkfunktionen und viele andere Attribute. Sie können einen Attributknoten erstellen, indem Sie **Erstellen»Attributknoten** im Popup-Menü für das Frontpanel-Objekt oder über das Terminal im Blockdiagramm wählen. Die folgende Abbildung illustriert diesen Vorgang.



Zu Beginn weist der Attributknoten nur eine einzelne Charakteristik auf. Sie können den Knoten erweitern, damit mehrere Charakteristiken angezeigt werden. Wählen Sie dazu den Attributknoten mit dem Positionierwerkzeug aus. Plazieren Sie den Cursor über der unteren rechten Ecke des Knotens, und ziehen Sie den Cursor über die gewünschten Charakteristiken, sobald er sich in einen Rahmen verwandelt hat. Sie können anschließend die Attribute ändern, indem Sie wie in der folgenden

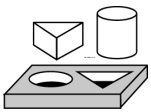
Abbildung mit dem Bedienwerkzeug auf den Knoten klicken und das neue Attribut im Popup-Menü wählen.



Es stehen viele verschiedene Attribute für Frontpanel-Objekte zur Verfügung. Zur Erleichterung der Auswahl können Sie im Hilfefenster die Beschreibungen, Datentypen und akzeptablen Werte von Attributen anzeigen. Wählen Sie **Hilfe»Hilfe anzeigen**, um auf das Hilfefenster zuzugreifen.

Im Abschnitt *Hilfe bekommen* in Kapitel 1, *Einführung in die Programmierung in G*, im *Referenzhandbuch zur Programmierung in G* finden Sie weitere Informationen über den Zugriff auf die Hilfe in LabVIEW.

Mit Attributknoten können Sie Charakteristiken festlegen oder den aktuellen Stand eines Attributs ermitteln. Rufen Sie dazu das Popup-Menü für das Attribut auf, und wählen Sie **In Lesen ändern**.

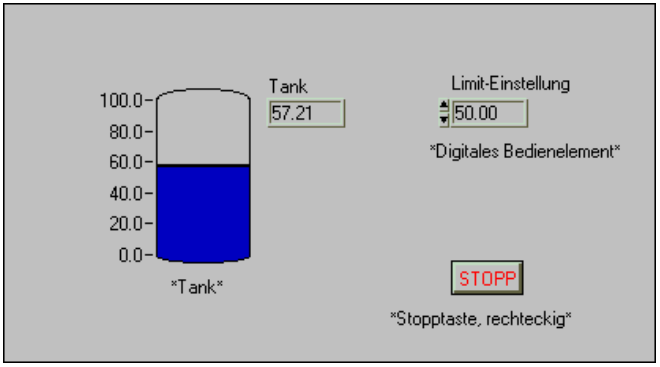


Übung 27-1. Attributknoten verwenden

Übungsziel ist das Erstellen eines VIs, das mit Hilfe von Attributknoten anzeigt, daß eine Obergrenze erreicht wurde. Sie verwenden dabei das Attribut* Füllfarbe *einer* `Tank-Anzeige`*, um festzustellen, ob ein willkürlich generiertes Tankniveau die benutzerdefinierte Grenze überschritten hat.

Frontpanel

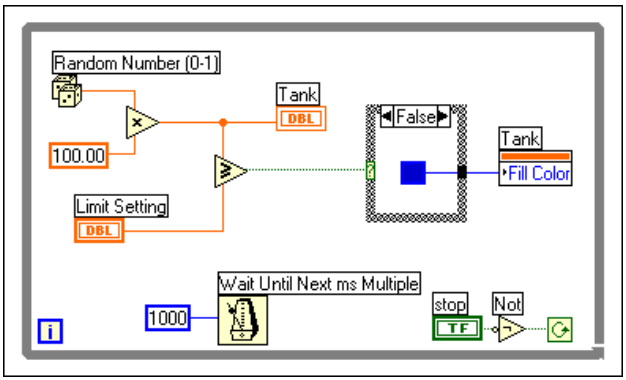
1. Öffnen Sie ein neues Frontpanel, und gestalten Sie es entsprechend der folgenden Abbildung.



2. Ändern Sie die Anzeigeskalierung für den Tank, so daß Werte zwischen `0,0` und `100,0` erfaßt werden.
3. Legen Sie als standardmäßigen **Grenzwert** den Wert `50,00` fest.

Blockdiagramm

4. Erstellen Sie das folgende Blockdiagramm.



Nicht (**Funktionen»Boolesch**)—In dieser Übung kehrt die Funktion "Nicht" den Wert der **STOPP**-Schaltfläche um, so daß die While-Schleife so lange ausgeführt wird, bis Sie auf die **STOPP**-Schaltfläche klicken. (Die Standardeinstellung dieser Schaltfläche ist FALSE.)

Zufallszahl-Generator (**Funktionen»Numerisch**)—Generiert Rohdaten zwischen 0 und 1, um den Tank auf dem Frontpanel zu füllen. Durch Multiplikation dieses Werts mit 100 erstellen Sie einen Wert zwischen 0 und 100.

Größer oder gleich? (**Funktionen»Vergleichsoperationen**)—Vergleicht die Rohdaten mit der **Grenzwert**-Eingabe. Wenn der Wert größer als der Grenzwert oder mit ihm identisch ist, wird der Wert TRUE an die Case-Struktur weitergegeben.

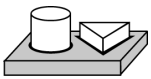
Attributknoten (Popup-Menü für das Tank-Terminal)—Wählen Sie die Option **Erstellen»Attributknoten** im Tank-Terminal. Rufen Sie das Popup-Menü für das Attribut auf, und wählen Sie **Auswählen»Füllfarbe**.

Farbfeldkonstante (**Funktionen»Numerisch»Zusätzliche Num. Konstanten**)—Stellen Sie zwischen dieser Konstante und dem Attribut **Füllfarbe** eine Verbindung her, so daß TRUE durch eine rote Verbindung und FALSE durch eine blaue Verbindung definiert ist. Klicken Sie mit dem Bedienwerkzeug auf die Konstante, um die Farbe auszuwählen.

Wartet bis zum nächsten Vielfachen von ms (**Funktionen» Zeit & Dialog**)—Stellen Sie eine Verbindung zu einer numerischen

Konstante mit dem Wert 1000 her, damit die Schleife einmal pro Sekunde ausgeführt wird.

5. Führen Sie das VI aus. Das Tankniveau wird mit dem **Grenzwert**-Bedienelement verglichen. Wenn der Tankwert größer als die **Grenzwert**-Einstellung oder mit ihr identisch ist, wird der Tank in roter Farbe angezeigt. Wenn die Daten unter den Grenzwert sinken, erscheint der Tank blau.
6. Speichern Sie das VI unter der Bezeichnung `Tankgrenze.vi` im Verzeichnis `LabVIEW\Activity`.



Ende der Übung 27-1.

28

Programmdesign

Nachdem Sie nun mit vielen Aspekten des Programmierens in G vertraut sind, können Sie Ihre Kenntnisse zur Entwicklung Ihrer eigenen Programme einsetzen. Dieses Kapitel enthält einige methodische Vorschläge für das Erstellen von Programmen sowie Empfehlungen für den Programmierstil.

Top-Down-Design

Es ist ratsam, große Projekte im *Top-Down-Design* zu gestalten. Gegenüber anderen Programmiersprachen bietet G beim Top-Down-Design den Vorteil, daß Sie mit der letzten Benutzerschnittstelle beginnen und sie dann animieren können.

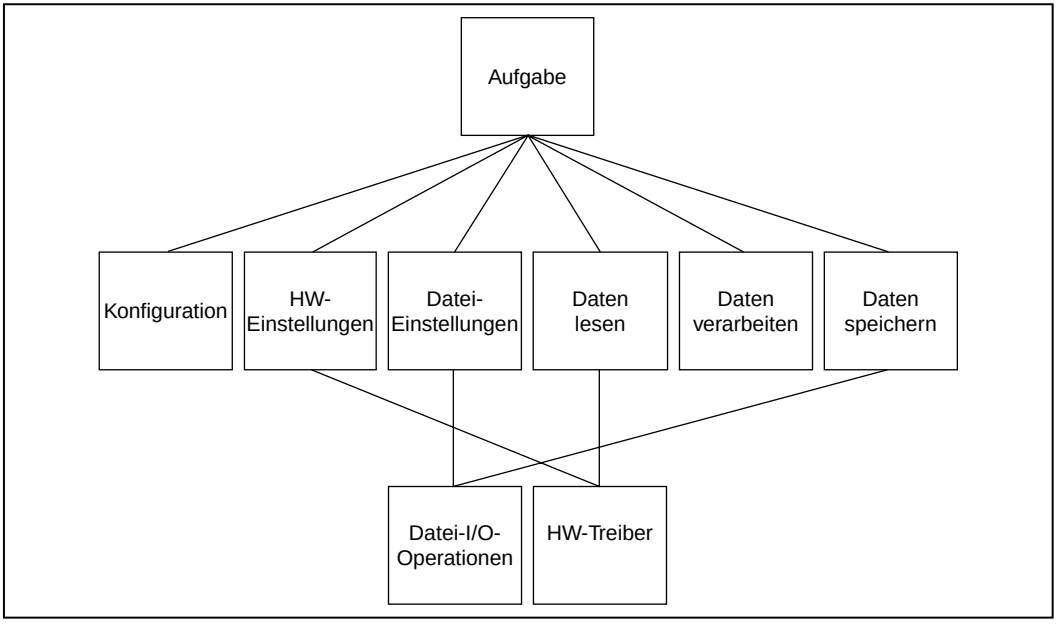
Fertigen Sie eine Liste der Benutzeranforderungen an

Erstellen Sie eine Liste der für einen Benutzer verfügbaren Panels, der Anzahl und des Typs der Bedien- und Anzeigeelemente für diese Panels, der erforderlichen Echtzeit-Analysen bzw. Datenpräsentationen usw. Stellen Sie danach Probe-Frontpanels zusammen, die Sie einem zukünftigen Benutzer zeigen können (oder die Sie selbst als Benutzer manipulieren können). Denken Sie darüber nach, welche Funktionen und Programmerkmale wichtig sind, und besprechen Sie diese Aspekte mit den Benutzern. Setzen Sie dieses interaktive Verfahren dazu ein, die Benutzeroberfläche den jeweiligen Anforderungen entsprechend umzugestalten. Unter Umständen müssen Sie schon auf dieser frühen Stufe einige Untersuchungen auf Maschinenebene durchführen, um sicherzustellen, daß Sie die notwendigen technischen Erfordernisse erfüllen können.

Bestimmen Sie das Design der VI-Hierarchie

Die Leistungsfähigkeit von G liegt in der hierarchischen Struktur von VIs. Nachdem Sie ein VI erstellt haben, können Sie es als SubVI im Blockdiagramm eines übergeordneten VIs verwenden. Dabei können Sie eine nahezu unbegrenzte Anzahl von Hierarchieebenen einrichten.

Teilen Sie die jeweilige Aufgabe in leicht überschaubare, logische Teilaufgaben auf. Wie das folgende Flußdiagramm illustriert, können Sie davon ausgehen, daß Sie für jedes Datenerfassungssystem mehrere größere Blöcke der einen oder anderen Art benötigen.



In manchen Fällen können Sie auf einige dieser Blöcke verzichten, oder Sie benötigen möglicherweise andere Blöcke. Manche Anwendungen schließen z.B. keine Datei-I/O-Operationen ein. Bei anderen Anwendungen benötigen Sie dagegen zusätzliche Blöcke, z.B. zum Darstellen von Eingabeaufforderungen. Ihr Hauptanliegen sollte darin bestehen, die Programmieraufgabe in High-Level-Blöcke zu unterteilen, die Sie leicht verwalten können.

Wenn Sie die übergeordneten Blöcke, die Sie benötigen, bestimmt haben, können Sie versuchen, ein Blockdiagramm zusammenzustellen, in dem Sie diese High-Level-Blöcke einsetzen. Erstellen Sie für jeden Block ein neues *VI-Gerüst* (ein arbeitsunfähiger Prototyp eines späteren SubVIs). Richten

Sie für dieses VI-Gerüst sowohl ein Icon als auch ein Frontpanel ein, das die notwendigen Ein- und Ausgänge enthält. Sie müssen für dieses VI noch kein Blockdiagramm anfertigen. Stellen Sie stattdessen fest, ob dieses VI-Gerüst ein notwendiger Bestandteil Ihres Top-Level-Blockdiagramms ist.

Nachdem Sie eine Gruppe von VI-Gerüsten zusammengestellt haben, sollten Sie sich unter allgemeinen Gesichtspunkten vergegenwärtigen, welche Funktion jeder Block hat und auf welche Weise jeder Block die gewünschten Ergebnisse liefert. Denken Sie darüber nach, ob ein einzelner Block die Informationen erzeugt, die ein nachfolgendes VI benötigt. Falls dies zutrifft, sollten Sie sich vergewissern, daß die Skizze für Ihr Top-Level-Blockdiagramm Verbindungen aufweist, über die Daten zwischen VIs übertragen werden können.

Versuchen Sie dabei, unnötige globale Variablen zu vermeiden, da sie die Datenabhängigkeit zwischen VIs verdecken. Je größer Ihr System wird, desto schwieriger wird das Debugging, falls Sie sich auf globale Variablen zur Übertragung von Informationen zwischen VIs verlassen.

Erstellen Sie das Programm

Sie können nun das Programm in G erstellen:

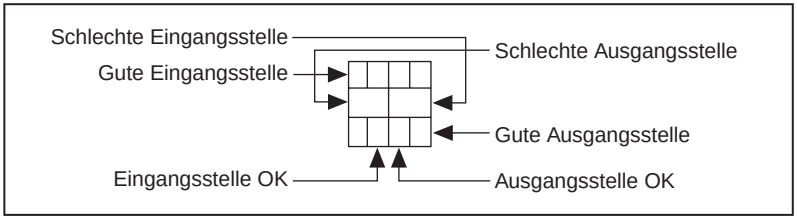
- Verfolgen Sie einen modularen Ansatz, indem Sie SubVIs erstellen, wo sich eine logische Arbeitstrennung ergibt oder wo die Möglichkeit besteht, Code mehrmals zu verwenden.
- Lösen Sie gleichzeitig Ihre allgemeinen und speziellen Probleme.
- Testen Sie die SubVIs schon beim Erstellen. Unter Umständen müssen Sie dazu zwar übergeordnete Testroutinen entwickeln; es ist jedoch einfacher, Fehler in einem kleinen Modul zu ermitteln als in einer Hierarchie aus mehreren VIs.

Bei der detaillierten Durchsicht der SubVIs stellen Sie möglicherweise fest, daß das ursprüngliche Design unvollständig ist. Eventuell fällt Ihnen z.B. auf, daß mehr Informationen von einem SubVI zu einem anderen SubVI übertragen werden müssen. An dieser Stelle ist darum unter Umständen eine Überprüfung des Top-Level-Designs notwendig. Bei Verwendung modularer SubVIs zur Ausführung spezieller Aufgaben läßt sich eine solche Neuorganisation eines Programms wesentlich leichter durchführen.

Planen Sie Ihre Anschlußfelder im voraus

Wenn Sie davon ausgehen, daß Sie zu einem späteren Zeitpunkt zusätzliche Ein- oder Ausgänge ergänzen müssen, sollten Sie ein Anschlußfeldmuster mit zusätzlichen Terminals wählen. Durch diese zusätzlichen Terminals, die Sie zunächst unverbunden lassen können, ersparen Sie sich eine andernfalls notwendige Änderung des Anschlußfelds für das VI, falls Sie später feststellen, daß Sie einen weiteren Ein- oder Ausgang benötigen. Aufgrund dieser Flexibilität können Sie Änderungen mit minimalen Auswirkungen für die Hierarchie durchführen.

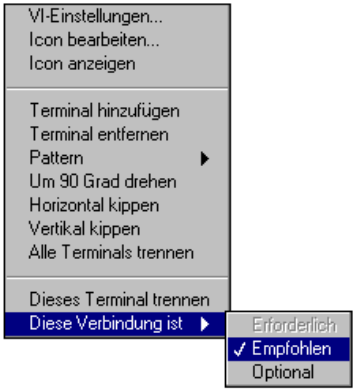
Beim Verbinden von Bedien- und Anzeigeelementen mit dem Anschluß sollten Sie darauf achten, daß Sie Eingänge auf der linken Seite und Ausgänge auf der rechten Seite plazieren. Auf diese Weise vermeiden Sie komplizierte, unübersichtliche Verbindungsmuster in Ihren VIs.



Wenn Sie eine Gruppe aus SubVIs erstellen, die oft zusammen verwendet werden, sollten Sie versuchen, die SubVIs mit einem übereinstimmenden Anschlußfeld zu versehen, bei dem sich die gemeinsamen Anschlüsse an der gleichen Stelle befinden. Sie können sich auf diese Weise leichter merken, wo die einzelnen Eingänge liegen, ohne daß Sie im Hilfefenster nachsehen müssen. Wenn Sie ein SubVI erstellen, dessen Ausgabewerte als Eingaben für ein anderes SubVI verwendet werden, sollten Sie versuchen, die Eingangs- und Ausgangsverbindungen aufeinander auszurichten. Diese Technik ermöglicht eine Vereinfachung des Verbindungsmusters.

SubVIs mit erforderlichen Eingaben

Sie können die erforderlichen Eingaben für SubVIs auf dem Frontpanel bearbeiten, indem Sie auf das Icon-Feld in der oberen rechten Ecke des Fensters klicken und **Anschluß anzeigen»Diese Verbindung ist** wählen. Wählen Sie dann im Untermenü die Option **Erforderlich**, **Empfohlen** oder **Optional**. Die folgende Abbildung zeigt die im Untermenü verfügbaren Optionen.



Sie können zum Icon-Feld im Frontpanel zurückkehren, indem Sie das Popup-Menü für das Anschlußfeld aufrufen und **Icon anzeigen** wählen.

Guter Diagrammstil

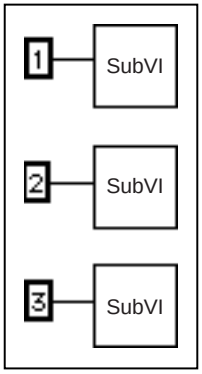
Sie sollten generell vermeiden, Blockdiagramme zu erstellen, die mehr als ein oder zwei Bildschirmflächen einnehmen. Bei Diagrammen, die sehr groß werden, sollten Sie darüber nachdenken, ob einige Komponenten des Diagramms auch in anderen VIs verwendet werden können oder ob ein bestimmter Abschnitt des Diagramms als logische Komponente zusammengefaßt werden kann. In diesen Fällen könnte es von Vorteil sein, das Diagramm in SubVIs zu unterteilen.

Weitsichtiges Denken und sorgfältiges Planen erleichtern das Design von Diagrammen, in denen spezielle Aufgaben durch SubVIs erfüllt werden. Durch Verwendung von SubVIs sind Sie in der Lage, einzelne Änderungen und das Debugging Ihrer Diagramme schnell durchzuführen. Die Funktion eines gut strukturierten Programms läßt sich bereits nach einer kurzen Überprüfung leicht bestimmen.

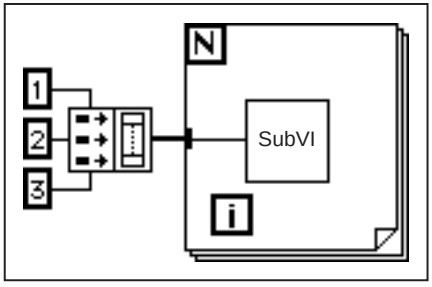
Identifizieren Sie häufig wiederholte Operationen

Eventuell stellen Sie beim Design von Programmen fest, daß Sie eine bestimmte Operation wiederholt ausführen. Je nach Situation sollten Sie in diesem Fall erwägen, SubVIs oder Schleifen zur wiederholten Ausführung zu verwenden.

Überprüfen Sie z.B. das folgende Diagramm, in dem drei ähnliche Vorgänge unabhängig voneinander ausgeführt werden.



Als Alternative zu diesem Design bietet sich eine Schleife an, die den Vorgang dreimal ausführt. Sie können ein Array der verschiedenen Argumente zusammenstellen und durch Auto-Indizierung den korrekten Wert für jede Schleifeniteration einstellen.



Wenn es sich um konstante Array-Elemente handelt, können Sie eine Array-Konstante verwenden, anstatt das Array auf dem Blockdiagramm zu erstellen.

Verwenden Sie von links nach rechts ausgerichtete Layouts

G wurde für Layouts entwickelt, die von links nach rechts (und gelegentlich von oben nach unten) verlaufen. Organisieren Sie, wenn möglich, alle Elemente Ihres Programms in einem entsprechenden Layout.

Suchen Sie nach Fehlern

Bei allen I/O-Operationen sollten Sie damit rechnen, daß Fehler auftreten können. Nahezu alle I/O-Funktionen geben Informationen über Fehler aus. Falls Sie direkte I/O-Operationen durchführen, sollten Sie darauf achten, daß das Programm Fehler feststellen kann und daß Sie diese Fehler in angemessener Weise behandeln.

LabVIEW bearbeitet Fehler nicht automatisch, da verschiedene Benutzer normalerweise unterschiedliche Fehlerbehandlungsmethoden anwenden wollen. Wenn beispielsweise eine Zeitbegrenzung für ein I/O-VI in einem Blockdiagramm eintritt, ziehen es manche Benutzer vor, das Programm sofort anzuhalten; andere Benutzer bevorzugen es jedoch, wenn das VI über einen bestimmten Zeitraum weitere Versuche durchführt.
In LabVIEW können Sie solche Entscheidungen über die Art der Fehlerbehandlung in das Blockdiagramm des VIs integrieren.

In der folgenden Liste werden einige Situationen beschrieben, in denen Fehler häufig auftreten:

- Falsche Initialisierung der Kommunikation oder der Daten, die auf unsachgemäße Weise an ein externes Gerät geschrieben wurden.
- Stromausfall in einem externen Gerät oder ein beschädigtes oder nicht ordnungsgemäß arbeitendes externes Gerät.
- Änderung der Funktionalität in einer Anwendung oder Bibliothek beim Aktualisieren der Betriebssystemsoftware.

Es sind Situationen möglich, in denen Sie bei Auftreten eines Fehlers vermeiden wollen, daß nachfolgende Operationen durchgeführt werden. Wenn z.B. eine analoge Ausgabeoperation versagt, da Sie das falsche Gerät angegeben haben, wollen Sie möglicherweise eine nachfolgende analoge Eingabeoperation verhindern.

Eine Methode zur Behandlung eines solchen Problems besteht darin, nach jeder Funktion einen Fehlertest durchzuführen und nachfolgende Funktionen in Case-Strukturen unterzubringen. Diese Methode kann jedoch bewirken, daß Ihre Diagramme sehr kompliziert werden und daß schließlich der Zweck Ihrer Anwendung nicht mehr erkennbar ist.

Ein alternativer Ansatz, der in einer Reihe von Anwendungen und in vielen der VI-Bibliotheken erfolgreich angewendet wurde, besteht darin, die Fehlerbehandlung in die SubVIs zu integrieren, die I/O-Operationen durchführen. Jedes VI kann mit einem Fehlereingang und einem Fehlerausgang ausgestattet werden. Ein VI kann so gestaltet werden, daß es den Fehlereingang auf zuvor aufgetretene Fehler überprüft. Sie können

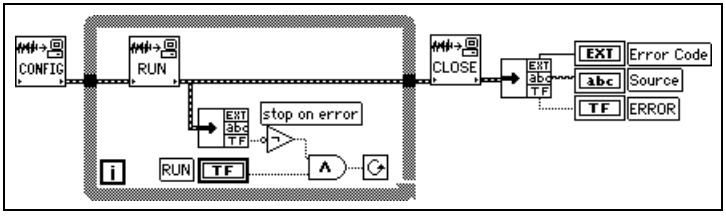
das VI so konfigurieren, daß bei vorhandenen Fehlern nichts geschieht, außer daß die Fehlereingabe an den Fehlerausgang weitergeleitet wird. Wenn kein Fehler vorliegt, kann das VI die Operation ausführen und das Ergebnis an den Fehlerausgang weiterleiten.

☞ **Hinweis** ***In manchen Fällen, z.B. bei einem Schließvorgang, ist es unter Umständen sinnvoll, daß das VI den Vorgang unabhängig vom eventuell weitergeleiteten Fehler durchführt.***

Die beschriebene Methode macht es möglich, mehrere VIs so miteinander zu verbinden, daß über die miteinander verbundenen Fehlereingänge und -ausgänge Fehlerinformationen von einem VI an das nächste weitergeleitet werden. Am Ende der VI-Reihe können Sie das VI “Einfacher Error-Handler” dazu verwenden, ein Dialogfeld anzuzeigen, falls ein Fehler vorhanden ist. Sie können auf das VI “Einfacher Error-Handler” über die Optionen **Funktionen»Zeit & Dialog** zugreifen. Außer zur Fehlerbehandlung eignet sich diese Methode auch dazu, die Reihenfolge mehrerer I/O-Operationen festzulegen.

Einer der Hauptvorteile bei der Verwendung von Fehlereingangs- und Fehlerausgangs-Clustern besteht darin, daß Sie mit ihrer Hilfe die Reihenfolge der Ausführung verschiedener Operationen, die keine Ähnlichkeit miteinander haben, steuern können.

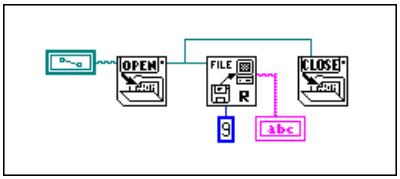
Die Fehlerinformationen werden im allgemeinen durch ein Cluster repräsentiert, das einen numerischen Fehlercode enthält, d.h. eine Zeichenfolge, die den Namen der Funktion, die den Fehler verursacht hat, und einen Booleschen Fehlerwert zum schnellen Testen enthält. Die folgende Abbildung zeigt, wie Sie diese Methode in Ihren eigenen Anwendungen verwenden können. Beachten Sie, daß die While-Schleife angehalten wird, wenn ein Fehler festgestellt wird.



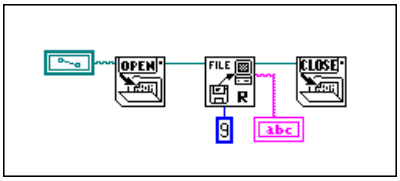
Vermeiden Sie fehlende Abhängigkeiten

Vergewissern Sie sich, daß Sie die Ereignisfolge ausdrücklich definiert haben, wo es notwendig ist. Gehen Sie nicht davon aus, daß die Ausführung von links nach rechts oder von oben nach unten erfolgt, wenn keine Datenabhängigkeit vorhanden ist.

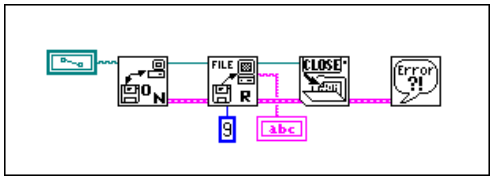
Im folgenden Beispiel existiert keine Abhängigkeit zwischen dem VI “Datei lesen” und dem VI “Datei schließen”. Dieses Programm arbeitet unter Umständen anders als erwartet.



In der folgenden Version des Blockdiagramms sorgt die Verbindung zwischen einem Ausgang des Lese-VIs und dem Schließen-VI dafür, daß eine Abhängigkeit besteht. Der Vorgang kann erst dann beendet werden, wenn das Schließen-VI den Ausgabewert des Lesen-VIs erhält.



Beachten Sie, daß auch das obige Beispiel keine Fehlerüberprüfung durchführt. Wenn die Datei z.B. nicht existiert, zeigt das Programm keine Warnung an. Die folgende Version des Blockdiagramms illustriert eine mögliche Methode, dieses Problem zu beheben. In diesem Beispiel verwendet das Blockdiagramm die Fehler-I/O-Eingaben und -Ausgaben dieser Funktionen, um alle Fehler an das VI “Einfacher Error-Handler” zu übermitteln.



Vermeiden Sie die übermäßige Verwendung von Sequenzstrukturen

Da die VIs eine Menge von inhärent parallelen Operationen verarbeiten können, sollten Sie es vermeiden, Sequenzstrukturen zu verwenden. Durch den Einsatz einer Sequenzstruktur kann zwar garantiert werden, daß die Ausführung in einer bestimmten Reihenfolge erfolgt; aber andererseits werden dadurch parallele Operationen verhindert. Asynchrone Aufgaben, die I/O-Geräte (GPIB, serielle Ports und DAQ-Einsteckkarten) verwenden, können beispielsweise verzahnt mit anderen Operationen ausgeführt werden, wenn sie nicht durch Sequenzstrukturen daran gehindert werden.

Sequenzstrukturen neigen dazu, Teile des Programms zu verdecken, und unterbrechen dadurch den natürlichen Datenfluß von links nach rechts. Durch Verwendung von Sequenzstrukturen wird zwar die Leistung nicht beeinträchtigt. Wenn Sie jedoch einzelne Operationen in einer Sequenz anordnen müssen, kann es sinnvoller sein, wenn Sie anstelle von Sequenzstrukturen für den entsprechenden Datenfluß sorgen. Bei I/O-Operationen können Sie z.B. die zuvor beschriebene Fehler-I/O-Methode anwenden, um sicherzustellen, daß die einzelnen I/O-Operationen in der notwendigen Reihenfolge ablaufen.

Lernen Sie aus den Beispielen

Wenn Sie weitere Informationen zum Programmdesign benötigen, können Sie die vielen Blockdiagrammbeispiele untersuchen, die in LabVIEW enthalten sind. Diese Beispielprogramme verschaffen Ihnen weitere Einblicke in den Stil und die Methodik des Programmierens in G. Sie können dieses Blockdiagramme einsehen, indem Sie die VIs im Verzeichnis `Examples` öffnen.

29

Weiteres Vorgehen

Sie haben eine Reihe von Übungen bewältigt, die Sie in die Lage versetzen, eigene LabVIEW-Anwendungen zu erstellen. Bevor Sie jedoch die Arbeit an Ihren eigenen Anwendungen beginnen, sollten Sie sich eventuell damit vertraut machen, welche zusätzlichen Ressourcen Ihnen zur Verfügung stehen.

Andere nützliche Ressourcen

Die folgenden Abschnitte geben Ihnen einen Überblick über verschiedene Ressourcen, die Ihnen dabei helfen können, problemfreie Anwendungen zu erstellen.

Die Optionen "Solution Wizard" und "Beispiele suchen"

Wenn Sie Beispiele sehen möchten, die Ihrer Anwendung ähneln, können Sie die Optionen **Solution Wizard** und **Beispiele suchen** im LabVIEW-Dialogfeld wählen. Der **Solution Wizard** erstellt aufgrund der von Ihnen angegebenen Kriterien Beispiele für die Datenerfassung und für Geräte-I/O-Vorgänge. Durch die Option **Beispiele suchen** werden Beispiele geöffnet, die mehrere G-Programmierkonzepte sowie Analysevorgänge, das Arbeiten im Netzwerk, die Datenerfassung und Geräte-I/O-Vorgänge illustrieren.

Anwendungen zur Datenerfassung

(Windows, Macintosh) Das LabVIEW-Handbuch *Grundlagen der Datenerfassung* enthält Informationen zum Starten von Datenerfassungsanwendungen, die analoge Eingaben, analoge Ausgaben, digitale I/O-Vorgänge und Counter/Timer verwenden. Das LabVIEW-Handbuch *Grundlagen der Datenerfassung* behandelt außerdem die Grundkonzepte der Datenerfassung und die VIs, die zum Implementieren dieser Konzepte eingesetzt werden.

Programmiermethoden in G

In Teil I, *Einführung in das Programmieren in G*, und Teil V, *Fortgeschrittenes Programmieren in G*, in diesem Handbuch haben Sie eine Einführung in grundsätzliche Methoden des Programmierens in G erhalten. Weiterführende Hinweise zu den Fähigkeiten von G finden Sie im *Referenzhandbuch zur Programmierung in G*. Dort erhalten Sie Informationen zur Ausführung und zum Debugging, zur Einrichtung von VIs, zu den Frontpanel-Objekten, zu Verknüpfungen, Strukturen und Attributknoten. Außerdem enthält dieses Handbuch auch Informationen, die nicht im *LabVIEW Benutzerhandbuch* besprochen werden, so u.a. Informationen zum Drucken, zum Anpassen der G-Umgebung mit Hilfe von **Voreinstellungen**, zu benutzerdefinierten Bedienelementen, zum Multithreading und zu Leistungsfragen.

Funktionen- und VI-Referenz

Sie finden einen Überblick über die in LabVIEW verfügbaren Funktionen und VIs im *LabVIEW Funktionen- und VI-Referenzhandbuch.* Dieses Handbuch bietet kurze Beschreibungen der Funktionen und der VIs in derselben Reihenfolge, in der sie in der Palette **Funktionen** erscheinen.

Ressourcen für fortgeschrittene Funktionen

Das *LabVIEW Benutzerhandbuch* macht Sie mit den grundsätzlichen Aspekten des Erstellens einer LabVIEW-Anwendung vertraut. LabVIEW bietet eine Reihe von fortgeschrittenen Funktionen, die in diesem Handbuch entweder nicht besprochen oder nur am Rande erwähnt werden. Die folgenden Abschnitte geben Ihnen einen Überblick über diese Funktionen und weisen auf zusätzliche Ressourcen hin, auf die Sie, wenn nötig, in Ihren Anwendungen zurückgreifen können.

Attributknoten

In Kapitel 27, *Attribute der Frontpanel-Objekte*, in diesem Handbuch finden Sie eine kurze Beschreibung von Attributknoten. Mit Hilfe von Attributknoten können Sie die Einstellungen, die sich auf Bedien- und Anzeigeelemente auswirken, über das Programm verwalten. Sie können z.B. entscheiden, ob bestimmte Bedienelemente sichtbar sein sollen. Verwenden Sie einen Attributknoten, wenn Sie die Optionen in einem Ring- oder Listenbedienelement über das Programm ändern, den Inhalt eines Diagramms löschen oder die Skalierung eines Diagramms oder eines Graphen ändern müssen. In Kapitel 22, *Attributknoten*, im *Referenzhandbuch zur Programmierung in G* werden Attributknoten detailliert beschrieben.

Einrichtung und Voreinstellungen von VIs

Sie können verschiedene VI-Attribute sowie LabVIEW-Attribute mit Hilfe der VI-Server-Funktion über das Programm steuern. Sie können VIs in den Arbeitsspeicher laden, VIs ausführen und das Aussehen von VIs sowie das Verhalten eines VIs beim Ausführen ändern. Einige der Optionen, die Sie interaktiv über **Voreinstellungen** und **VI-Einstellungen…** festlegen können, können auch über das Programm eingestellt werden. Die Optionen, die über **Voreinstellungen** verfügbar sind, werden als Anwendungseinstellungen behandelt, da sie sich auf alle in LabVIEW angezeigten VIs auswirken. Bei den Optionen unter **VI-Einstellungen…** handelt es sich um VI-Einstellungen, da sie sich nur auf ein VI und auf die Instanzen auswirken, in denen Sie das VI als SubVI einsetzen.

Sie können nicht nur Attribute ändern, die als Eigenschaften bezeichnet werden, sondern Sie können auch bestimmte Aktionen auf LabVIEW oder auf ein einzelnes VI anwenden. Sie können beispielsweise eine Aktion (auch Methode genannt) einrichten, um ein VI zu speichern. Das Einstellen der Eigenschaften und Methoden für LabVIEW und für VIs ist auf dem lokalen Computer, über ein TCP/IP-Netzwerk oder durch eine andere ActiveX-Anwendung möglich. Weitere Informationen zu den Fähigkeiten von ActiveX finden Sie in Kapitel 22, *ActiveX-Unterstützung*, in diesem Handbuch. Weitere Informationen zu den TCP/IP-Einstellungen und zur Verwendung der TCP/IP-Funktion finden Sie in Kapitel 7, *Bearbeiten von VIs*, und in Kapitel 21, *VI-Server*, im *Referenzhandbuch zur Programmierung in G*.

Lokale und globale Variablen

Verwenden Sie lokale Variablen, um Bedienelemente an mehreren Stellen im Blockdiagramm abzulesen. Setzen Sie lokale Variablen auch dann ein, wenn Sie ein Frontpanel-Objekt an manchen Stellen als Bedienelement und an anderen Stellen als Anzeige verwenden möchten. Verwenden Sie nicht zu viele lokale Variablen, da sie den Datenfluß in Ihren Diagrammen verdecken. Dadurch wird es schwieriger, den Zweck eines Programms zu erkennen und das Debugging für lokale Variablen durchzuführen. In Kapitel 23, *Globale und lokale Variablen*, im *Referenzhandbuch zur Programmierung in G* finden Sie Erläuterungen zu lokalen Variablen.

Globale Variablen speichern Daten, die von mehreren VIs verwendet werden. Setzen Sie globale Variablen ebenfalls mit Bedacht ein, da auch sie den Datenfluß in Ihrem Diagramm verdecken. In manchen Anwendungen ist der Einsatz globaler Variablen zwar notwendig. Sie sollten sie jedoch nicht verwenden, wenn Sie Ihr Programm so strukturieren können, daß eine alternative Datenflußmethode zum Übertragen der Daten verwendet wird. Weitere Details finden Sie in Kapitel 23, *Globale und lokale Variablen*, im *Referenzhandbuch zur Programmierung in G*.

SubVIs erstellen

Sie können einen ausgewählten Bereich eines Blockdiagramms in SubVIs umwandeln, indem Sie **Bearbeiten»SubVI aus Auswahl** wählen. Zusätzlich verbindet LabVIEW automatisch die richtigen Eingänge und Ausgänge mit dem SubVI. In manchen Fällen ist das Erstellen eines SubVIs aus einem VI nicht möglich. Eine detaillierte Besprechung dieser Funktion finden Sie in Kapitel 3, *Verwendung von SubVIs*, im *Referenzhandbuch zur Programmierung in G*.

VI-Profile

Sie können die VI-Profil-Funktion (**Projekt»Profilfenster anzeigen**) verwenden, um auf detaillierte Informationen zur Timing-Statistik und zu Timing-Details eines VIs zuzugreifen. Diese Funktion hilft Ihnen dabei, die Leistung Ihrer VIs zu optimieren. Eine detaillierte Besprechung der Profil-Funktion finden Sie in Kapitel 28, *Leistung*, im *Referenzhandbuch zur Programmierung in G*.

Bedienelement-Editor

Verwenden Sie den Bedienelement-Editor, um das Aussehen Ihrer Frontpanel-Bedienelemente Ihren Vorstellungen entsprechend anzupassen. Sie können mit dem Editor auch benutzerdefinierte Bedienelemente speichern, damit Sie sie auch in anderen Anwendungen einsetzen können. Eine detaillierte Besprechung des Bedienelement-Editors finden Sie in Kapitel 24, *Benutzerspezifische Bedienelemente und Typendefinitionen*, im *Referenzhandbuch zur Programmierung in G*.

Listen- und Ring-Bedienelemente

Verwenden Sie Listen- und Ring-Bedienelemente, wenn eine Optionsliste für den Benutzer angezeigt werden soll. Eine detaillierte Besprechung dieser Frontpanel-Objekte finden Sie in Kapitel 13, *Bedien- und Anzeigeelemente vom Typ Liste und Ring*, im *Referenzhandbuch zur Programmierung in G*.

Die Funktion “Aufruf ext. Bibliotheken”

Über die Funktion “Aufruf ext. Bibliotheken” können Sie in LabVIEW eine gemeinsam benutzte Bibliothek oder DLL-Datei aufrufen. Mit dieser Funktion ist es möglich, in LabVIEW eine Schnittstelle zu erstellen, über die ein vorhandener Code oder Treiber aufgerufen wird. Weitere Hinweise zu diesen Funktionen finden Sie im Kapitel 25, *Aufrufen von Code aus anderen Sprachen*, im *Referenzhandbuch zur Programmierung in G*.

Code Interface Nodes

Code Interface Nodes (CINs) bieten eine alternative Methode, Quellcode, der in einer konventionellen textorientierten Programmiersprache geschrieben wurde, von LabVIEW-Blockdiagrammen aus aufzurufen. Verwenden Sie CINs für Aufgaben, die mit konventionellen Programmiersprachen schneller als mit LabVIEW durchgeführt werden können, oder für Aufgaben, die nicht direkt von einem Blockdiagramm aus ausgeführt werden können, sowie zum Verbinden von vorhandenem Code mit LabVIEW. Im allgemeinen ist es jedoch leichter, Quellcode über die Funktion “Aufruf ext. Bibliotheken” aufzurufen, als über CINs. Sie sollten CINs vor allem dann benutzen, wenn eine engere Integration mit LabVIEW und dem Quellcode notwendig ist. Weitere Informationen über CINs finden Sie im *LabVIEW Code Interface Reference Manual*, das nur im übertragbaren Dokumentenformat PDF (Portable Document Format) verfügbar ist, und in Kapitel 25, *Aufrufen von Code aus anderen Sprachen*, im *Referenzhandbuch zur Programmierung in G*.

A

Analyse-Referenzen

In diesem Anhang werden die Referenzmaterialien aufgeführt, die beim Herstellen der Analyse-VIs in diesem Handbuch verwendet wurden. Die genannten Texte enthalten weitere Informationen zu den Theorien und Algorithmen, die in der Analyse-Bibliothek angewendet werden.

Baher, H. *Analog & Digital Signal Processing*. New York: John Wiley & Sons, 1990.

Bates, D.M. und Watts, D.G. *Nonlinear Regression Analysis and its Applications*. New York: John Wiley & Sons, 1988.

Bracewell, R.N. "Numerical Transforms." *Science* 248 (11. Mai 1990).

Burden, R.L. und Faires, J.D. *Numerical Analysis.* 3. Aufl., Boston: Prindle, Weber & Schmidt, 1985.

Chen, C.H. et al. *Signal Processing Handbook*. New York: Marcel Decker, Inc., 1988.

DeGroot, M. *Probability and Statistics. 2. Aufl.*, Reading, Massachusetts: Addison-Wesley Publishing Co., 1986.

Dowdy, S. und Wearden, S. *Statistics for Research*. 2. Aufl., New York: John Wiley & Sons. 1991.

Dudewicz, E.J. und Mishra, S.N. *Modern Mathematical Statistics.* New York: John Wiley & Sons, 1988.

Duhamel, P. et al. "On Computing the Inverse DFT." *IEEE Transactions on ASSP* 34, 1 (Februar 1986).

Dunn, O. und Clark, V. *Applied Statistics: Analysis of Variance and Regression* 2. Aufl., New York: John Wiley & Sons. 1987.

Elliot, D.F. *Handbook of Digital Signal Processing Engineering Applications*. San Diego: Academic Press, 1987.

Golub, G.H. und Van Loan, C.F. *Matrix Computations.* Baltimore: The John Hopkins University Press, 1989.

Harris, Fredric J. “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform.” Proceedings of the IEEE-66 (1978)-1.

Maisel, J.E. “Hilbert Transform Works With Fourier Transforms to Dramatically Lower Sampling Rates.” *Personal Engineering and Instrumentation News* 7, 2 (Februar 1990).

Miller, I. und Freund, J.E. *Probability and Statistics for Engineers*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1987.

Neter, J. et al. *Applied Linear Regression Models*. Richard D. Irwin, Inc., 1983.

Neuvo, Y., Dong, C.-Y., und Mitra, S.K. “Interpolated Finite Impulse Response Filters” *IEEE Transactions on ASSP.* ASSP-32, 6 (Juni 1984).

O’Neill, M.A. “Faster Than Fast Fourier.” *BYTE*. (April 1988).

Oppenheim, A.V. und Schafer, R.W. *Discrete-Time Signal Processing*. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

Parks, T.W. und Burrus, C.S. *Digital Filter Design*. New York: John Wiley & Sons, Inc., 1987.

Pearson, C.E. *Numerical Methods in Engineering and Science*. New York: Van Nostrand Reinhold Co., 1986.

Press, W.H. et al. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1988.

Rabiner, L.R. & Gold, B. *Theory and Application of Digital Signal Processing*. Englewood Cliffs, New Jersey: Prentice Hall, 1975.

Sorensen, H.V. et al. “On Computing the Split-Radix FFT.” *IEEE Transactions on ASSP.* ASSP-34, 1 (Februar 1986).

Sorensen, H.V. et al. “Real-Valued Fast Fourier Transform Algorithms.” *IEEE Transactions on ASSP.* ASSP-35, 6 (Juni 1987).

Spiegel, M. *Schaum’s Outline Series on Theory and Problems of Probability and Statistics*. New York: McGraw-Hill, 1975.

Stoer, J. und Bulirsch, R. *Introduction to Numerical Analysis*. New York: Springer-Verlag, 1987.

Vaidyanathan, P.P. *Multirate Systems and Filter Banks.* Englewood Cliffs, New Jersey: Prentice Hall, 1993.

Wichman, B. und Hill, D. “Building a Random-Number Generator: A Pascal Routine for Very-Long-Cycle Random-Number Sequences.” *BYTE* (März 1987): 127–128.

B

Häufig gestellte Fragen

In diesem Anhang werden häufig gestellte Fragen zur LabVIEW-Netzwerkkommunikation und zum Geräte-I/O, insbesondere zum GPIB und seriellen I/O, beantwortet.

Häufig gestellte Fragen zur Kommunikation

In diesem Abschnitt werden häufig gestellte Fragen zur LabVIEW-Netzwerkkommunikation beantwortet. Die Fragen werden in verschiedenen Abschnitten für die jeweils relevanten Plattformen behandelt: Alle Plattformen, Windows und Macintosh. Wenden Sie sich bitte an National Instruments, falls Sie weitere Fragen haben oder Vorschläge zu LabVIEW machen möchten.

Fragen für alle Plattformen

Wie kann die Kommunikation zwischen LabVIEW und anderen Anwendungen hergestellt werden?

Die Kommunikation mit anderen Anwendungen—oft auch als Interprozeß- oder Interapplikations-Kommunikation bezeichnet—kann auf jeder Plattform über die Standard-Netzwerkprotokolle erfolgen. LabVIEW unterstützt das TCP- (Transmission Control Protocol) und UDP-Protokoll (User Datagram Protocol) auf allen Plattformen.

(Windows) LabVIEW für Windows unterstützt außerdem auch das DDE-Protokoll (Dynamic Data Exchange).

(Macintosh) LabVIEW für Macintosh unterstützt außerdem auch das IAC-Protokoll (Interapplication Communication). IAC schließt die Apple Events- und PPC-Protokolle (Program-to-Program Communication) ein.

(UNIX) LabVIEW für UNIX unterstützt nur die TCP- und UDP-Protokolle.

Teil II, *I/O-Schnittstellen*, enthält weitere Hinweise für die Verwendung von LabVIEW zur Kommunikation mit anderen Anwendungen. In vielen Instrumentierungsanwendungen bietet Datei-I/O eine zusätzliche einfache und angemessene Methode zum Senden von Informationen zwischen Anwendungen.

Wie wird eine andere Anwendung mit LabVIEW gestartet?

Verwenden Sie in Windows und UNIX das VI `System Exec.vi` (**Funktionen»Kommunikation**). Verwenden Sie AESend Finder Open (**Funktionen»Kommunikation»Apple Event**) auf dem Macintosh.

Wann sollte UDP anstelle von TCP verwendet werden?

Normalerweise wird UDP für Anwendungen verwendet, bei denen Zuverlässigkeit nicht absolut wichtig ist. Bei einer Anwendung, die informative Daten häufig genug an eine Zielstelle überträgt, ergeben sich unter Umständen keine Probleme, wenn einige Datensegmente verlorengehen. UDP kann auch für das Broadcasting an alle Computer verwendet werden, die zum Empfang der Daten vom Server bereit sind.

Welche Portnummern können mit TCP und UDP verwendet werden?

Ein Port wird durch eine Zahl zwischen 0 und 65535 repräsentiert. Unter UNIX sind Portnummern unter 1024 für privilegierte Anwendungen (z.B. FTP) reserviert. Wenn Sie einen lokalen Port angeben, können Sie den Wert 0 verwenden, um einen unbenutzten Port zu benutzen oder zu wählen.

Warum ist mit UDP kein Broadcasting möglich?

Da sich die Broadcast-Adresse von Domäne zu Domäne unterscheidet, müssen Sie bei Ihrem Systemadministrator nachfragen, welche Broadcast-Adresse Sie verwenden müssen. Die Broadcast-Adresse 0xFFFFFFFF ist z.B. mit Sicherheit nicht für Ihre Domäne korrekt. Darüber hinaus läßt Ihr Computer möglicherweise standardmäßig kein Broadcasting zu, es sei denn, dieser Prozeß geht vom Root-Benutzer aus.

Nur für Windows

Welche WinSock-DLLs können mit LabVIEW verwendet werden?

Diese Frage betrifft nur Windows 3.*x*, da Windows 95 und Windows NT diese Datei bereits enthalten.

All WinSock-Treiber, die mit dem Standard 1.1 übereinstimmen, dürften ohne Probleme mit LabVIEW arbeiten.

Empfohlen

National Instruments empfiehlt, die WinSock-DLL-Datei zu verwenden, die von Microsoft für Windows für Workgroups bereitgestellt wird. Vor der Freigabe dieser WinSock-DLL-Datei hat National Instruments eine Reihe

von WinSock-DLLs getestet. Aufgrund dieser Tests wurden damals die folgenden DLL-Dateien empfohlen:

- TCPOpen Version 1.2.2 der Lanera Corporation.
- Trumpet Version 1.0.
- Super-TCP Version 3.0 R1 der Frontier Technologies Corporation.
- NEWT/Chameleon Version 3.11 von NetManage, Inc.

Nicht empfohlen

Bei den beschränkten Tests von National Instruments führten die folgenden Produkte beim Versuch, die TCP/IP-Kommunikation aufzunehmen, zu verschiedenen Problemen und Abstürzen. Zum gegenwärtigen Zeitpunkt können wir diese Produkte weder empfehlen noch Unterstützung für Kunden anbieten, die versuchen, die TCP/IP-Kommunikation mit diesen WinSock-DLLs aufzubauen.

- Distinct TCP/IP Version 3.1 der Distinct Corporation.
- PCTCP Version 2.*x* von FTP Software, Inc.

Wie wird ein Excel-Makro mit Hilfe von DDE aufgerufen?

Verwenden Sie das VI `DDE Execute.vi`. Dieses VI erteilt dem DDE-Server die Anweisung, eine Befehlszeichenfolge auszuführen, in der Sie die Aktion, die Excel durchführen soll, und den Namen des Makros angeben. Achten Sie darauf, daß Sie den Befehl mit den korrekten Klammern umgeben. Weitere Informationen finden Sie im *Excel Benutzerhandbuch*. Die folgende Tabelle enthält einige gebräuchliche Beispiele:

Befehlszeichenfolge	Aktion
[RUN("MACRO1")]	Führt MACRO1 aus
[RUN("MACRO1!R1C1")]	Führt MACRO1 ab Reihe 1, Spalte 1 aus
[OPEN("C:\EXCEL\SURVEY.XLS")]	Öffnet `SURVEY.XLS`

Warum arbeitet DDE Poke nicht mit Microsoft Access?

Microsoft Access kann keine Daten direkt von DDE-Clients empfangen. Wenn Sie Daten in eine Access-Datenbank aufnehmen möchten, müssen Sie in der entsprechenden Datenbank ein Makro für das Importieren der Daten aus einer Datei erstellen. In einfachen Fällen genügt es, wenn diese Makros lediglich zwei Aktionen umfassen. Geben Sie zuerst den Befehl

"SetWarnings" in das Makro ein, um Access-Dialogfelder zu unterdrücken. Mit den Befehlen "TransferSpreadsheet" oder "TransferText" können Sie die Daten dann abrufen. Wenn Sie dieses Makro definiert haben, können Sie das Makro aufrufen, indem Sie einen Ausführbefehl, der den Makronamen als Dateneingabe enthält, an die Datenbank senden. Das Beispiel-VI `Sending Data to Access.vi`, das sich in der Bibliothek `examples\network\access.llb` befindet, demonstriert, wie Sie dabei vorgehen müssen.

Welche Befehle müssen verwendet werden, um mit einer anderen als einer LabVIEW-Anwendung mit Hilfe von DDE zu kommunizieren?

Jede Anwendung, mit der Sie eine Schnittstelle herstellen, verwendet eigene DDE-Befehle. Sehen Sie in der Dokumentation für die Anwendung nach, welche Befehle zur Verfügung stehen.

Wie wird LabVIEW zur gemeinsamen Nutzung auf einem Datei-Server installiert?

Wenn Sie für jeden Client eine eigene Lizenz besitzen, können Sie folgendermaßen verfahren:

- Installieren Sie das Full Development System von LabVIEW auf dem Server. (Sofern sich keine NI-Hardware auf dem Server befindet, ist es nicht notwendig, NI-DAQ oder `GPIB.DLL` zu installieren.)
- Jeder lokale Computer sollte eine eigene `labview.ini`-Datei für LabVIEW-Voreinstellungen verwenden. Wenn auf einem lokalen Computer nicht bereits eine `labview.ini`-Datei vorhanden ist, können Sie dieses (leere) Textdokument mit einem Texteditor wie z.B. Microsoft Notepad erstellen. Die erste Zeile in `labview.ini` muß den Eintrag [`labview`] enthalten. Damit eine lokale Einstellung für `labview.ini` möglich ist, benötigt LabVIEW ein Befehlszeilenargument, das den Pfad zu den Voreinstellungen angibt. Wenn sich z.B. die Datei `labview.exe` auf dem Laufwerk `W:\LABVIEW` und die Datei `labview.ini` auf `C:\LVWORK` (d.h. auf der Festplatte des lokalen Computers) befindet, müssen Sie die Befehlszeilenoption für das LabVIEW-Symbol im Programm-Manager folgendermaßen abändern:
  `W:\LABVIEW\LABVIEW.EXE -pref C:\LVWORK\LABVIEW.INI`

☞ **Hinweis** `pref` ***muß in Kleinbuchstaben eingegeben werden. Darüber hinaus muß jeder lokale Computer ein eigenes temporäres LabVIEW-Verzeichnis aufweisen, das Sie in LabVIEW durch Wahl von*** **Bearbeiten»Voreinstellungen...** ***festlegen können.***

- Auf dem Server-Computer ist keine `GPIB.DLL`-Datei notwendig, es sei denn, Sie verwenden eine GPIB-Karte auf diesem Computer. Sie benötigen die Datei `gpibdrv`, die sich im LabVIEW-Verzeichnis befindet. Sie müssen den Treiber für diese GPIB-Karte auf jedem Computer installieren, der mit einer GPIB-Karte ausgestattet ist. Verwenden Sie entweder die Treiber, die mit der Karte geliefert wurden, oder passen Sie die Installation von LabVIEW so an, daß nur der gewünschte GPIB-Treiber auf dem lokalen Computer installiert wird.
- Für NI-DAQ müssen Sie dasselbe Verfahren anwenden wie für die `GPIB.DLL`-Datei.

Warum blockiert der Synch DDE Client / Server in Windows NT nach vielen Übertragungen?

In LabVIEW für Windows NT gibt es einige Probleme mit DDE, die bewirken, daß VIs im Verlauf von DDE Poke- und DDE Anforderungs-Operationen blockieren. Diese Einschränkung bezieht sich nur auf Windows NT.

Nur für Macintosh

Was ist eine Ziel-ID?

Ziel-IDs werden in den VIs für Apple Events und für PPC auf dem Macintosh verwendet. Eine Ziel-ID stellt einen Verweis auf die Anwendung dar, die Sie zu starten, auszuführen oder abzubrechen versuchen. Mit einem der folgenden VIs können Sie die Ziel-ID für eine Anwendung ermitteln:

- Das VI “Get Target ID” übernimmt den Namen und Speicherort der Anwendung als Eingabe, durchsucht das Netzwerk nach der Anwendung und gibt die Ziel-ID aus.
- Durch den PPC Browser wird ein Dialogfeld eingeblendet, über das Sie eine Anwendung über das Netzwerk oder auf Ihrem eigenen Computer auswählen können.

Sie können die Ziel-ID, die Sie ermitteln, als Eingabe für alle nachfolgenden Apple Event-Funktionen verwenden, die zum Öffnen, Schließen und Ausführen der Anwendung sowie zum Drucken aus der Anwendung dienen.

Warum wird eine Anwendung nicht im Dialogfeld angezeigt, das durch den PPC Browser generiert wird?

Wenn die Anwendung, zu der Sie eine Verbindung herstellen möchten, nicht mit Apple Events verwendet werden kann, wird sie auch nicht im Dialogfeld angezeigt, das vom `PPC Browser` erstellt wird. Falls Sie sicher sind, daß die gewünschte Anwendung Apple Events unterstützt, sollten Sie das **Apple**-Menü auf dem Computer öffnen, auf der sich die Anwendung befindet. Wählen Sie **Kontrollfelder»File Sharing (MacOS 8)** oder **Kontrollfelder»Gemeinschaftsfunktionen (System 7 und früher)**, und vergewissern Sie sich, daß **Programmverbindung** eingeschaltet ist.

Wie kann der Finder mit Hilfe von Apple Events geschlossen werden?

Schließen Sie den Finder sowie alle anderen Anwendungen mit dem VI `AESend Quit Application`.

GPIB

Alle Plattformen

Bei Verwendung eines LabVIEW-Instrumententreibers entstehen Probleme bei der Kommunikation mit einem Instrument.

Stellen Sie sicher, daß die GPIB-Schnittstelle ordnungsgemäß arbeitet. Verwenden Sie das Beispiel-VI `LabVIEW<->GPIB.vi`, das sich in `examples\instr\smplgpib.llb` befindet. Versuchen Sie, einen einfachen Befehl an das Instrument zu senden. Wenn es sich dabei z.B. um ein 488.2-Instrument handelt, erhält es durch den String `*IDN?` die Aufforderung, seinen Identifikations-String (ungefähr 100 Zeichen) zu senden.

Sobald die Kommunikation hergestellt ist, sollten keine weiteren Probleme mit dem Instrumententreiber auftreten.

Durch GPIB Lesen/Schreiben werden in LabVIEW Zeitbegrenzungsfehler verursacht.

Versuchen Sie, ein einfaches Programm auszuführen, um die Kommunikation zwischen LabVIEW und GPIB herzustellen. Verwenden Sie das Beispiel-VI `LabVIEW<->GPIB.vi`, das sich in `examples\instr\smplgpib.llb` befindet. Versuchen Sie, einen einfachen Befehl an das Instrument zu senden. Wenn es sich dabei z.B. um ein 488.2-Instrument handelt, erhält das Instrument durch den String `*IDN?`

die Aufforderung, seinen Identifikations-String (ungefähr 100 Zeichen) zu senden.

Wenn Sie mit GPIB trotzdem noch Fehler erhalten, liegt unter Umständen ein Konfigurationsproblem vor. Öffnen Sie das Konfigurationsprogramm für GPIB (Windows: `wibconf`; Mac: `NI-488 Config`; Sun: `ibconf`; HP-UX: `ibconf`). Vergewissern Sie sich, daß die Einstellungen mit Ihren Hardware-Einstellungen übereinstimmen. Beenden Sie das Konfigurationsprogramm, und führen Sie das `ibic`-Utility-Programm (Interface Bus Interactive Control) für Ihre Windows-Plattform aus: `wibic` (Mac: `ibic` – wird mit der GPIB-Karte geliefert; Sun: `ibic`; HP-UX: `ibic`).

Versuchen Sie, die folgende Sequenz zu verwenden:

`: ibfind gpib0`	Sucht GPIB-Schnittstelle.
`id = 32000` `gpib0: ibsic`	Löscht Daten im GPIB-Bus (Send Interface Clear).
`[0130] [cmpl cic atn]`	Operation erfolgreich durchgeführt.
`gpib0: ibfind dev1`	Sucht Gerät 1. Verwenden Sie die entsprechende Instrumentenadresse.
`id = 32xxx` `dev1: ibwrt "*IDN?"`	Schreibt den String an das Instrument. 488.2-Instrumente erkennen diesen Befehl und geben ihren Identifikations-String aus.
`count = 5` `dev1: ibrd 100`	Es wurden fünf Bytes gesendet. Bis zu 100 Bytes können aus dem Instrument gelesen werden.
`Fluke xxx Multimeter…`	Das Instrument gibt den Identifikations-String aus.

Falls Konfigurationsfehler vorliegen, wird für einen dieser Schritte eine Fehlermeldung ausgegeben. Das NI 488.2-Softwarehandbuch, das Sie zusammen mit der GPIB-Karte erhalten haben, enthält detaillierte Beschreibungen zu den Fehlermeldungen.

Warum ist die Kommunikation mit einem GPIB-Instrument möglich, wenn ein LabVIEW-VI im Highlight-Modus ausgeführt wird, jedoch nicht, wenn das VI mit voller Geschwindigkeit arbeitet?

Es scheint sich dabei um ein Timing-Problem zu handeln. VIs arbeiten wesentlich langsamer als normal, wenn die Highlight-Option eingeschaltet ist. Eventuell benötigt das Instrument mehr Zeit für die Vorbereitung der zu sendenden Daten. Ergänzen Sie eine Verzögerungsfunktion, oder verwenden Sie Service-Anforderungen, um dem Instrument vor dem `GPIB Read.vi` ausreichend Zeit zu lassen, die Daten, die an den Computer zurückgesendet werden müssen, zu erzeugen.

Warum ist das Schreiben von Daten an das GPIB-Instrument problemlos möglich, während beim Lesen Probleme auftreten?

Während des Ausführens von `GPIB Write.vi` befindet sich der Computer im Talk-Modus und das Instrument im Listen-Modus. Wenn `GPIB Read.vi` ausgeführt wird, schaltet das Instrument normalerweise in den Talk-Modus und der Computer in den Listen-Modus um. Das Instrument wird durch ein Abschlußsignal, das als ein Zeichen (End Of String/Ende des Strings) oder als GPIB-Buszeile (End Or Identify/Ende oder Identifikation) gesendet wird, zum Umschalten in den anderen Modus aufgefordert. Wenn sich also für das `GPIB Read.vi` eine Zeitbegrenzung ergibt oder wenn es einen EABO-Fehler (Operation abgebrochen) ausgibt, bedeutet dies, daß das Instrument kein korrektes Abschlußsignal erhält. Schlagen Sie im Handbuch für das jeweilige Instrument nach, um den zutreffenden Abschlußmodus zu ermitteln. Als allgemeine Regel gilt, daß alle IEEE 488.2-Geräte `<CR><LF>` und die Bestätigung der EOI-Zeile (End Or Identify) im GPIB-Bus als Abschlußsignal erkennen.

Verwenden Sie das Konfigurationsprogramm für Ihre Plattform (Windows: `wibconf`; Mac: `NI-488 Config`; Sun: `ibconf`; HP-UX: `ibconf`), um das Abschlußzeichen zu ändern.

Ein VI, das mit einem GPIB-Instrument kommuniziert, arbeitet problemlos auf einer Plattform, aber nicht auf einer anderen.

Vergewissern Sie sich, daß das Instrument ordnungsgemäß in wibconf, NI-488 Config oder ibconf konfiguriert ist. Manche älteren 488.1-Instrumente schalten nicht automatisch in den Remote-Status um. Wechseln Sie zum GPIB-Konfigurationsprogramm für Ihre Plattform, und aktivieren Sie das Feld `Assert REN when SC`. Auf diese Weise stellen Sie sicher, daß das Instrument in den Remote-Status (im Gegensatz zum lokalen Status) umschaltet, wenn es angesprochen wird.

Nur für Windows

Die Kommunikation mit dem Instrument ist zwar in einem QuickBasic-Programm möglich, aber nicht in LabVIEW.

Die GPIB-Karte verfügt über separate Handler für DOS und Windows. Quick Basic greift auf den DOS-Handler zu, während LabVIEW den Windows-Handler benötigt. Stellen Sie sicher, daß die Karte und das Instrument korrekt über `wibconf.exe` konfiguriert sind.

Serielle I/O-Vorgänge

Alle Plattformen

Warum reagiert das Instrument nicht auf Befehle, die mit dem seriellen Port Write VI gesendet werden?

Viele Instrumente erwarten einen Wagenrücklauf- oder Zeilenvorschubbefehl als Abschluß eines Befehls-Strings. Das VI `Serial Port Write.vi` in LabVIEW sendet nur die Zeichen, die als String-Eingabe vorhanden sind; es wird kein Abschlußzeichen an den String angehängt. Viele Terminalemulationspakete (z.B. Windows Terminal) hängen automatisch am Ende aller Übertragungen einen Wagenrücklaufbefehl an. Sofern Ihr Instrument ein Abschlußzeichen benötigt, müssen Sie jedoch bei LabVIEW das korrekte Abschlußzeichen als Teil der String-Eingabe für das `Serial Port Write.vi` eingeben.

Manche Instrumente benötigen einen Wagenrücklaufbefehl (`\r`), andere Instrumente dagegen einen Zeilenvorschubbefehl (`\n`). Wenn Sie auf die Rücklauftaste auf der Tastatur (auf einer PC-Tastatur ist das die Enter- oder Eingabetaste auf der alphanumerischen Tastatur) drücken, fügt LabVIEW `\n` ein. Wenn Sie einen Wagenrücklaufbefehl eingeben möchten, können Sie entweder die Funktion “Strings verknüpfen” verwenden und eine Wagenrücklaufkonstante an den String anfügen, oder Sie können manuell `\r` nach Auswahl von '`\`' **Code-Anzeige** im Popup-Menü für den String eingeben.

Vergewissern Sie sich, daß das Kabel korrekt arbeitet. Viele der Probleme, die von unserer technischen Support-Abteilung bearbeitet werden, sind auf schlechte Kabelverbindungen zurückzuführen. Bei der Kommunikation zwischen Computern über den seriellen I/O sollten Sie ein Null-Modem verwenden, um die Empfangs- und Übertragungssignale umzukehren.

Im Beispiel `LabVIEW<->Serial.vi` können Sie sehen, wie Sie die Kommunikation mit Ihrem Instrument herstellen können. Dieses Beispiel befindet sich in `examples\instr\smplserl.llb`. Das VI demonstriert außerdem die Verwendung des VIs `Bytes at Serial Port.vi` vor dem Lesen der Daten des seriellen Ports.

Wie sollte der serielle Port geschlossen werden, damit andere Anwendungen ihn anschließend verwenden können?

Es ist unter Umständen sinnvoll, den Port zu schließen, wenn Sie ihn nicht mehr benötigen. Es ist z.B. möglich, daß ein VI in Windows mit dem VI `Serial Port Write.vi` Informationen an den mit einem Drucker verbundenen Port lpt1 schreibt. Auch nach Abschluß dieser Operation hat LabVIEW immer noch die Kontrolle über den seriellen Port. Andere Anwendungen können diesen Port erst dann verwenden, wenn LabVIEW die Kontrolle abgegeben hat.

LabVIEW verfügt auf allen Plattformen über ein `Close Serial Driver.vi`. Durch dieses VI wird LabVIEW angewiesen, die Kontrolle über den angegebenen Port abzugeben. Das Close Serial Driver.vi befindet sich nicht in der Palette **Seriell**, sondern in `vi.lib\Instr\_sersup.llb`. Sie können auf das VI über die Befehle **Funktionen**»**VI...** oder **Datei**»**Öffnen...** zugreifen.

Wie werden die Daten im seriellen Portpuffer gelöscht?

Lesen Sie die verbleibenden Daten im Puffer, und ignorieren Sie sie.

Wie kann der Computer mit zusätzlichen seriellen Ports ausgestattet werden?

Mit seriellen AT-Bus-Schnittstellenkarten von National Instruments können Sie Ihrem IBM-kompatiblen Computer zusätzliche serielle Ports hinzufügen. Wenn Sie auf einer anderen Plattform arbeiten, können Sie die Karten von Drittherstellern verwenden, um Ihren Computer mit weiteren seriellen Ports auszustatten. Für manche Drittprodukte ist eine spezielle Sprachschnittstelle notwendig, die mit Standard-API für serielle Ports auf der Plattform nicht konform ist. In diesem Fall müssen Sie Ihre eigene Schnittstelle in den Treiber schreiben (vermutlich als CIN oder DLL). Die folgenden Abschnitte beschreiben, wie Sie die VIs für serielle Ports, die in

LabVIEW enthalten sind, verwenden können, um auf verschiedenen Plattformen Karten anzusprechen, die die Standardschnittstelle für serielle Ports verwenden.

(Windows 3.*x*) LabVIEW für Windows verwendet die Standardschnittstelle von Microsoft Windows für den seriellen Port. Eventuelle Einschränkungen in der Verwendbarkeit von seriellen Ports in LabVIEW sind daher auf Windows zurückzuführen. Da Windows z.B. nur in der Lage ist, die Ports COM1 bis COM9 anzusprechen, kann auch LabVIEW nur diese Ports ansprechen. Darüber hinaus erlaubt Windows den gleichzeitigen Zugriff nur auf höchstens acht serielle Ports. Aus diesem Grund kann auch LabVIEW nur höchstens acht serielle Ports gleichzeitig steuern.

National Instruments bietet mittlerweile serielle Plug & Play AT-Karten für eine Vielzahl von Ansätzen in der seriellen Kommunikation an. Die asynchronen seriellen Schnittstellenkarten AT-485 und AT-232 stehen in 2- oder 4-Port-Konfigurationen zur Verfügung. Aufgrund der vollen Plug & Play-Kompatibilität können Sie die Vorteile einer umschaltfreien Konfiguration nutzen, wodurch die Installation und die Wartung erleichtert wird. Die Karten AT-485 und AT-232 schließen die folgenden Softwarekomponenten für den Einsatz unter Windows ein: Gerätetreiber, Hardwarediagnosetest und Konfigurationsprogramm. Diese Karten wurden in Verbindung mit LabVIEW für Windows getestet. Wenden Sie sich an National Instruments, falls Sie weitere Informationen benötigen:

Hersteller:	National Instruments
Produktname:	AT-485 und AT-232
Telefon:	512 794 0100 (USA)
Fax:	512 794 8411 (USA)
Faxabruf:	800 329 7177 (USA), Bestellnummer 1430

Eine Karte eines Drittherstellers, für die ein alternativer Ansatz zur Umgehung der Windows-Beschränkungen notwendig ist, arbeitet mit großer Wahrscheinlichkeit nicht einwandfrei oder möglicherweise überhaupt nicht mit LabVIEW. Es ist möglich, ein CIN oder DLL zu schreiben, das auf solche Karten zugreift; es wird jedoch davon abgeraten. Im allgemeinen sollten alle Karten, die die Standardschnittstelle von Windows verwenden, mit LabVIEW vollkommen kompatibel sein.

(Windows 95 und **Windows NT)** Anders als Windows 3.*x* sind Windows 95 und Windows NT nicht auf acht serielle Ports beschränkt. Unter Windows 95 und Windows NT kann LabVIEW bis zu 256 serielle Ports ansprechen. Der Standardparameter für die Portnummer ist 0 für COM1, 1 für COM2, 2 für COM3 und so weiter.

Die Datei `labview.ini` enthält die LabVIEW-Konfigurationsoptionen. Sie können die Geräte bestimmen, die von den VIs für den seriellen Port verwendet werden, indem Sie für die Konfigurationsoption `labview.serialDevices` die Liste der zu verwendenden Geräte festlegen. Geben Sie folgenden Text ein, um die Geräte ähnlich wie in Windows 3.*x* einzurichten:

```
serialDevices="COM1;COM2;COM3;COM4;COM5;COM6;COM7;COM8;
COM9;\\.\COM10;\\.\COM11;\\.\COM12;\\.\COM13;\\.\COM14;
\\.\COM15;\\.\COM16;LPT1;LPT2"
```

Der obige Text sollte in der Konfigurationsdatei als ununterbrochene Zeile erscheinen.

(Macintosh) LabVIEW verwendet reguläre System-INITs zur Kommunikation mit den seriellen Ports. LabVIEW benutzt standardmäßig die Treiber `.aIn` und `.aOut`, d.h. den Modemport, für Port 0 und die Treiber `.bIn` und `.bOut`, d.h. den Druckerport, für Port 1. Wenn Sie auf zusätzliche Ports zugreifen möchten, müssen Sie zusätzliche Karten mit den dazugehörigen INITs installieren.

Nach der Installation der Karte und der INIT-Datei(en) können Sie LabVIEW mitteilen, wie die zusätzlichen Ports verwendet werden sollen. Die in LabVIEW 5.0 verwendete Methode unterscheidet sich geringfügig von der Methode, die in früheren Versionen verwendet wurde.

Modifizieren Sie das globale VI `serpOpen.vi` (in `vi.lib\Instr\_sersup.llb`), so daß die zusätzlichen Ports berücksichtigt werden. Alle VIs für serielle Ports rufen das VI `Open Serial Driver.vi` auf, das die entsprechenden Eingabe- und/oder Ausgabetreibernamen im `serpOpen.vi` liest. Das Frontpanel von `serpOpen.vi` enthält jeweils ein String-Array für Eingabe- und Ausgabetreibernamen.

Wenn das VI `Open Serial Driver.vi` aufgerufen wird, verwendet es die eingegebene Portnummer als Index für die String-Arrays. Damit LabVIEW also zusätzliche serielle Ports erkennen kann, müssen Sie zusätzliche String-Elemente zu den Eingabe- und Ausgabetreibernamen hinzufügen und dann die Option **Aktuelle Werte als Standard** wählen. Speichern Sie die Änderungen im VI `serpOpen.vi`.

Jeder serielle Port auf einer Einsteckkarte hat jeweils einen Namen für Eingaben und für Ausgaben. Die genauen Namen und die Anleitungen für die Installation der Treiber erhalten Sie als Teil der Dokumentation für die Karte. Wenden Sie sich an den Hersteller der Karte, falls die Anleitungen fehlen oder unverständlich sind.

Die folgenden Karten sind für den Einsatz mit LabVIEW für Macintosh geeignet:

Hersteller:	Creative Solutions, Inc.
Produktname:	Hurdler HQS (4 Ports) oder HDS (2 Ports)
Telefon:	301 984 0262 (USA)
Fax:	301 770 1675 (USA)

Hersteller:	Greensprings
Produktname:	RM 1280 (4 Ports)
Telefon:	415 327 1200 (USA)
Fax:	415 327 3808 (USA)

(UNIX) Auf einer Sun SPARCstation unter Solaris 1 und auf Concurrent PowerMAX wird für die VIs für serielle Ports als **Portnummer**-Parameter 0 für `/dev/ttya` verwendet, 1 für `/dev/ttyb` und so weiter. Unter Solaris 2 bezieht sich Port 0 auf `/dev/cua/a`, 1 auf `/dev/cua/b` und so weiter. Unter HP-UX bezieht sich 0 auf `/dev/tty00`, 1 auf `/dev/tty01` und so weiter.

Auf Concurrent PowerMAX bezieht sich Port 0 auf `/dev/console`, Port 1 auf `/dev/tty1`, Port 2 auf `/dev/tty2` und so weiter.

Da andere Hersteller eventuell beliebige Gerätenamen für ihre Karten für serielle Ports verwenden, hat LabVIEW eine unkomplizierte Schnittstelle entwickelt, über die das Numerieren der Ports auf einfache Weise erfolgt. In LabVIEW für Sun, HP-UX und Concurrent PowerMAX existiert eine Konfigurationsoption, mit der LabVIEW mitgeteilt wird, wie die seriellen Ports adressiert werden sollen. LabVIEW unterstützt alle Karten, die UNIX-Standardgeräte verwenden. Manche Hersteller empfehlen für ihre

Karten die Verwendung von `cua-` anstelle von `tty`-Geräteknoten. LabVIEW kann beide Knotentypen ansprechen.

Die Datei `.labviewrc` enthält die LabVIEW-Konfigurationsoptionen. Sie können die Geräte bestimmen, die von den VIs für den seriellen Port verwendet werden, indem Sie für die Konfigurationsoption `labview.serialDevices` die Liste der Geräte festlegen, die Sie zu verwenden beabsichtigen.

Die Standardeinstellung lautet beispielsweise:

```
labview.serialDevices:/dev/ttya:/dev/ttyb:/dev/ttyc:...
:/dev/ttyz.
```

☞ **Hinweis** ***Es wird dabei vorausgesetzt, daß alle Installationen von Karten eines Drittherstellers eine Methode zum Erstellen einer*** `standard/dev`***-Datei (Knoten) einschließen und daß dem Benutzer der Name dieser Datei bekannt ist.***

Die folgenden Karten sollten für den Einsatz mit LabVIEW für Sun geeignet sein:

Hersteller:	Sun
Produktname:	SBus Serial Parallel Controller (8 seriell, 1 parallel)
Hersteller:	Artecon, Inc.
Produktname:	SUNX-SB-300P ArtePort SBus Card mit 3 ser. Ports/ 1 par. Port SUNX-SB-400P ArtePort SBus Card mit 4 ser. Ports/ 1 par. Port SUNX-SB-1600 ArtePort SBus Card mit 16 seriellen Ports

Wenden Sie sich bei Fragen zu einem dieser Produkte unter der Rufnummer (800) 873-7869 (nur in Nordamerika) an SunExpress.

Wie können die seriellen DTR- und RTS-Leitungen kontrolliert werden?

Das VI `Serial Port Init.vi` kann dazu verwendet werden, den seriellen Port für das Hardware-Handshaking zu konfigurieren. Bei manchen Anwendungen ist jedoch ein manuelles Umschalten der DTR- und RTS-Leitungen notwendig. Da die Schnittstelle für die seriellen Ports von der Plattform abhängt, erfolgt die Steuerung der Leitungen für jede Plattform auf unterschiedliche Weise.

(Windows) Die LabVIEW-Version für Windows enthält ein VI, das zur Steuerung der seriellen DTR- und RTS-Leitungen eingesetzt werden kann. Sie finden dieses VI unter der Bezeichnung `serial line ctrl.vi`, in `vi.lib\Instr\_sersup.llb`. Das VI schaltet zwischen den beiden Leitungen entsprechend der Funktionseingabe hin und her. Die gültigen Eingabecodes lauten:

0 noop
1 clear DTR
2 set DTR
3 clear RTS
4 set RTS
5 set DTR protocol
6 clr DTR protocol
7 noop2

(Macintosh) Auf dem Macintosh können Sie den seriellen Port mit der Funktion “Gerätesteuerung/Status” steuern. Das Handbuch *Inside Macintosh* (siehe Band II, Seiten 245–259 und Band IV, Seiten 225–228) enthält genaue Hinweise, welche Codes an den seriellen Port gesendet werden können. Die folgende Tabelle enthält eine Zusammenfassung der Codes und ihrer Funktionen:

Code	Param	Wirkung
13	baudRate	Stellt Baudrate ein (tatsächliche Rate, als ganzzahliger Wert).
14	serShk	Stellt die Handshake-Parameter ein.
16	byte	Stellt verschiedene Steueroptionen ein.
17		Bestätigt DTR.
18		Negiert DTR.
19	char	Ersetzt Paritätsfehler.

Code	Param	Wirkung
20	2 chars	Ersetzt Paritätsfehler durch alternative Zeichen.
21		Stellt XOff uneingeschränkt für Ausgabeflußsteuerung ein.
22		Stellt XOff uneingeschränkt für Ausgabeflußsteuerung frei.
23		Sendet XOn für Eingabeflußsteuerung, falls XOff zuletzt gesendet wurde.
24		Sendet XOn uneingeschränkt für Eingabeflußsteuerung.
25		Sendet XOff für Eingabeflußsteuerung, falls XOn zuletzt gesendet wurde.
26		Sendet XOff uneingeschränkt für Eingabeflußsteuerung.
27		Stellt SCC-Kanal zurück.

(Sun) LabVIEW für Sun enthält keine spezielle Unterstützung für das Umschalten zwischen den Hardware-Handshaking-Leitungen der seriellen Ports. Zur manuellen Steuerung dieser Leitungen müssen Sie einen CIN erstellen. Detaillierte Hinweise, wie ein CIN geschrieben wird, finden Sie im Abschnitt *Steps for Creating a CIN* in Kapitel 1, *CIN Overview,* im *LabVIEW Code Interface Reference Manual*. Dieses Handbuch ist nur im PDF-Format (Portable Document Format) verfügbar.

Warum ist es nicht möglich, einen seriellen Puffer zuzuweisen, der größer als 32 Kbyte ist?

Sie können im `Serial Port Init.vi` keine Puffer verwenden, die größer als 32 K sind, da die Größe des seriellen Portpuffers sowohl in Windows als auch auf dem Macintosh auf 32 K beschränkt ist. Wenn Sie trotzdem einen größeren Puffer zuweisen, beschneidet LabVIEW die Größe des Puffers auf 32 K. Auf Sun-Systemen existiert dieses Problem nicht.

Nur für Windows

Wie erfolgt der Zugriff auf den parallelen Port?

In LabVIEW für Windows 3.*x* entspricht Port 10 LPT1, Port 11 entspricht LPT2 und so weiter. Unter Windows 95/NT können Sie einen Port über

“Serielle Geräte” als LPT1 festlegen. Wenn Sie Daten an einen Drucker senden möchten, der an einen parallelen Port angeschlossen ist, sollten Sie das VI `Serial Port Write.vi` verwenden.

Welche Bedeutung haben die Fehlernummern, die von den VIs für serielle Ports ausgegeben werden?

Die VIs für serielle Ports in LabVIEW für Windows geben die Fehler aus, die von der GetCommError-Funktion in Windows gemeldet werden. Die Fehlerwerte, die von den VIs für serielle Ports gemeldet werden, setzen sich aus einer ‘Oder’-Verknüpfung von 0x4000 (16384) mit den Fehlerwerten in der folgenden Tabelle zusammen. Beachten Sie, daß der ausgegebene Fehler den Status des seriellen Ports widerspiegelt; es ist möglich, daß der Fehler als Resultat einer Funktion erzeugt wird, die zuvor über den seriellen Port ausgeführt wurde. Die ausgegebenen Werte können eine Kombination der folgenden Fehler darstellen:

Hex. Wert	Fehlername	Beschreibung
0x0001	CE_RXOVER	Die Empfangsschlange ist übergelaufen. In der Eingangsschlange war entweder kein Platz mehr, oder nach Eingang des EOF-Zeichens wurde noch ein weiteres Zeichen empfangen.
0x0002	CE_OVERRUN	Ein von der Hardware ausgegebenes Zeichen wurde nicht vor Eintreffen des nächsten Zeichens gelesen. Das Zeichen ist verlorengegangen.
0x0004	CE_RXPARITY	Die Hardware hat einen Paritätsfehler festgestellt.
0x0008	CE_FRAME	Die Hardware hat einen Rahmenfehler festgestellt.
0x0010	CE_BREAK	Die Hardware hat einen Break-Zustand festgestellt.

Hex. Wert	Fehlername	Beschreibung
0x0020	CE_CTSTO	CTS (Clear-To-Send)-Zeitbegrenzung. Während der Übertragung eines Zeichens war CTS für die durch fCtsHold in COMSTAT festgelegte Dauer niedrig.
0x0040	CE_DSRTO	DSR (Data-Set-Ready)-Zeitbegrenzung. Während der Übertragung eines Zeichens war CTS für die durch fDsrHold in COMSTAT festgelegte Dauer niedrig.
0x0080	CE_RLSDTO	RLSD (Receive-Line-Signal-Detect)-Zeitbegrenzung. Während der Übertragung eines Zeichens war RLSD für die durch fRlsdHold in COMSTAT festgelegte Dauer niedrig.
0x0100	CE_TXFULL	Die Übertragungsschlange war voll, als eine Funktion versuchte, ein Zeichen einzureihen.
0x0200	CE_PTO	Während eines Versuchs, mit einem parallelen Gerät zu kommunizieren, ist eine Zeitbegrenzung eingetreten.
0x0400	CE_IOE	Während eines Versuchs, mit einem parallelen Gerät zu kommunizieren, ist ein I/O-Fehler eingetreten.
0x0800	CE_DNS	Es wurde kein paralleles Gerät ausgewählt.

Hex. Wert	Fehlername	Beschreibung
0x1000	CE_OOP	Ein paralleles Gerät hat angezeigt, daß kein Papier mehr vorhanden ist.
0x8000	CE_MODE	Der angeforderte Modus wird nicht unterstützt, oder der Parameter `idComDev` ist ungültig. Sofern eingestellt, ist CE_MODE der einzige gültige Fehler.

Mit Hilfe dieser Tabelle können Sie die ausgegebene Fehlernummer in ihre einzelnen Fehlerkomponenten aufteilen. Wenn das VI `Serial Port Write.vi` z.B. den Fehler 16,408 ausgibt, werden dadurch die Fehler `CE_BREAK und CE_FRAME` angezeigt
(16,408 = 16,384 + 16 + 8 = 0x4000 + 0x0010 + 0x0008).

Nur für Sun

Beim Durchführen eines seriellen I/O wird der Fehler –37 ausgegeben.

Der Fehler –37 bedeutet, daß LabVIEW das entsprechende serielle Gerät nicht finden kann. Es wird dadurch angezeigt, daß a) die Dateien `/dev/tty?` nicht auf Ihrem Computer vorhanden sind oder b) LabVIEW die Datei `serpdrv` nicht finden kann.

LabVIEW adressiert `/dev/ttya` standardmäßig als Port 0, `/dev/ttyb` als Port 1 und so weiter. Diese Geräte müssen vorhanden sein, und der Benutzer muß Lese- und Schreibzugriff auf diese Geräte haben. Sie können die Geräte, auf die LabVIEW zugreift, mit den VIs für serielle Ports ändern, indem Sie die Konfigurationsoption “serialDevices” in Ihrer `.Xdefaults`-Datei ergänzen. Die *LabVIEW Versionshinweise* enthalten Hinweise zur Verwendung dieser Option.

Die Datei `serpdrv` wird mit LabVIEW ausgeliefert und dient als Schnittstelle zwischen LabVIEW und den seriellen Ports von Sun-Computern. Diese Datei muß sich an dem Speicherplatz befinden, der durch die libdir-Konfigurationsoption gekennzeichnet wird—standardmäßig handelt es sich dabei um das LabVIEW-Verzeichnis. Dies bedeutet, daß `serpdrv` im selben Verzeichnis gespeichert sein muß wie `gpibdrv` und `vi.lib`.

Serielle I/O-Vorgänge blockieren auf einem Solaris 1.*x*-Computer.

LabVIEW für Sun verwendet asynchrone I/O-Aufrufe bei der Durchführung von Operationen über den seriellen Port. Im Kernel `Generic_Small` wurde der asynchrone I/O-Vorgang deaktiviert. Um von LabVIEW für Sun auf die seriellen Ports zugreifen zu können, muß der Benutzer entweder den Standard-Generic-Kernel (nicht `Generic_Small`) verwenden oder den Kernel `Generic_Small` überarbeiten und SPARC neu starten.

C

Kundenbetreuung

Um Ihnen im Falle technischer Probleme die Zusammenarbeit mit uns zu erleichtern, sind in diesem Anhang Formulare enthalten, die Ihnen dabei helfen, die technischen Informationen zusammenzustellen, die wir zur Lösung Ihrer technischen Probleme benötigen. Außerdem ist ein Formular enthalten, auf dem Sie Kommentare zur Produktdokumentation abgeben können. Wenn Sie sich mit uns in Verbindung setzen, benötigen wir die Informationen auf dem Formular für Technische Unterstützung sowie auf dem Konfigurationsformular, soweit dies in Ihrem Handbuch enthalten ist, mit dessen Hilfe wir Ihre Fragen zur Systemkonfiguration so schnell wie möglich beantworten.

National Instruments bietet technische Unterstützung auf elektronischem Wege und per Fax und Telefon und läßt Ihnen somit die von Ihnen benötigten Informationen schnellstmöglich zukommen. Zu unserem elektronischen Service gehört ein Bulletin-Board, eine FTP-Site, ein Faxabrufsystem sowie E-Mail-Unterstützung. Versuchen Sie bei Hardware- oder Softwareproblemen erst unser elektronisches Unterstützungssystem. Sollten Ihre Fragen durch die hier zur Verfügung stehenden Informationen nicht beantwortet werden, bieten wir Ihnen durch unsere mit Anwendungsingenieuren besetzten technischen Support-Zentren per Fax und Telefon Hilfe.

Elektronischer Service

Bulletin Board Support

National Instruments hat BBS- und FTP-Sites, die Ihnen rund um die Uhr zur Verfügung stehen und auf denen sich Dateien und Dokumente mit Antworten zu den am häufigsten vorkommenden Kundenfragen befinden. Von diesen Sites können Sie auch die neuesten Instrumententreiber sowie Updates und Beispielprogramme herunterladen. Aufgezeichnete Anweisungen zur Benutzung des BBS- und FTP-Service sowie automatisierte BBS-Informationen erhalten Sie unter der Tel.Nr. USA 1-512 795 6990. Sie können auf diesen Service folgendermaßen zugreifen:

Vereinigte Staaten: 1-512 794 5422
Bis zu 14.400 Baud, 8 Datenbits, 1 Stoppbit, keine Parität

Großbritannien: 01635 551422
Bis zu 9.600 Baud, 8 Datenbits, 1 Stoppbit, keine Parität

Frankreich: 01 48 65 15 59
Bis zu 9.600 Baud, 8 Daten bits, 1 Stoppbit, keine Parität

FTP-Unterstützung

Melden Sie sich zum Zugriff auf unsere FTP-Site bei unserem Internet-Host `ftp.natinst.com`, `anonymous` an und benutzen Sie Ihre E-Mail-Adresse, wie z.B. `joesmith@anywhere.com` als Paßwort. Die Unterstützungsdateien und -dokumente befinden sich in den Verzeichnissen `/support`.

Faxabruf-Unterstützung

Bei der Faxabfrage handelt es sich um ein Informationssystem, das rund um die Uhr zugänglich ist und eine Bibliothek an Dokumenten zu einem umfassenden Bereich technischer Informationen enthält. Sie können auf die Faxabfrage von einem Telefon mit Tonwahl unter der folgenden Nummer zugreifen: 1-512 418 1111.

Unterstützung per E-Mail

Sie können dem Anwendungstechnikerteam Fragen zur technischen Unterstützung per E-Mail an unsere unten aufgeführte Internet-Adresse zukommen lassen. Bitte vergessen Sie nicht, Ihren Namen, Anschrift und Telefonnummer anzugeben, damit wir Ihnen unsere Lösungen und Vorschläge mitteilen können.

`ni.germany@natinst.com`

Unterstützung per Telefon und Fax

National Instruments unterhält auf der ganzen Welt Niederlassungen. In nachstehender Liste finden Sie die Nummer für technische Unterstützung für Ihr Land. Sollte in Ihrem Land kein Büro von National Instruments bestehen, setzen Sie sich bitte mit der Stelle in Verbindung, von der Sie die Software gekauft haben, für die Sie Unterstützung benötigen.

Land	Telefon	Fax
Australien	03 9879 5166	03 9879 6277
Belgien	02 757 00 20	02 757 03 11
Brasilien	011 288 3336	011 288 8528
Dänemark	45 76 26 00	45 76 26 02
Deutschland	089 741 31 30	089 714 60 35
Finnland	09 725 725 11	09 725 725 55
Frankreich	01 48 14 24 24	01 48 14 24 14
Großbritannien	01635 523545	01635 523154
Hongkong	2645 3186	2686 8505
Israel	03 6120092	03 6120095
Italien	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Kanada (Ontario)	905 785 0085	905 785 0086
Kanada (Québec)	514 694 8521	514 694 4399
Korea	02 596 7456	02 596 7455
Mexiko	5 520 2635	5 520 3282
Niederlande	0348 433466	0348 430673
Norwegen	32 84 84 00	32 84 86 00
Österreich	0662 45 79 90 0	0662 45 79 90 19
Singapur	2265886	2265887
Spanien	91 640 0085	91 640 0533
Schweden	08 730 49 70	08 730 43 70
Schweiz	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
Vereinigte Staaten	512 795 8248	512 794 5678

Formular für technische Unterstützung

Fotokopieren Sie dieses Formular und bringen Sie es jedes Mal, wenn Sie Änderungen an Ihrer Software oder Hardware vornehmen, auf den neuesten Stand. Verwenden Sie die ausgefüllte Kopie dieses Formulars als Nachschlagequelle für Ihre derzeitige Konfiguration. Wenn Sie dieses Formular sorgfältig ausfüllen, bevor Sie sich wegen technischer Unterstützung mit National Instruments in Verbindung setzen, helfen Sie damit unseren Anwendungsingenieuren, Ihre Fragen effizienter zu beantworten.

Sollten Sie Hardware- oder Softwareprodukte von National Instruments verwenden, die mit diesem Problem zusammenhängen, legen Sie bitte Konfigurationsformulare aus den jeweiligen Handbüchern bei. Verwenden Sie, wenn nötig, zusätzliche Seiten.

Name _____

Firma _____

Adresse _____

_____

Fax (___) _____Tel. (___) _____

Computermarke_____ Modell _____Prozessor_____

Betriebssystem (inkl. Versionsnr.) _____

Taktgeschw. _____MHz RAM _____MB Anzeigeadapter_____

Maus ___ja ___nein Andere installierte Adapter_____

Festplatten-Kapazität _____MB Marke_____

Verwendete Instrumente _____

_____

National Instruments Hardware-Produktmodell _____ Revision _____

Konfiguration _____

National Instruments Softwareprodukt _____ Version _____

Konfiguration _____

Beschreibung des Problems: _____

_____

_____

_____

_____

Bitte alle Fehlermeldungen aufführen: _____

_____

_____

Durch folgende Schritte wird das Problem wiederholt: _____

_____

_____

_____

_____

LabVIEW-Hardware- und Software-Konfigurationsformular

Notieren Sie die Einstellungen und Revisionen Ihrer Hardware und Software, die sich in der Zeile rechts von jedem Artikel befindet. Füllen Sie bei jeder Änderung an Ihrer Software- oder Hardware-Konfiguration jedes Mal eine neue Kopie dieses Formulars aus, und benutzen Sie dieses Formular als Nachschlagequelle für Ihre derzeitige Konfiguration. Wenn Sie dieses Formular sorgfältig ausfüllen, bevor Sie sich wegen technischer Unterstützung mit National Instruments in Verbindung setzen, helfen Sie damit unseren Anwendungsingenieuren, Ihre Fragen effizienter zu beantworten.

National Instruments Produkte

DAQ-Hardware _____

Interrupt-Level der Hardware _____

DMA-Kanäle der Hardware _____

Basis-I/O-Adresse der Hardware _____

Programmierungswahl _____

HiQ-, NI-DAQ-, LabVIEW- oder LabWindows-Version _____

Andere Karten in Ihrem System _____

Basis-I/O-Adresse anderer Karten _____

DMA-Kanäle anderer Karten _____

Interrupt-Level anderer Karten _____

Andere Produkte

Computer-Hersteller- und Modell _____

Mikroprozessor _____

Taktfrequenz oder -geschwindigkeit_____

Typ der installierten Videokarte _____

Betriebssystemversion _____

Betriebssystemmodus _____

Programmiersprache _____

Programmiersprachenversion _____

Andere Karten in Ihrem System _____

Basis-I/O-Adresse anderer Karten _____

DMA-Kanäle anderer Karten _____

Interrupt-Level anderer Karten _____

Formular für Kommentare zur Dokumentation

National Instruments bittet Sie um Ihre Meinung zu der mit unseren Produkten mitgelieferten Dokumentation. Mit Hilfe dieser Informationen sind wir besser in der Lage, die von Ihnen benötigten Qualitätsprodukte zu liefern.

Titel: *LabVIEW™ Benutzerhandbuch*

Ausgabe: Juli 1998

Artikelnummer: 321200B-01

Ihre Meinung zu Vollständigkeit, Klarheit und Aufbau des Handbuchs.

_____
_____
_____
_____
_____
_____
_____

Falls Sie im Handbuch Fehler entdecken, geben Sie bitte die Seite an und beschreiben Sie den Fehler.

_____
_____
_____
_____
_____
_____
_____

Wir bedanken uns für Ihre Hilfe.

Name _____

Titel _____

Firma _____

Adresse _____

_____

Tel. (___) _____ Fax (___) _____

E-Mail-Adresse _____

Senden an: Technical Publications
National Instruments Germany GmbH
Konrad-Celtis-Str. 79
81369 München

Fax an: Technical Publications
National Instruments Germany GmbH
(089) 714 60 35

Glossar

Präfix	Bedeutung	Wert
n-	nano-	10^{-9}
µ-	mikro-	10^{-6}
m-	milli-	10^{-3}

Zahlen/Symbole

•	Unendlich
π	Pi
D	Delta. Differenz. Δx bezeichnet den Wert, um den sich *x* von einem Index zum nächsten verändert.
1D	Eindimensional
2D	Zweidimensional

A

A/D	Analog/Digital
Ablaufinvariante Ausführung	Ein Modus, in dem mehrere Instanzen eines SubVIs zur parallelen Ausführung bei distinkter und separater Speicherung der Daten aufgerufen werden können.
Absoluter Pfad	Relativer Datei- oder Verzeichnispfad, durch den ein Speicherort in Bezug auf die oberste Ebene des Dateisystems beschrieben wird.
Aktives Fenster	Ein Fenster, das bereit ist, Eingaben des Benutzers entgegenzunehmen; in der Regel handelt es sich dabei um das vorderste Fenster. Die Titelleiste eines aktiven Fensters ist hervorgehoben. Sie können ein Fenster als aktives Fenster auswählen, indem Sie darauf klicken oder indem Sie es im Menü **Fenster** wählen.

Aktuelles VI	Ein VI, dessen Frontpanel, Blockdiagramm oder Icon-Editor als aktuelles Fenster geöffnet ist.
Anschluß	Ein Teil eines VIs oder eines Funktionsknotens mit Eingangs- und Ausgangsterminals. Die Daten werden durch einen Anschluß zum Knoten und wieder zurück geleitet.
Anschlußfeld	Ein Bereich in der oberen rechten Ecke eines Frontpanel-Fensters, in dem das VI-Terminalmuster angezeigt wird. Das Anschlußfeld liegt unter dem Icon-Feld.
ANSI	American National Standards Institute (Nationales amerikanisches Institut für Standards)
Anzeige	Ein Frontpanel-Objekt, das Ausgabewerte anzeigt.
Array	Geordnete, indizierte Gruppe von Datenelementen desselben Typs.
Array-Shell	Frontpanel-Objekt, das ein Array enthält. Eine Array-Shell besteht aus einer Indexanzeige, einem Datenobjekt-Fenster und einem optionalen Label. Sie kann verschiedene Datentypen aufnehmen.
ASCII	American Standard Code for Information Interchange (Amerikanischer Standardcode für den Informationsaustausch)
Asynchrone Ausführung	Ein Modus, in dem mehrere Verfahren einen Prozessor zur selben Zeit gemeinsam nutzen. Dabei wird z.B. ein Vorgang ausgeführt, während andere Vorgänge auf Interrupts im Geräte-I/O warten oder während auf ein Taktsignal gewartet wird.
ATE	Automatic Test Equipment (Automatisches Testgerät)
Aufrufen eines Popup-Menüs	Das Öffnen eines speziellen kontextsensitiven Menüs durch Klicken auf ein Objekt mit der rechten Maustaste (Windows) oder durch Klicken bei gedrückter Befehlstaste (Macintosh).
Auto-Indizierung	Die Fähigkeit von Schleifenstrukturen, Arrays an ihren Rändern zu disassemblieren und zu assemblieren. Wenn ein Array bei aktivierter Auto-Indizierung in eine Schleife eingeführt wird, disassembliert die Schleife das Array automatisch. Dabei werden Skalare aus eindimensionalen Arrays extrahiert, eindimensionale Arrays aus zweidimensionalen Arrays usw. In Umkehrung desselben Verfahrens assemblieren Schleifen Daten in Arrays, während sie aus der Schleife austreten.

Automatische Größenanpassung	Die automatische Anpassung der Größe von Labels entsprechend des eingegebenen Textes.
Automatische Skalierung	Die Fähigkeit von Skalen, eine automatische Anpassung an den Bereich der Plot-Werte durchzuführen. Bei Skalen für Graphen wird durch diese Funktion auch der Maximal- und Minimalwert für die Skala festgelegt.

B

Bedienwerkzeug	Ein Werkzeug, das zur Eingabe von Daten in Bedienelemente und zur Benutzung dieser Bedienelemente dient. Es ähnelt einer Hand mit einem ausgestreckten Zeigefinger.
Bedingungsanschluß	Ein Terminal einer While-Schleife, das einen Booleschen Wert enthält, durch den bestimmt wird, ob das VI eine weitere Iteration ausführen soll.
Benutzer	*Siehe* Operator.
Benutzerdefinierte Konstante	Ein Blockdiagramm-Objekt, das einen von Ihnen eingestellten Wert ausgibt.
Benutzerdefinierte PICT-Bedien- und Anzeigeelemente	Bedien- und Anzeigeelemente, deren Einzelteile durch Grafiken und Anzeigen ersetzt werden können, die der Benutzer bereitstellt.
Beschreibungsfeld	Ein Dialogfeld, das eine Online-Beschreibung für ein G-Objekt enthält.
Beschriftungswerkzeug	Ein Werkzeug, das zum Erstellen von Labels und zur Eingabe von Text in Textfenstern verwendet wird.
Blockdiagramm	Verbildlichte Beschreibung oder Darstellung eines Programms oder Algorithmus. In G entspricht das Blockdiagramm dem Quellcode für das VI. Es besteht aus ausführbaren Icons, die als Knoten bezeichnet werden, und Verbindungen, die Daten zwischen den Knoten übertragen.
BNF	Backus-Naur-Form. Eine gebräuchliche Notationsform für die Syntax von Computersprachen.
Boolesche Bedien- und Anzeigeelemente	Frontpanel-Objekte, die zum Manipulieren und Anzeigen von Booleschen Daten (TRUE oder FALSE) verwendet werden. Es stehen mehrere Stile zur Auswahl, z.B. Schalter, Schaltflächen und LEDs.

Breakpoint	Auch Haltepunkt. Der Punkt, an dem die Ausführung angehalten wird, wenn ein SubVI aufgerufen wird. Sie können einen Breakpoint einrichten, indem Sie mit dem Breakpoint-Werkzeug aus der Palette **Werkzeuge** auf ein VI, einen Knoten oder eine Verbindung klicken.
Breakpoint-Werkzeug	Ein Werkzeug, mit dem ein Breakpoint auf einem VI, auf einem Knoten oder einer Verbindung eingerichtet wird.
Bündelknoten	Eine Funktion, die Cluster aus verschiedenen Elementtypen zusammenstellt.
Byte-Stream-Datei	Eine Datei, in der Daten als Sequenz von ASCII-Zeichen oder Bytes gespeichert werden.

C

Case	Ein Unterdiagramm einer Case-Struktur.
Case-Struktur	Struktur zur bedingungsabhängigen Verzweigungssteuerung, wobei je nach Eingabe jeweils nur ein Unterdiagramm ausgeführt wird. Sie entspricht der Kombination der Aussagen IF, THEN, ELSE und CASE in Steuerstruktur-Sprachen.
CIN	*Siehe* Code Interface Node.
Cloning	Das Anfertigen einer Kopie eines Bedienelements oder eines anderen G-Objekts. Dabei klicken Sie das Objekt zuerst an, dann drücken Sie <Strg> (Windows), <Option> (Macintosh), <Meta> (Sun) oder <Alt> (HP-UNIX) und ziehen die Kopie schließlich an eine neue Stelle.
Cluster	Ein Satz geordneter, nicht indizierter Datenelemente eines beliebigen Datentyps (numerischer Wert, Boolescher Wert, String, Array oder Cluster). Bei den Elementen muß es sich entweder ausschließlich um Bedienelemente oder um Anzeigeelemente handeln.
Cluster-Shell	Frontpanel-Objekt, das die Elemente eines Clusters enthält.
Code Interface Node	CIN. Ein spezieller Blockdiagrammknoten, über den Sie konventionellen, textgestützten Code mit einem VI verknüpfen können.
CPU	Central Processing Unit (zentrale Verarbeitungseinheit)

D

DAQ	*Siehe* Datenerfassung.
Datei-Refnum	Eine Identifikationsnummer, die G mit einer Datei assoziiert, die von einem Benutzer geöffnet wird. Mit Hilfe der Datei-Refnum kann angegeben werden, daß durch eine Funktion oder ein VI eine bestimmte Operation für eine offene Datei ausgeführt werden soll.
Datenabhängigkeit	Eine Bedingung in einer Datenfluß-Programmiersprache. Dabei kann ein Knoten erst dann ausgeführt werden, wenn Daten von einem anderen Knoten empfangen werden. *Siehe auch* Künstliche Datenabhängigkeit.
Datenerfassung	Auch DAQ (Data Acquisition). Das Verfahren zur Erfassung von Daten, normalerweise von A/D- oder digitalen Eingabe-Einsteckkarten.
Datenfluß	Ein Programmiersystem, das aus ausführbaren Knoten besteht, die nur dann ausgeführt werden, wenn sie alle erforderlichen Eingabedaten erhalten haben, und die nach ihrer Ausführung automatisch Ausgabedaten produzieren. G ist ein Datenflußsystem.
Datenprotokolldatei	Eine Datei, in der Daten als Sequenz von Datensätzen eines einzigen, arbiträren Datentyps gespeichert werden. Sie können diesen Datentyp beim Erstellen der Datei festlegen. Alle Datensätze in einer Datenprotokolldatei müssen zwar in einem einzigen Datentyp vorliegen; dieser Datentyp kann jedoch eine komplexe Struktur haben. Sie können z.B. festlegen, daß jeder Datensatz aus einem Cluster besteht, das einen String, eine Zahl und ein Array enthält.
Datenprotokollierung	Das Erfassen von Daten und gleichzeitige Speichern dieser Daten in einer Datei. Die Datei-I/O-Funktionen von G können Daten protokollieren.
Datenspeicherformate	Die Anordnung und Darstellung von Daten in einem Speicher.
Datentyp	Das Format, in dem Informationen vorliegen. In BridgeVIEW werden die folgenden Datentypen zur Tag-Konfiguration akzeptiert: Analog, Diskret, Bit-Array und String. In LabVIEW werden für die meisten Funktionen die folgenden Datentypen akzeptiert: Numerisch, Array, String und Cluster.
Datentyp-Deskriptor	Ein Code zur Identifikation von Datentypen, der bei der Speicherung und Darstellung von Daten verwendet wird.

Dauerausführung	Der Ausführmodus, bei dem ein VI so lange ausgeführt wird, bis der Benutzer es anhält. Die Dauerausführung wird durch Klicken der Schaltfläche **Dauerausführung** eingeleitet.
Dauerausführungs-schaltfläche	Ein Icon, durch das der Ausführstatus eines VIs angezeigt wird.
DDE	*Siehe* Dynamischer Datenaustausch.
Diagramm	*Siehe* Oszilloskop-Diagramm, Strip-Diagramm und Sweep-Diagramm.
Dialogfeld	Ein Fenster, das angezeigt wird, wenn in einer Anwendung zusätzliche Informationen zum Ausführen eines Befehls benötigt werden.
Dimension	Größen- und Strukturattribut eines Arrays.
DUT	Device Under Test (Gerät im Test).
Dynamischer Datenaustausch	Auch DDE (Dynamic Data Exchange). Das Übertragen von Daten zwischen Anwendungen, das ohne Eingriff oder Überwachung durch den Benutzer erfolgt.

E

Echtzeit	Auch Real-Time. Bezieht sich auf die Durchführung von Berechnungen in der tatsächlichen Zeitspanne, die für den Ablauf des damit zusammenhängenden physischen Prozesses benötigt wird. Die Ergebnisse der Berechnung können also zur Steuerung des physischen Prozesses verwendet werden.
Element, Bedienelement	Frontpanel-Objekt zur interaktiven Eingabe von Daten in ein VI oder zur programmatischen Eingabe von Daten in ein SubVI.
Elementepalette	Eine Palette, die Bedien- und Anzeigeelemente des Frontpanels enthält.
Entwickler	*Siehe* Systementwickler.
EOF	End of File (Dateiende). Zeichenversatz des Dateiendes gegenüber dem Dateianfang (d.h., EOF kennzeichnet die Größe der Datei).
Externe Routine	*Siehe* Gemeinsam genutzte externe Routine.

F

Farbkopierwerkzeug	Ein Werkzeug, mit dem Sie Farben kopieren können, die Sie anschließend mit dem Farbwerkzeug einfügen können.
Farbwerkzeug	Ein Werkzeug, mit dem Sie Vordergrund- und Hintergrundfarben festlegen können.
Fehlermeldung	Hinweis auf eine Software- oder Hardwarefehlfunktion oder auf einen Versuch, nicht akzeptable Daten einzugeben.
FFT	Fast Fourier Transform (Schnelle Fourier-Transformation)
Formatumwandlung	Automatische Umwandlung, die von G zur Änderung der numerischen Darstellung eines Datenelements durchgeführt wird.
Formatumwandlungs-punkt	Grauer Punkt auf einem Terminal, durch den angezeigt wird, daß eines von zwei miteinander verbundenen Terminals so umgewandelt wurde, daß es dem Datentyp des anderen Terminals entspricht.
Formel-Element	Ein Knoten, durch den als Text eingegebene Formeln ausgeführt werden. Formel-Elemente sind besonders nützlich bei langen Formeln, die nur mühsam in Blockdiagrammform zusammengestellt werden können.
For-Schleife	Iterative Schleifenstruktur, durch die ein Unterdiagramm für eine bestimmte Anzahl von Wiederholungen ausgeführt wird. Entspricht dem konventionellen Code: `For i = 0 to n – 1, do....`
Freies Label	Ein Label auf dem Frontpanel oder Blockdiagramm, das zu keinem Objekt gehört.
Frontpanel	Interaktive Benutzeroberfläche eines VIs. Ähnelt dem Frontpanel oder der Bedienfläche eines echten Instruments und besteht aus Schaltern, Schiebereglern, Drehspulinstrumenten, Graphen, Diagrammen, Runduminstrumenten, LEDs oder anderen Bedien- oder Anzeigeelementen.
Funktion	Integriertes Ausführelement; vergleichbar mit einem Operator, einer Funktion oder einer Aussage in einer konventionellen Sprache.
Funktionspalette	Eine Palette, die Blockdiagrammstrukturen, Konstanten, Kommunikationsfunktionen und VIs enthält.

G

G	Grafische Programmiersprache, die zur Entwicklung von LabVIEW- und BridgeVIEW-Anwendungen verwendet wird.
Geglättete Daten	Daten eines beliebigen Typs, die in einen String umgewandelt wurden; in der Regel mit der Absicht, die Daten in eine Datei zu schreiben.
Gehäuse	Auch Umrandung. Unbeweglicher Teil der Bedien- und Anzeigeelemente eines Frontpanels, der Schieberegler und Skalen enthält.
Gemeinsam genutzte externe Routine	Subroutine, die von mehreren CIN-Coderessourcen gemeinsam genutzt werden kann.
Globale Variable	Ein SubVI mit lokalem Speicher, das ein nicht ablaufinvariantes und ein nicht initialisiertes Schieberegister zur Speicherung von Daten zwischen den einzelnen Ausführungen verwendet. Die Speicherinformationen der Kopien dieser SubVIs stehen zur gemeinsamen Nutzung zur Verfügung und können deswegen zum Übertragen von Daten zwischen den SubVIs verwendet werden.
Glyph	Kleines Bild oder Symbol.
GPIB	General Purpose Interface Bus (Schnittstellenbus für allgemeine Zwecke). Gebräuchlicher Name für das Kommunikations-Schnittstellensystem, das durch die Normen ANSI/IEEE Standard 488.1-1987 und ANSI/IEEE Standard 488.2-1987 definiert ist. Hewlett-Packard, das Unternehmen, das diesen Bus entwickelt hat, nennt ihn HP-IB.
Graph-Bedienelement	Ein Frontpanel-Objekt, das Daten in einer kartesischen Ebene anzeigt.
Größenveränderungs-Handle	Ein winkliger Griff in der Ecke eines Objekts, der auf einen Ziehpunkt zur Größenveränderung hinweist.

H

Handle	Ein Zeiger, der auf einen Zeiger für einen Speicherblock verweist, von dem Referenz-Arrays und -Strings verwaltet werden. Ein Array aus Strings ist ein Handle für einen Speicherblock, der Handles für Strings enthält.
Hex	Hexadezimal. Ein Zahlsystem, das die Zahl 16 als Basis verwendet.
Hierarchische Palette	Ein Menü, das Paletten und Unterpaletten enthält.

Hierarchiefenster	Ein Fenster, das die Hierarchie von VIs und SubVIs in grafischer Form anzeigt.
Highlight-Funktion	Eine Funktion, durch die die Ausführung eines VIs animiert wird. Dadurch wird der Datenfluß im VI illustriert.
Hilfe	Online-Anleitungen, die die Verwendung einer Windows-Anwendung erläutern. Das Menü **Hilfe** zeigt spezielle Hilfethemen an. Durch Drücken auf <F1> können Sie eine Liste der Hilfethemen öffnen.
Hilfefenster	Ein spezielles Fenster, in dem die Namen und Speicherorte der Terminals einer Funktion oder eines SubVIs, die Beschreibung der Bedien- und Anzeigeelemente, die Werte von universalen Konstanten sowie Beschreibungen und Datentypen der Bedienelementattribute angezeigt werden.
Hz	Hertz. Zyklen pro Sekunde.

I

Icon	Grafische Darstellung eines Knotens auf einem Blockdiagramm.
Icon-Editor	Eine Benutzeroberfläche, die der Oberfläche eines Malprogramms ähnelt und zum Erstellen von VI-Icons dient.
Icon-Feld	Ein Bereich in der oberen rechten Ecke des Panel- und Diagrammfensters, in dem das VI-Icon angezeigt wird.
IEEE	Institute for Electrical and Electronic Engineers (Institut der Elektro- und Elektronikingenieure).
Inf	Digitaler Anzeigewert für die Darstellung eines unendlichen Werts mit Hilfe einer Fließkommazahl.
Inplace-Ausführung	Die Fähigkeit einer Funktion oder eines VIs, Speicher wiederholt zu verwenden, anstatt neuen Speicher zuzuteilen.
Instrumententreiber	Ein VI, durch das ein programmierbares Instrument gesteuert wird.

I/O	Eingabe/Ausgabe. Die Übertragung von Daten zu einem Computersystem und zurück über Kommunikationskanäle, Eingabegeräte für Benutzereingaben und/oder Datenerfassungs- und Bedienelementschnittstellen.
Iterationsterminal	Ein Terminal einer For- oder While-Schleife, das die aktuelle Anzahl der ausgeführten Iterationen enthält.

K

Keine Refnum	Vordefinierter Wert, durch den angezeigt wird, daß die Referenznummer ungültig ist.
Kein Pfad	Vordefinierter Wert für das Pfadbedienelement, durch den angezeigt wird, daß der Pfad ungültig ist.
Knoten	Programmausführelement. Knoten entsprechen Aussagen, Operatoren, Funktionen und Subroutinen in konventionellen Programmiersprachen. In einem Blockdiagramm können Knoten aus Funktionen, Strukturen und SubVIs bestehen.
Kompilieren	Ein Verfahren, durch das High-Level-Code in maschinenlesbaren Code umgewandelt wird. VIs werden nach dem Erstellen oder nach einer Änderung automatisch kompiliert, bevor sie zum ersten Mal ausgeführt werden.
Konstante	*Siehe* Universale Konstante und Benutzerdefinierte Konstante.
Kontrollkästchen	Quadratisches Kästchen in einem Dialogfeld, das bei Bedarf markiert werden kann. Durch Kontrollkästchen können in der Regel mehrere Optionen eingestellt werden. Es ist möglich, mehrere Kontrollkästchen in einem Dialogfeld zu markieren.
Künstliche Datenabhängigkeit	Eine Bedingung in einer Datenfluß-Programmiersprache, in der das Eintreffen von Daten und nicht deren Wert die Ausführung eines Knotens auslöst.
Kurvendiagramm	Ein Anzeigeelement, das Datenpunkte in einer bestimmten Geschwindigkeit anzeigt.

L

Label	Textobjekt, das zur Bezeichnung oder Beschreibung anderer Objekte oder Bereiche auf einem Frontpanel oder Blockdiagramm dient.
LabVIEW	Laboratory Virtual Instrument Engineering Workbench. Eine Anwendung für die Programmentwicklung; beruht auf der Programmiersprache G, die häufig zu Test- und Meßzwecken eingesetzt wird.
Laufwerk	Ein Buchstabe von a–z, auf den ein Doppelpunkt (:) folgt; auf diese Weise werden logische Laufwerke bezeichnet.
LED	Leuchtdiode. *Auch* Luminiszenzdiode. (LED = Light-emitting Diode)
Leeres Array	Ein Array, das zwar keine Elemente enthält, für das jedoch ein Datentyp definiert wurde. Ein Array, das ein numerisches Bedienelement in seinem Datenanzeigefenster enthält, aber für kein Element einen definierten Wert aufweist, ist z.B. ein leeres numerisches Array.
Legende	Ein Objekt, das zu einem Diagramm oder Graphen gehört und den Namen und Stil von Plots in diesem Diagramm oder Graphen angibt.
Listenfeld	Ein Feld innerhalb eines Dialogfelds, das alle Wahlmöglichkeiten für einen Befehl aufführt, z.B. eine Namensliste der auf einem Datenträger vorhandenen Dateien. In der Regel wählen Sie ein Element in diesem Listenfeld aus und klicken dann auf **OK**. Falls die gesamte Liste nicht in das verfügbare Feld paßt, steht eine vertikale Bildlaufleiste zur Verfügung. Sie können das Listenfeld öffnen, indem Sie auf den Abwärtspfeil neben dem ersten sichtbaren Element in der Liste klicken.
Lokale Variable	Eine Variable, die Ihnen den Schreib- und Lesezugriff auf ein Bedien- oder Anzeigeelement auf dem Frontpanel Ihres VIs ermöglicht.

M

Marquee	Gestrichelter Rand, der ausgewählte Objekte umfließt.
Matrix	Zweidimensionales Array
MB	Megabytes (Speicher)

Menüleiste	Horizontale Leiste, die die Namen der Hauptmenüs einer Anwendung auflistet. Die Menüleiste erscheint unter der Titelleiste eines Fensters. Jede Anwendung hat ihre eigene charakteristische Menüleiste. Einige Menüs (und Befehle) kommen jedoch in ähnlicher Form in vielen verschiedenen Anwendungen vor.
Mnemonischer String	Ein String, der mit einem ganzzahligen Wert assoziiert ist.
Modulares Programmieren	Eine Art des Programmierens, bei der miteinander austauschbare Computerroutinen verwendet werden.

N

NaN	Digitaler Anzeigewert für die Fließkommadarstellung von *Keine Zahl*; normalerweise wird dieser Wert als Ergebnis einer nicht definierten Operation, wie z.B. log(–1), ausgegeben.
Nichtdarstellbare Zeichen	ASCII-Zeichen, die nicht angezeigt werden können, wie z.B. eine neue Zeile, ein Tabulator usw.
Numerische Bedien- und Anzeigeelemente	Frontpanel-Objekte, die zum Manipulieren und Anzeigen oder zur Eingabe und Ausgabe numerischer Daten dienen.

O

Objekt	Allgemeiner Begriff für Elemente auf dem Frontpanel oder Blockdiagramm, einschließlich von Bedienelementen, Knoten, Verbindungen und importierten Bildern.
Objekt-Popup-Menü-Werkzeug	Ein Werkzeug, das für den Zugriff auf ein kontextsensitives Menü für ein Objekt verwendet wird.
OPC-Server	OLE für die Prozeßsteuerung. Ein Standard auf COM-Basis, der durch die OPC-Grundlage definiert wird und durch den festgelegt wird, wie die Interaktion mit Geräte-Servern durchgeführt wird. COM ist eine 32-Bit-Technologie in Windows.
Ordner	*Siehe* Verzeichnis.
Oszilloskop-Diagramm	Ein numerisches Anzeigeelement, das einem Oszilloskop nachgebildet ist.

P

Palette	Eine Anzeige mit Symbolen für mögliche Optionen.
Pixmap	Standardformat zum Speichern von Bildern, bei dem jedes Pixel durch einen Farbwert repräsentiert wird. Eine Bitmap-Grafik ist eine Schwarzweißversion einer Pixmap-Grafik.
Plattform	Ein Computer und sein Betriebssystem.
Plot	Grafische Darstellung eines Daten-Arrays, die entweder in Form eines Graphen oder eines Diagramms erfolgt.
Polymorphismus	Die Fähigkeit eines Knotens, sich automatisch auf Daten unterschiedlicher Darstellungsformen, Typen oder Strukturen einzustellen.
Popup-Menü	Ein spezielles kontextsensitives Menü, das durch Klicken auf ein Objekt mit der rechten Maustaste (Windows) oder durch Klicken bei gedrückter Befehlstaste (Macintosh) aufgerufen wird. Die vorhandenen Menüoptionen hängen vom jeweiligen Objekt ab.
Positionierwerkzeug	Ein Werkzeug, das zum Verschieben, Auswählen und zur Größenveränderung von Objekten verwendet wird. Es ähnelt einem Pfeil.
Probe	Eine Debugging-Funktion zum Überprüfen von Zwischenwerten in einem VI.
Probe-Werkzeug	Ein Werkzeug, das zum Erstellen von Proben für Verbindungen verwendet wird.
Programmäßiges Drucken	Automatisches Drucken eines VI-Frontpanels nach der Ausführung.
Pseudocode	Vereinfachte, sprachunabhängige Darstellung von Programmiercode.
Pulldown-Menü	Ein Einblendmenü, auf das über eine Menüleiste zugegriffen werden kann. Pulldown-Menüs enthalten in der Regel allgemeine Menüoptionen.

Q

Quellterminal	Auch Datenquelle. Ein Terminal, das Daten ausgibt.

R

Rahmen	Unterdiagramm einer Sequenzstruktur.
Refnum	Die Identifikationsnummer einer DDE-Konversation oder einer offenen Datei, die von damit zusammenhängenden VIs als Referenz verwendet werden kann.
Repräsentation/ Darstellung	Eine Untergruppe des numerischen Datentyps. Repräsentationen schließen vorzeichenbehaftete und vorzeichenlose Byte-, Word- und Long-Integer-Werte sowie reale als auch komplexe Fließkommazahlen in Single-, Double- und Extended-Präzision ein.
Ring-Bedienelement	Spezielles numerisches Bedienelement, das 32-Bit-Integer-Werte (von 0 sequentiell ansteigend) mit einer Reihe von Textbeschriftungen oder Grafiken assoziiert.
Rollwerkzeug	Ein Werkzeug, das das Blättern durch Anwendungsfenster ermöglicht.

S

Schieber	Beweglicher Teil von Schiebebedien- oder -anzeigeelementen.
Schieberegister	Optionaler Mechanismus in Schleifenstrukturen, der dazu dient, den Wert einer Variablen von einer Iteration einer Schleife an eine nachfolgende Iteration weiterzuleiten.
Sensor	Ein Gerät, das durch Spannungs- oder Stromveränderungen auf die Veränderungen einer gemessenen physischen Eigenschaft reagiert und so z.B. Geschwindigkeit, Temperatur oder Durchfluß mißt.
Sequenzstruktur	Eine Programmsteuerstruktur, die ihre Unterdiagramme in numerischer Reihenfolge ausführt. Diese Struktur wird normalerweise dazu verwendet, Knoten, die nicht datenabhängig sind, zur Ausführung in einer bestimmten Reihenfolge zu zwingen.
Sequenz-Variable	Ein Terminal, das Daten zwischen den Rahmen einer Sequenzstruktur überträgt.
Sink-Terminal	Ein Terminal, das Daten absorbiert. Wird auch als Zielterminal oder Datensenke bezeichnet.

Skala	Ein Teil eines mechanischen, Diagramm- oder Graph-Bedienelements bzw. -Anzeigeelements; eine Skala weist eine Reihe von Markierungen oder Punkten in bekannten Abständen auf, durch die Maßeinheiten angezeigt werden.
Skalar	Eine Zahl, die auf einer Skala als Punkt wiedergegeben werden kann. Im Unterschied zu einem Array handelt es sich um einen einzelnen Wert. Skalare Boolesche Werte und skalare Cluster stellen explizit Einzelinstanzen ihres jeweiligen Datentyps dar.
Solutions Gallery	Eine Option im DAQ Solution Wizard, die die Auswahl aus zahlreichen Kategorien gebräuchlicher DAQ-Anwendungen ermöglicht.
Steuerungsfluß	Ein Programmiersystem, bei dem die Ausführreihenfolge durch die sequentielle Anordnung von Anweisungen bestimmt wird. Die meisten konventionellen, textorientierten Programmiersprachen, wie z.B. C, Pascal und BASIC, sind Steuerstruktursprachen.
String-Bedien- und -Anzeigeelemente	Frontpanel-Objekte, die zum Manipulieren und Anzeigen oder zur Eingabe und Ausgabe von Text dienen.
Strip-Diagramm	Eine numerische Plot-Anzeige, die einem Aufzeichnungsgerät für Papierstreifendiagramme nachgebildet wurde; beim Aufzeichnen neuer Daten werden die bereits aufgezeichneten Daten aus dem Anzeigebereich verschoben.
Struktur	Ein Programmsteuerelement, z.B. eine Sequenz, ein Case, eine For-Schleife oder eine While-Schleife.
SubVI	Ein VI, das im Blockdiagramm eines anderen VIs verwendet wird. Es ist mit einer Subroutine vergleichbar.
Sweep-Diagramm	Ein numerisches Anzeigeelement, das einem Oszilloskop nachgebildet ist. Das Sweep-Diagramm ähnelt einem Oszilloskop-Diagramm; neue Daten werden jedoch durch eine über den Bildschirm aufende Linie von alten Daten getrennt.
Symbolleiste	Eine Leiste, die Befehlsschaltflächen zum Ausführen und zum Debugging von VIs enthält.
Systementwickler	Der Entwickler einer ausführbaren Anwendungssoftware.

T

Tabellengetriebene Ausführung	Eine Ausführmethode, bei der Einzelaufgaben separate Cases in einer Case-Struktur darstellen, die in eine While-Schleife integriert ist. Sequenzen werden als Arrays aus Case-Nummern gekennzeichnet.
Terminal	Ein Objekt (oder ein Bereich) auf einem Knoten, durch das Daten geleitet werden.
Tip-Strip	Ein Textstreifen, auf dem der Name eines Objekts, eines Bedienelements oder eines Terminals angezeigt wird.
Top-Level-VI	Ein VI an oberster Stelle in der VI-Hierarchie. Dieser Begriff dient zur Unterscheidung des VIs von seinen SubVIs.
Tunnel	Ein Dateneingangs- oder -ausgangsterminal in einer Struktur.
Typ-Deskriptor	*Siehe* Datentyp-Deskriptor.
Typenformung	Das Ändern des Typdeskriptors eines Datenelements, ohne daß das Speicherbild für die Daten geändert wird.

U

Umwandlung	Das Verfahren zur Änderung des Typs eines Datenelements.
Universale Konstante	Nicht bearbeitbares Blockdiagramm-Objekt, das ein besonderes ASCII-Zeichen oder eine numerische Standardkonstante ausgibt, z.B. pi.
Unterbrochene Ausführung (Schaltfläche)	Die Schaltfläche, die anstelle der Ausführschaltfläche angezeigt wird, wenn ein VI aufgrund von Fehlern nicht ausgeführt werden kann. Klicken Sie auf die Schaltfläche, um das Dialogfeld mit der Fehlerliste aufzurufen.
Unterbrochenes VI	Ein VI, das nicht kompiliert oder ausgeführt werden kann. Es ist durch einen unterbrochenen Pfeil auf der Ausführschaltfläche gekennzeichnet.
Unterdiagramm	Ein Blockdiagramm innerhalb der Grenzen einer Struktur.
Unterpalette	Eine Palette, die im Icon einer anderen Palette enthalten ist.
UUT	Unit Under Test (Einheit im Test)

V

Verbindung	Ein Datenpfad zwischen Knoten. *Siehe auch* Datenfluß.
Verbindungskreuzungen	Ein Punkt, an dem drei oder mehr Verbindungssegmente aufeinandertreffen.
Verbindungssegment	Ein einzelnes horizontales oder vertikales Verbindungsstück.
Verbindungswerkzeug	Ein Werkzeug, das zur Definition der Datenpfade zwischen Quell- und Sink-Terminals verwendet wird.
Verbindungszweig	Ein Abschnitt einer Verbindung, der alle Verbindungssegmente von einer Verbindungskreuzung zur nächsten Kreuzung, von einem Terminal zur nächsten Kreuzung oder zwischen zwei Terminals, zwischen denen sich keine Kreuzungen befinden, enthält.
Verzeichnis	Eine Struktur zur übersichtlichen Organisation von Dateien in zusammengehörige Gruppen. Ein Verzeichnis ist mit einer Adresse vergleichbar, die angibt, wo sich die Dateien befinden. Ein Verzeichnis kann Dateien oder Unterverzeichnisse mit Dateien enthalten.
VI	*Siehe* Virtuelles Instrument.
VI-Bibliothek	Eine spezielle Datei, die eine Ansammlung verwandter VIs für einen speziellen Zweck enthält.
VI-Gerüst	Ein arbeitsunfähiger Prototyp eines SubVIs. Ein VI-Gerüst verfügt zwar über Ein- und Ausgänge, ist jedoch unvollständig. Es wird in der ersten Planungsphase für ein VI-Design als Platzhalter für die später zu entwickelnden VIs eingesetzt.
Virtual Instrument Software Architecture	Eine Einzelschnittstellenbibliothek zur Steuerung von GPIB-, VXI-, RS-232- und ähnlichen Instrumenten.
Virtuelles Instrument	VI. Ein Programm in der grafischen Programmiersprache G; die Bezeichnung weist darauf hin, daß das Programm dem Erscheinungsbild und der Funktionsweise eines realen Instruments nachgebildet ist.
VISA	*Siehe* Virtual Instrument Software Architecture.
VXI	VME eXtensions for Instrumentation (Bus)

W

Werkzeug	Ein spezieller Cursor, den Sie für spezielle Operationen einsetzen können.
Werkzeugpalette	Eine Palette, die die Werkzeuge für das Bearbeiten und das Debugging von Frontpanel- und Blockdiagramm-Objekten enthält.
While-Schleife	Eine Schleifenstruktur, die einen Codeabschnitt so lange wiederholt, bis eine festgelegte Bedingung erfüllt ist. Die Schleifenstruktur ist vergleichbar mit einer Do-Schleife oder einer Repeat-Until-Schleife in konventionellen Programmiersprachen.

Z

Zählterminal	Ein Terminal in einer For-Schleife, durch dessen Wert bestimmt wird, wie oft eine For-Schleife ihr Unterdiagramm ausführt.
Zweidimensional	Ein Objekt mit zwei Dimensionen, z.B. ein Array, das Reihen und Spalten enthält.

Stichwörterverzeichnis

Zahlen

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z