LabVIEW™

Benutzerhandbuch

Ausgabe Juni 2003
Artikelnummer 321200E-01

Deutschsprachige Niederlassungen

National Instruments
Germany GmbH
Konrad-Celtis-Straße 79
81369 München
Tel.: +49 (0) 89 741 31 30
Fax: +49 (0) 89 714 60 35

National Instruments
Ges.m.b.H.
Plainbachstraße 12
5101 Salzburg-Bergheim
Tel.: +43 0 662 45 79 90 0
Fax: +43 0 662 45 79 90 19

National Instruments
Switzerland
Sonnenbergstraße 53
CH-5408 Ennetbaden
Tel.: +41 56 200 51 51, 41 21 320 51 51 (Lausanne)
Fax: +41 56 200 51 55

Lokaler technischer Support

Deutschland:	`ni.germany@ni.com`	`www.ni.com/germany`
Österreich:	`ni.austria@ni.com`	`www.ni.com/austria`
Schweiz:	`ni.switzerland@ni.com`	`www.ni.com/switzerland`

Technischer Support und Produktinformation weltweit

`ni.com`

National Instruments Corporate Firmenhauptsitz

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 001 512 683 0100

Internationale Niederlassungen

Australien 1800 300 800, Belgien 32 0 2 757 00 20, Brasilien 55 11 3262 3599, China 86 21 6555 7838, Dänemark 45 45 76 26 00, Finnland 385 0 9 725 725 11, Frankreich 33 0 1 48 14 24 24, Griechenland 30 2 10 42 96 427, Großbritannien 44 0 1635 523545, Indien 91 80 51190000, Israel 972 0 3 6393737, Italien 39 02 413091, Japan 81 3 5472 2970, Kanada (Calgary) 403 274 9391, Kanada (Montreal) 514 288 5722, Kanada (Ottawa) 613 233 5949, Kanada (Québec) 514 694 8521, Kanada (Toronto) 905 785 0085, Kanada (Vancouver) 514 685 7530, Korea 82 02 3451 3400, Malaysia 603 9131 0918, Mexiko 001 800 010 0793, Neuseeland 1800 300 800, Niederlande 31 0 348 433 466, Norwegen 47 0 66 90 76 60, Polen 48 0 22 3390 150, Portugal 351 210 311 210, Russland 7 095 238 7139, Schweden 46 0 8 587 895 00, Singapur 65 6226 5886, Slovenien 386 3 425 4200, Spanien 34 91 640 0085, Südafrika 27 0 11 805 8197, Taiwan 886 2 2528 7227, Thailand 662 992 7519, Tschechische Republik 420 2 2423 5774

Weitere Informationen finden Sie im Anhang unter *Technische Unterstützung und professioneller Service*. Wenn Sie Vorschläge oder Kritik zur Dokumentation haben, senden Sie diese per Email an: `techpubs@ni.com`.

Wichtige Informationen

Garantie

National Instruments gewährt für LabVIEW eine Garantie für einen Zeitraum von 90 ab Versand (nachweisbar durch Lieferschein oder andere Dokumente) gegen Material- und Verarbeitungsfehler. National Instruments wird ein Produkt, das sich innerhalb des Garantiezeitraums als fehlerhaft erweist, nach eigener Wahl reparieren oder ersetzen. Diese Garantie schließt Ersatzteile und Arbeitszeit ein.

Für die Datenträger, auf denen Sie die Software von National Instruments erhalten, wird für den Zeitraum von 90 Tagen nach Erhalt der Lieferung (nachweisbar durch Lieferschein oder andere Dokumente) garantiert, dass sie keine Material- oder Verarbeitungsfehler aufweisen, die die Ausführung der Programmieranweisungen behindern. Wird National Instruments während der Garantiezeit über bestehende Schäden informiert, so wird National Instruments nach eigener Wahl Software-Datenträger, auf denen die Ausführung der Programmieranweisungen nicht möglich ist, entweder reparieren oder ersetzen. National Instruments leistet keine Gewähr dafür, dass die Ausführung der Software zu jeder Zeit oder fehlerfrei erfolgen kann.

Einsendungen werden nur dann zur Garantiebearbeitung angenommen, wenn sie deutlich auf der Außenseite durch eine Autorisierungsnummer für die Rücksendung, eine sogenannte RMA-Nummer (Return Material Authorization), gekennzeichnet sind. National Instruments übernimmt die Versandkosten für Teile, die im Rahmen einer Garantieleistung an den Kunden zurückgesandt werden.

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

Soweit in dieser Garantieerklärung nicht ausdrücklich vorgesehen, übernimmt National Instruments weder ausdrücklich noch stillschweigend irgendeine Gewähr. Insbesondere wird keine Gewähr für marktgängige Qualität oder die Eignung für einen bestimmten Zweck übernommen. Schadenersatzansprüche 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, entgangenen Gewinn, durch die Einschränkung der Verwendbarkeit der Produkte oder durch mittelbare Schäden oder Folgeschäden entstehen. Dies gilt auch dann, wenn National Instruments auf die Möglichkeit solcher Schäden hingewiesen wurde. Diese Einschränkung der Haftung von National Instruments gilt für alle Arten von Schadenersatzansprüchen, sei es aufgrund Vertrags oder unerlaubter Handlung, und gilt auch bei Verschulden. Gerichtliche Schritte gegen National Instruments müssen innerhalb eines Jahres nach Entstehen des Anspruchs eingeleitet werden. National Instruments ist nicht für die Verzögerung von Leistungen haftbar, die durch Vorgänge verursacht werden, über die National Instruments bei vernünftiger Betrachtung keine Kontrolle ausüben kann. Vorliegende Garantieerklärung 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 Veränderungen des Produkts, durch Missbrauch oder fahrlässiges Verhalten aufseiten des Benutzers, durch Stromausfälle oder Spannungsstöße, durch Brand, Überschwemmungen, Unfälle, Handlungen Dritter oder andere Vorfälle verursacht werden, die bei vernünftiger Betrachtung nicht kontrolliert werden können.

Copyright

Marken

Patente

Patentinformationen zu Produkten von National Instruments erhalten Sie unter **Hilfe»Patente** in der Software, der Datei `patents.txt` auf Ihrer CD oder über `ni.com/patents`.

Warnhinweise zum Einsatz von National-Instruments-Produkten

(1) Die Softwareprodukte von National Instruments wurden nicht mit Komponenten und Tests für ein Sicherheitsniveau entwickelt, welches für eine Verwendung bei oder in Zusammenhang mit chirurgischen Implantaten oder als kritische Komponenten von lebenserhaltenden Systemen, deren Fehlfunktion bei vernünftiger Betrachtungsweise zu erheblichen Verletzungen von Menschen führen kann, geeignet ist.

(2) Bei jeder Anwendung, einschließlich der oben genannten, kann die Zuverlässigkeit der Funktion der Softwareprodukte durch entgegenwirkende Faktoren, einschließlich zum Beispiel Spannungsunterschieden bei der Stromversorgung, Fehlfunktionen der Computer-Hardware, fehlende Eignung der Software für das Computerbetriebssystem, fehlende Eignung von Übersetzungs- und Entwicklungssoftware, die zur Entwicklung einer Anwendung eingesetzt werden, Installationsfehler, Probleme bei der Software- und Hardwarekompatibilität, Funktionsstörungen oder Ausfall der elektronischen Überwachungs- oder Kontrollgeräte, vorübergehende Fehler der elektronischen Systeme (Hardware und/oder Software) unvorhergesehener Einsatz oder Missbrauch sowie Fehler des Anwenders oder des Anwendungsentwicklers (entgegenwirkende Faktoren wie diese werden nachstehend zusammenfassend "Systemfehler" genannt) beeinträchtigt werden. Jede Anwendung, bei der ein Systemfehler ein Risiko für Sachwerte oder Personen darstellt (einschließlich der Gefahr körperlicher Schäden und Tod), sollte aufgrund der Gefahr von Systemfehlern nicht lediglich auf eine Form von elektronischem System gestützt werden. Um Schäden und unter Umständen tödliche Verletzungen zu vermeiden, sollte der Nutzer oder Anwendungsentwickler angemessene Sicherheitsmaßnahmen ergreifen, um Systemfehlern vorzubeugen. Hierzu gehören unter anderem Sicherungs- oder Abschaltmechanismen. Da jedes Endnutzersystem den Kundenbedürfnissen angepasst ist und sich von dem Testumfeld unterscheidet, und da ein Nutzer oder Anwendungsentwickler Softwareprodukte von National Instruments in Verbindung mit anderen Produkten in einer von

National Instruments nicht getesteten oder vorhergesehenen Form einsetzen kann, trägt der Nutzer beziehungsweise der Anwendungsentwickler die letztendliche Verantwortung für die Überprüfung und Bewertung der Eignung von National Instruments Produkten, wenn Produkte von National Instruments in ein System oder eine Anwendung integriert werden. Dies erfordert unter anderem die entsprechende Entwicklung und Verwendung sowie Einhaltung einer entsprechenden Sicherheitsstufe bei einem solchen System oder einer solchen Anwendung.

Inhaltsverzeichnis

Über dieses Handbuch

Teil I
LabVIEW-Konzepte

Kapitel 1
Einführung in LabVIEW

Kapitel 2
Einführung in die Welt der virtuellen Instrumente

Kapitel 3
LabVIEW-Umgebung

Kapitel 4
Erstellen des Frontpanels

Kapitel 5
Erstellen des Blockdiagramms

Kapitel 6
Ausführen von VIs und Fehlersuche

Kapitel 7
Erstellen von VIs und SubVIs

Teil II
Erstellen und Bearbeiten von VIs

Kapitel 8
Schleifen und Strukturen

Kapitel 9
Ereignisgesteuerte Programmierung

Kapitel 10
Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern

Kapitel 11
Lokale und globale Variablen

Kapitel 12
Graphen und Diagramme

Kapitel 13
VIs für Grafiken und Sound

Kapitel 14
Datei-I/O

Kapitel 15
Dokumentieren und Drucken von VIs

Kapitel 16
Anpassen von VIs

Kapitel 17
Programmatische Steuerung von VIs

Kapitel 18
Arbeiten mit LabVIEW im Netzwerk

Kapitel 19
Windows-Konnektivität

Kapitel 20
Aufrufen von Code aus textbasierten Programmiersprachen

Kapitel 21
Formeln und Gleichungen

Anhang A
Die Struktur von LabVIEW

Anhang B
Polymorphe Funktionen

Anhang C
Vergleichsfunktionen

Anhang D
Technische Unterstützung und professioneller Service

Glossar

Stichwortverzeichnis

Über dieses Handbuch

In diesem Handbuch finden Sie eine Erläuterung zur grafischen Programmierumgebung von LabVIEW, zu Techniken für das Erstellen von Applikationen in LabVIEW, wie beispielsweise Prüf- und Messapplikationen oder Applikationen für die Datenerfassung, Gerätesteuerung, Datenprotokollierung, Messdatenanalyse bzw. Reporterzeugung.

Außerdem lernen Sie die Komponenten der LabVIEW-Programmierung kennen. Dazu zählen die Benutzeroberfläche, die Arbeitsbereiche sowie die Paletten und Werkzeuge. Das vorliegende Handbuch soll allgemeine Kenntnisse über LabVIEW vermitteln und enthält daher keine spezifischen Informationen zu den einzelnen Paletten, Werkzeugen, Menüs, Dialogfeldern, Steuerelementen oder den integriertes VIs und Funktionen. Nähere Einzelheiten zu diesen Elementen sowie detaillierte schrittweise Anleitungen zur Verwendung der LabVIEW-Komponenten und zum Erstellen spezieller Applikationen entnehmen Sie bitte der *LabVIEW-Hilfe*. Weitere Informationen zur *LabVIEW-Hilfe* und den Zugriff auf die Hilfe finden Sie im Abschnitt *LabVIEW-Dokumentation* des Kapitels 1, *Einführung in LabVIEW*.

Das *LabVIEW Benutzerhandbuch* steht auch im PDF-Format (Portable Document Format) zur Verfügung. Wenn Sie die Installations-Option **Vollständig** wählen, werden PDF-Versionen aller LabVIEW Handbücher installiert. Sie können diese Handbücher öffnen, indem Sie aus der Menüleiste in LabVIEW **Hilfe»Suchen in der LabVIEW-Bibliothek** wählen.

Hinweis Zur Anzeige der PDF-Dateien muss Adobe Acrobat Reader 5.0.5 (oder eine aktuellere Version) mit Suchfunktion installiert sein. Diese Software kann auch von der Internetseite von Adobe Systems unter `www.adobe.de` heruntergeladen werden.

Sie können auch von der *LabVIEW-Hilfe* auf die PDF-Dateien zugreifen, die Sie dafür vorher installieren müssen. Weitere Informationen zum Zugriff auf die PDF-Dateien in der *LabVIEW-Bibliothek* finden Sie im Abschnitt *LabVIEW-Dokumentation* des Kapitels 1, *Einführung in LabVIEW*.

Gliederung dieses Handbuchs

Das *LabVIEW Benutzerhandbuch* besteht aus zwei Teilen. Im Teil I, *LabVIEW-Konzepte*, werden Programmierkonzepte zum Erstellen von Applikationen in LabVIEW erörtert. Die Kapitel in diesem Teil geben eine Einführung in die LabVIEW-Programmierumgebung und helfen bei der Planung einer Applikation.

Im Teil II, *Erstellen und Bearbeiten von VIs*, werden die LabVIEW-Komponenten, -VIs und -Funktionen beschrieben, mit denen Sie eine Applikation gestalten. Die Kapitel in diesem Teil beschreiben die Einsatzmöglichkeiten jeder einzelnen LabVIEW-Komponente, sowie die einzelnen Klassen von VIs und Funktionen.

Symbole und Darstellungen

In diesem Handbuch werden die folgenden Symbole und Darstellungen verwendet:

» Das Symbol » kennzeichnet die Reihenfolge, in der Menübefehle und Dialogfeldoptionen auszuführen sind. So wird zum Beispiel mit der Abfolge **Datei»Seiteneinstellungen»Optionen** angezeigt, dass zunächst das Menü **Datei** zu öffnen ist, hieraus die Option **Seiteneinstellungen** und daraus der Befehl **Optionen** zu wählen ist.

Dieses Symbol kennzeichnet einen Tipp mit wertvollen Hinweisen.

Dieses Symbol steht für einen Hinweis und macht Sie auf wichtige Informationen aufmerksam.

Dieses Symbol kennzeichnet eine Warnung vor möglichen Verletzungen, Datenverlust oder Systemabsturz.

fett In fettgedruckter Schrift sind Elemente dargestellt, die ausgewählt oder angeklickt werden müssen, wie beispielsweise Menüelemente oder Dialogfeldoptionen. Bei fettgedrucktem Text kann es sich auch um Parameternamen, Bedienelemente und Tasten auf dem Frontpanel, Dialogfelder, Bereiche in Dialogfeldern, Menünamen und Palettennamen handeln.

`fett gesperrt` Mit gesperrt dargestelltem Text im Fettdruck sind auf dem Bildschirm ausgegebene Meldungen gekennzeichnet. Außerdem finden Sie in dieser Darstellungsart Befehlszeilen, die sich von anderen Beispielen unterscheiden.

`gesperrt`	Im Sperrdruck sind neben Textstellen oder Zeichen, die über die Tastatur einzugeben sind, Code-Abschnitte, Programmierbeispiele und Syntaxbeispiele dargestellt. Daneben wird diese Darstellungsweise für Laufwerke, Pfade, Verzeichnisse, Programme, Unterprogramme, Subroutinen, Gerätenamen, Funktionen, Operationen, Variablen, Dateinamen und -erweiterungen sowie für Ausschnitte aus Programmcodes verwendet.
`gesperrt kursiv`	An Textstellen, die im kursiven Sperrdruck dargestellt sind, sind Eingaben des Anwenders erforderlich.
kursiv	Kursiv gedruckt sind Variablen, Hervorhebungen, Querverweise oder Einführungen in wichtige Begriffe. Außerdem sind in diesem Schriftstil Textstellen gekennzeichnet, an denen ein Wort oder ein korrekter Wert einzusetzen ist.
Plattform	Text in dieser Schrift zeigt an, dass der darauffolgende Text nur für die bezeichnete Plattform gilt.
Rechter Mausklick	**(Mac OS)** Ein Mausklick bei gedrückter <Befehlstaste> hat dieselbe Wirkung wie ein Klick mit der rechten Maustaste.

Teil I

LabVIEW-Konzepte

In diesem Teil des Handbuchs werden die Begriffe erläutert, die Sie zum Erstellen von Applikationen in LabVIEW benötigen. Die Kapitel in diesem Abschnitt sollen Ihnen einen Einblick in die LabVIEW-Programmierumgebung verschaffen und bei der Planung Ihrer Applikation behilflich sein.

Teil I, *LabVIEW-Konzepte*, enthält folgende Kapitel:

- In Kapitel 1, *Einführung in LabVIEW*, erfahren Sie mehr über LabVIEW, die dazugehörige umfassende Dokumentation sowie die Werkzeuge, mit denen VIs entworfen und erstellt werden können.
- In Kapitel 2, *Einführung in die Welt der virtuellen Instrumente*, erhalten Sie einen Überblick über die Komponenten von virtuellen Instrumenten (VIs).
- In Kapitel 3, *LabVIEW-Umgebung*, werden die LabVIEW-Paletten, -Werkzeuge und -Menüs beschrieben, mit denen Frontpanels und Blockdiagramme von VIs erstellt werden können. Außerdem lesen Sie in diesem Kapitel, wie LabVIEW-Paletten benutzerspezifisch angepasst werden und welche Optionen Ihnen für die Arbeitsumgebung zur Verfügung stehen.
- In Kapitel 4, *Erstellen des Frontpanels*, wird die Erstellung eines VI-Frontpanels erläutert.
- In Kapitel 5, *Erstellen des Blockdiagramms*, wird gezeigt, wie ein VI-Blockdiagramm erzeugt wird.
- In Kapitel 6, *Ausführen von VIs und Fehlersuche*, finden Sie Informationen dazu, welche Einstellungen zur VI-Ausführung möglich sind und wie Probleme bei der Anordnung der Blockdiagrammelemente oder der Übergabe der Daten zwischen den einzelnen Elementen erkannt werden können.

- In Kapitel 7, *Erstellen von VIs und SubVIs*, erfahren Sie, wie Sie eigene VIs und SubVIs erstellen können, Ihre VIs anderen Anwendern zugänglich machen und wie Sie ausführbare Anwendungen oder Shared Libraries (DLLs) erzeugen.

1

Einführung in LabVIEW

LabVIEW ist eine grafische Programmiersprache, die Symbole anstelle von Textzeilen verwendet, um Applikationen zu erstellen. Im Gegensatz zu textbasierten Programmiersprachen, bei denen die Programmausführung von Anweisungen gesteuert wird, ist LabVIEW eine datenflussorientierte Programmiersprache, das heißt, die Ausführung wird durch den Datenfluss gesteuert.

Zunächst erstellen Sie mit Hilfe von Werkzeugen und Objekten in LabVIEW eine Benutzeroberfläche. Die Benutzeroberfläche wird als Frontpanel bezeichnet. Sie fügen Code hinzu, indem Sie grafische Darstellungen von Funktionen zum Steuern der Frontpanel-Objekte verwenden. Dieser Code ist im Blockdiagramm enthalten. In mancher Hinsicht ähnelt dieses Blockdiagramm einem Flussdiagramm.

Für die Entwicklung spezieller Applikationen ist LabVIEW mit einer Reihe von Software-Werkzeugsätzen (Add-Ons) erweiterbar. All diese Werkzeugsätze lassen sich nahtlos in LabVIEW integrieren. Weitere Hinweise dazu finden Sie auf der Internetseite von National Instruments, `ni.com`.

LabVIEW-Dokumentation

Zum Lieferumfang von LabVIEW gehört eine umfangreiche Dokumentation für Einsteiger und fortgeschrittene Anwender. Alle LabVIEW-Handbücher und Applikationsinformationen stehen auch als PDF-Dateien zur Verfügung. Zur Anzeige der PDF-Dateien muss Adobe Acrobat Reader 5.0.5 (oder eine aktuellere Version) mit Suchfunktion installiert sein. Dieser kann auch von der Internetseite von Adobe Systems unter `www.adobe.de` heruntergeladen werden. Die jeweils aktuelle Version der Handbücher finden Sie in der Produktbibliothek von National Instruments unter `ni.com/manuals`.

- **LabVIEW-Bibliothek**—In dieser Datei sind alle LabVIEW-Handbücher und -Versionshinweise im PDF-Format zu finden. Zum Öffnen der *LabVIEW-Bibliothek* wählen Sie **Hilfe»Suchen in der LabVIEW-Bibliothek**.

- **Erste Schritte mit LabVIEW**—Anhand dieses Handbuchs können Sie sich mit der grafischen Programmierumgebung von LabVIEW und den grundlegenden LabVIEW-Funktionen zum Erstellen von Applikationen zur Datenerfassung und Gerätesteuerung vertraut machen.
- **LabVIEW-Referenzkarte**—Dient als Schnellübersicht zu Informationsquellen zu LabVIEW, Tastaturkombinationen, Anschlussdatentypen und Werkzeugen zur Bearbeitung und Ausführung von VIs sowie zur Fehlersuche.
- **LabVIEW Benutzerhandbuch**—Hier lernen Sie die Begriffe und Techniken der LabVIEW-Programmierung kennen sowie Komponenten, VIs und Funktionen, mit deren Hilfe Test- und Messapplikationen sowie Applikationen zur Datenerfassung, Gerätesteuerung, Datenprotokollierung, Analyse von Messdaten und Berichterzeugung erstellt werden können.
- **LabVIEW-Hilfe**—Die LabVIEW-Hilfe ist eine umfangreiche Informationsquelle zu den LabVIEW-Paletten, -Menüs, -Werkzeugen, -VIs und -Funktionen. Hier finden Sie auch schrittweise Anleitungen zur Benutzung der wesentlichen Funktionen von LabVIEW. Zum Öffnen der *LabVIEW-Hilfe* wählen Sie **Hilfe»VI-, Funktionen- und Anwendungshilfe**.

  Die *LabVIEW-Hilfe* umfasst Links zu den folgenden Ressourcen:

  – *LabVIEW-Bibliothek*—In dieser Datei finden Sie PDF-Versionen aller LabVIEW-Handbücher und Versionshinweise

  – Ressourcen für technische Unterstützung auf der Internetseite von National Instruments, wie beispielsweise die NI Developer Zone, die KnowledgeBase oder die Bibliothek der Produkthandbücher

Hinweis **(Mac OS und UNIX)** Zur Anzeige der *LabVIEW Help* wird empfohlen, Netscape Navigator 6.0 oder Internet Explorer 5.0 oder jeweils eine aktuellere Version zu verwenden.

- **LabVIEW Measurements Manual**—In diesem Handbuch erfahren Sie mehr über das Erstellen von Anwendungen zur Datenerfassung und Instrumentensteuerung mit LabVIEW. Wenn Sie mit LabVIEW noch nicht vertraut sind, lesen Sie das Handbuch *Erste Schritte mit LabVIEW* und das *LabVIEW-Benutzerhandbuch*, bevor Sie sich mit diesem Handbuch befassen.
- **Benutzerhandbuch zum LabVIEW Application Builder**—Enthält Informationen über den LabVIEW Application Builder, der im LabVIEW Professional Development System enthalten ist und zu den anderen LabVIEW-Paketen zusätzlich erworben werden kann. Sie

finden eine Anleitung zur Installation des Application Builders, und es werden die Systemvoraussetzungen für Applikationen sowie die Unterschiede zur vorherigen Versionen beschrieben. Ebenfalls enthalten sind Warnungen und Hinweise, die bei der Umwandlung von VIs in ausführbare Anwendungen oder eine Shared Library zu beachten sind.

- **LabVIEW Development Guidelines**—In diesem Handbuch wird die Erstellung leicht verständlicher VIs erklärt, die einfach in der Handhabung und Bearbeitung sind. Daneben erfahren Sie mehr zu Methoden der Projektverfolgung, -gestaltung und -dokumentation. Dazu sind auch die empfohlenen Gestaltungsrichtlinien aufgeführt.

Hinweis Das Handbuch *LabVIEW Development Guidelines* steht nur im LabVIEW Professional Development System (PDS) zur Verfügung. Die PDF-Version des Handbuchs ist in allen LabVIEW-Paketen enthalten.

- **LabVIEW Analysis Concepts**—Verschafft Innen einen Überblick über die in LabVIEW verwendeten Analysekonzepte. Daneben finden Sie in diesem Handbuch Informationen zu den Themen Signalerzeugung, schnelle Fouriertransformation (FFT), diskrete Fouriertransformation (DFT), Glättungsfenster, lineare Algebra, Punkt-für-Punkt-Analyse für die Echtzeitanalyse sowie die Grundbegriffe der Wahrscheinlichkeit und Statistik.

Hinweis Das Dokument *LabVIEW Analysis Concepts* ist nur als PDF-Datei verfügbar.

- **Using External Code in LabVIEW**—In diesem Handbuch erfahren Sie, wie Sie mit Hilfe von Code Interface Knoten (CINs) und externen Subroutinen Programmcode aus befehlsorientierten Programmiersprachen in LabVIEW integrieren können. Sie erhalten Informationen zum Aufrufen von DDLs und gemeinsam genutzten externen Unterprogrammen, zu Funktionsbibliotheken, Routinen für das Bearbeiten von Speicherinhalten und Dateien und zu Diagnoseroutinen.

Hinweis Das Dokument *Using External Code in LabVIEW* steht nur als PDF-Datei zur Verfügung.

- **LabVIEW-Versionshinweise**—Hier finden Sie Informationen zur Installation und Deinstallation von LabVIEW. Außerdem sind hier die Systemvoraussetzungen zur Installation von LabVIEW und bekannte Probleme im Zusammenhang mit LabVIEW beschrieben.

- **LabVIEW Upgrade Notes**—Dieses Dokument enthält Hinweise zur Aktualisierung von LabVIEW unter Windows, Mac OS und UNIX. Neben neuen Funktionen finden Sie hier auch Hinweise zu Problemen, die bei der Aktualisierung auftreten können.
- **LabVIEW Application Notes**—In den LabVIEW Application Notes erhalten Sie einen Einblick in erweiterte oder spezielle LabVIEW -Konzepte und -Applikationen. In der NI Developer Zone unter `ni.com/zone` finden Sie stets die aktuellen Application Notes.
- **LabVIEW VXI VI Reference Manual**—Ist ein Begleitheft zum *NI-VXI Programmer Reference Manual* und enthält Beschreibungen der VXI-VIs für LabVIEW. Für die Konfiguration und Programmierung von Instrumenten, die mit VXI-Hardware zusammenarbeiten, sowie für die Fehlersuche in solchen Systemen wird von National Instruments der Einsatz von VISA-Technologie empfohlen.

Hinweis Das Handbuch *LabVIEW VXI VI Reference Manual* steht nur als PDF-Datei zur Verfügung.

VI-Vorlagen, Beispiel-VIs und Werkzeuge in LabVIEW

Mit den VI-Vorlagen, Beispiel-VIs und Werkzeugen, die Ihnen in LabVIEW zur Verfügung stehen, können Sie ganz einfach eigene VIs erstellen.

Vorlagen für LabVIEW-VIs

Zu den in LabVIEW verfügbaren VI-Vorlagen gehören SubVIs, Funktionen, Strukturen und Frontpanel-Objekte. Sie dienen zur Erstellung von VIs für allgemeine messtechnische Anwendungen und sollen Ihnen den Einstieg in die LabVIEW-Programmierung erleichtern. Vorlagen werden immer unbenannt geöffnet und müssen daher zunächst benannt und gespeichert werden. Zum Öffnen des Dialogfensters **Neu**, in dem die VI-Vorlagen ausgewählt werden können, ist entweder **Datei»Neu** zu wählen oder die Schaltfläche **Neu** im **LabVIEW**-Startfenster anzuklicken.

Beispiel-VIs in LabVIEW

In LabVIEW stehen Ihnen Hunderte von Beispiel-VIs zur Verfügung, die Sie auch in eigene VIs integrieren können. So können Sie ein Beispiel Ihrer Anwendung entsprechend anpassen oder auch Beispiele kopieren und in Ihr eigenes VI einfügen. Über **Hilfe»Beispiele finden** kann auch nach Beispiel-VIs gesucht werden. Weitere Beispiel-VIs sind in der NI Developer Zone unter `ni.com/zone` verfügbar.

LabVIEW-Werkzeuge

In LabVIEW gibt es viele Werkzeuge, mit denen sich Messgeräte zeitsparend konfigurieren lassen. Auf die folgenden Werkzeuge können Sie über das **Werkzeuge**-Menü zugreifen.

- **(Windows)** Zur Konfiguration von Hard- und Software von National Instruments steht Ihnen der Measurement & Automation Explorer (MAX) zur Verfügung.
- **(Mac OS 9 oder eine ältere Version)** Bei der NI-DAQ Configuration Utility von National Instruments ist Ihnen das NI-DAQ Configuration Utility behilflich.
- **(Mac OS 9 oder eine ältere Version)** Mit Hilfe des DAQ Channel Wizard können Sie den Typen des an einen Hardware-Kanal angeschlossenen Gerätes bestimmen. Nachdem Sie einen Kanal definiert haben, speichert der DAQ-Kanalassistent die Einstellungen.
- **(Mac OS 9 oder eine ältere Version)** Alle DAQ Channel Viewer sind im DAQ-Kanalmonitor aufgeführt.
- **(Mac OS 9 oder eine ältere Version)** Im DAQ Solution Wizard finden Sie Beispiele für allgemeine Anwendungsarten der Datenerfassung. Hier finden Sie Beispiel-VIs oder können benutzerdefinierte VIs erstellen lassen.

Mit dem DAQ Assistant ist es möglich, Kanäle grafisch oder auf eine bestimmte Messart zu konfigurieren. Zum Start des DAQ Assistant wählen Sie eine der folgenden Möglichkeiten:

- Platzieren Sie das Express-VI “DAQ Assistant” in das Blockdiagramm.
- Klicken Sie mit der rechten Maustaste auf ein Bedienelement des Typs “DAQmx: Globaler Kanal” und wählen Sie im Kontextmenü den Befehl **New Channel (DAQ Assistant)** aus. Klicken Sie mit der rechten Maustaste auf ein Bedienelement des Typs “DAQmx-Taskname” und wählen Sie **New Task (DAQ Assistant)** aus. Klicken Sie mit der rechten Maustaste auf ein Bedienelement des Typs “DAQmx-Skalierungsname” und wählen Sie **New Scale (DAQ Assistant)** aus.
- Starten Sie den Measurement & Automation Explorer und klicken Sie in der Verzeichnisstruktur auf **Datenumgebung** oder **Skalierungen**. Klicken Sie auf die Schaltfläche **Neu**. Nehmen Sie dann die gewünschten Einstellungen des NI-DAQmx-Kanals, Tasks oder der Skalierung vor.

Weitere Hinweise zur Verwendung des DAQ-Assistenten finden Sie im *LabVIEW Measurements Manual*.

2

Einführung in die Welt der virtuellen Instrumente

Die LabVIEW-Programme werden als virtuelle Instrumente oder VIs bezeichnet, da mit Erscheinungsbild und Funktion physische Instrumente wie beispielsweise Oszilloskope und Multimeter nachgebildet werden. Jedes VI arbeitet mit Funktionen, die Eingaben von der Benutzeroberfläche oder aus anderen Quellen verarbeiten. Diese Informationen können dann angezeigt, oder in andere Dateien oder auf andere Computer verschoben werden.

Ein VI enthält die folgenden drei Komponenten:

- **Frontpanel**—Dient als Benutzeroberfläche.
- **Blockdiagramm**—Enthält den grafischen Quellcode, mit dem die Funktion des VIs definiert wird.
- **Symbol und Anschlussfeld**—Identifiziert das VI, so dass Sie das VI in einem anderen VI verwenden können. Ein VI, das einem anderen VI untergeordnet ist, wird als SubVI bezeichnet. Ein SubVI entspricht einem Unterprogramm in textbasierten Programmiersprachen.

Weitere Informationen …

Weitere Informationen zum Erstellen von VIs und SubVIs finden Sie in der *LabVIEW-Hilfe*.

Frontpanel

Das Frontpanel ist die Benutzeroberfläche des VIs. In Abbildung 2-1 finden Sie ein Beispiel für ein Frontpanel.

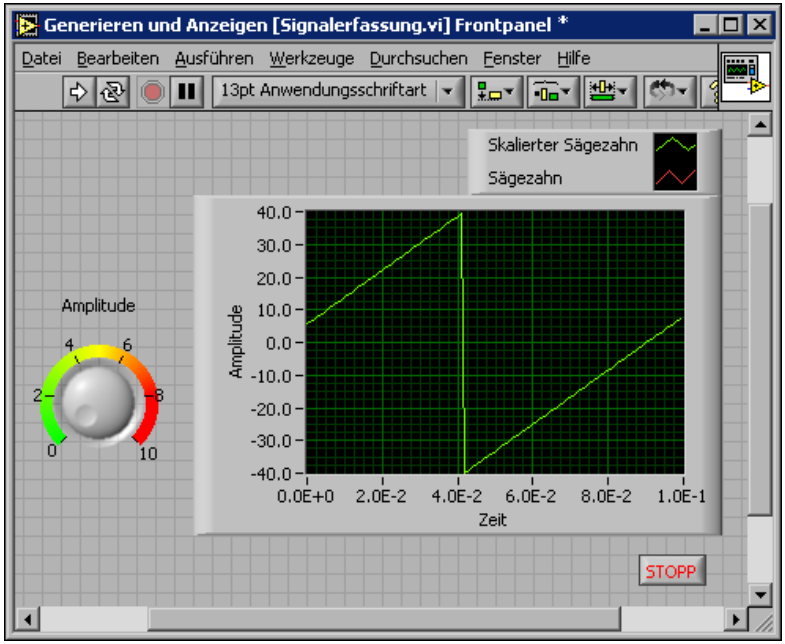


Abbildung 2-1. Beispiel für ein Frontpanel

Das Frontpanel wird mit Bedien- und Anzeigeelementen erstellt, welche die interaktiven Ein- beziehungsweise Ausgabeanschlüsse des VIs darstellen. Bedienelemente sind Knöpfe, Drucktasten, Drehregler und andere Eingabeelemente. Anzeigeelemente sind Graphen, LEDs und sonstige Anzeigen. Mit den Bedienelementen werden Eingabegeräte simuliert und Daten an das Blockdiagramm des VIs übergeben. Mit Anzeigeelementen werden Ausgabegeräte nachgeahmt und die Daten angezeigt, die vom Blockdiagramm erfasst oder erzeugt werden. Weitere Informationen zum Frontpanel finden Sie in Kapitel 4, *Erstellen des Frontpanels*.

Blockdiagramm

Nachdem Sie das Frontpanel erstellt haben, können Sie mit Hilfe grafisch dargestellter Funktionen Code hinzufügen, um die Frontpanel-Objekte zu steuern. Das Blockdiagramm enthält dann diesen grafischen Quell-Code.

Frontpanel-Objekte werden im Blockdiagramm als Anschluss-Terminals dargestellt. Weitere Informationen über das Blockdiagramm finden Sie in Kapitel 5, *Erstellen des Blockdiagramms*.

Im VI in Abbildung 2-2 sehen Sie verschiedene Blockdiagrammobjekte, die in LabVIEW von Bedeutung sind – Anschlüsse, Funktionen und Verbindungen.

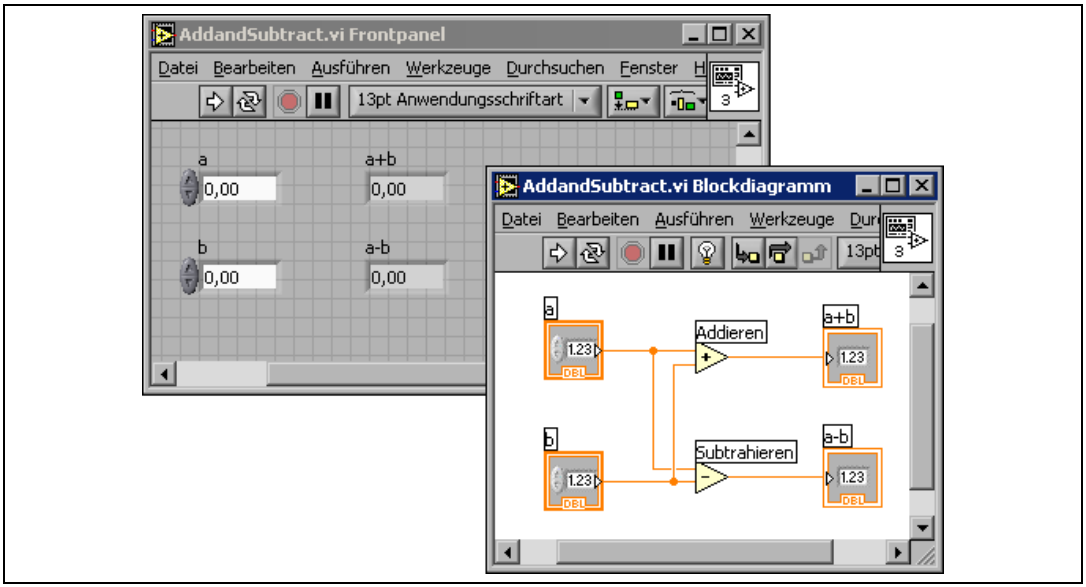


Abbildung 2-2. Beispiel für ein Blockdiagramm und das zugehörige Frontpanel

Anschlüsse

Die Symbole der Anschlüsse verweisen auf den Datentyp des Bedien- oder Anzeigeelements. Die Frontpanel-Elemente können so konfiguriert werden, dass sie im Blockdiagramm entweder als Symbol oder als Anschluss eines bestimmten Datentyps dargestellt werden. Per Voreinstellung werden Frontpanel-Objekte als Symbole dargestellt. So wird zum Beispiel, wie links angezeigt, mit einem Drehknopfsymbol ein auf dem Frontpanel befindlicher Drehknopf dargestellt. Das “DBL” am unteren Rand zeigt den Datentyp an. Das heißt, dieses Element arbeitet mit Fließkommazahlen mit doppelter Genauigkeit. Dagegen ist bei Darstellung als DBL-Symbol, wie links abgebildet, nur ersichtlich, dass es sich um ein numerisches Bedien- oder Anzeigeelement handelt, das diesen Datentyp verwendet. Weitere Informationen zu den Datentypen in LabVIEW und deren grafischer Darstellung finden Sie in dem Abschnitt *Datentypen für Bedien- und Anzeigeelemente* des Kapitels 5, *Erstellen des Blockdiagramms*.

Anschlüsse sind Eingangs- und Ausgangsports, über die Informationen zwischen dem Frontpanel und dem Blockdiagramm ausgetauscht werden. Daten, die Sie über die Frontpanel-Bedienelemente (**a** und **b** in Abbildung 2-2) eingeben, werden über die Bedienelement-Terminals an das Blockdiagramm übergeben. Anschließend passieren die Daten die Additions- und Subtraktionsfunktionen. Wenn die Additions- und Subtraktionsfunktionen die internen Berechnungen abgeschlossen haben, werden neue Datenwerte erstellt. Die Daten fließen zu den Anzeigeelementanschlüssen, wo sie das Blockdiagramm verlassen, um erneut an das Frontpanel übergeben und dann mit Hilfe der Frontpanel-Anzeigeelemente angezeigt zu werden (**a+b** und **a-b** in Abbildung 2-2).

Knoten

Knoten sind Objekte im Blockdiagramm, die über Eingänge und/oder Ausgänge verfügen und Funktionen ausführen, wenn ein VI ausgeführt wird. Knoten entsprechen Anweisungen, Operatoren, Funktionen und Subroutinen in textbasierten Programmiersprachen. Die Additions- und Subtraktionsfunktionen in Abbildung 2-2 sind Knoten. Weitere Informationen zu Knoten finden Sie in dem Abschnitt *Blockdiagrammknoten* des Kapitels 5, *Erstellen des Blockdiagramms*.

Verbindungen

Sie übertragen die Daten über Verbindungsleitungen zwischen den Blockdiagrammobjekten.In der Abbildung 2-2 werden die als Bedien- und Anzeigeelemente fungierenden DBL-Anschlüsse über Verbindungsleitungen mit den Additions- und Subtraktionsfunktionen verbunden. Jede Verbindung verfügt über eine einzige Datenquelle, die sie jedoch mit mehreren Daten lesenden VIs (Datensenken) und Funktionen verbinden können. Verbindungen weisen in Abhängigkeit ihres Datentyps unterschiedliche Farben, Stile und Stärken auf. Eine unterbrochene Verbindung wird als eine gestrichelte schwarze Linie mit einem roten `X` in der Mitte dargestellt. Weitere Informationen zu Verbindungsleitungen finden Sie in dem Abschnitt *Verbindung von Blockdiagrammobjekten* des Kapitels 5, *Erstellen des Blockdiagramms*.

Strukturen

Strukturen sind grafische Darstellungen der Schleifen und Case-Anweisungen in textbasierten Programmiersprachen. Verwenden Sie Strukturen im Blockdiagramm, um Codeblöcke zu wiederholen und Code bedingungsabhängig oder in einer bestimmten Reihenfolge auszuführen. Beispiele und weitere Informationen zu Strukturen finden Sie im Kapitel 8, *Schleifen und Strukturen*.

Symbol- und Anschlussfeld

Nachdem Sie ein VI-Frontpanel und Blockdiagramm erstellt haben, erstellen Sie das Symbol- und Anschlussfeld, damit das VI auch als SubVI verwendet werden kann. Jedes VI hat ein Symbol (siehe Abbildung links), das in der rechten oberen Ecke des Frontpanels bzw. Blockdiagramms angezeigt wird. Das Symbol ist eine grafische Darstellung eines VIs. Es kann eine Kombination aus Text- und Grafikelementen enthalten. Wenn Sie ein VI als SubVI verwenden, kennzeichnet das Symbol das SubVI im Blockdiagramm des VIs. Um das Symbol zu editieren und zu verändern, führen Sie einen Doppelklick darauf aus. Weitere Informationen zu Symbolen finden Sie in dem Abschnitt *VI-Symbole erstellen* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Um ein VI als SubVI verwenden zu können, müssen Sie außerdem ein Anschlussfeld (siehe links) erstellen. Das Anschlussfeld ist eine Sammlung von Anschlüssen, die – ähnlich der Parameterliste eines Funktionsaufrufs in textbasierten Programmiersprachen – den Bedien- und Anzeigeelementen dieses VIs entsprechen. Mit dem Anschlussfeld werden die Eingänge und Ausgänge definiert, die mit dem VI verbunden werden sollen, damit Sie es als SubVI einsetzen können. Ein Anschlussfeld empfängt an seinen Eingangsanschlüssen Daten, übergibt diese über die Frontpanel-Bedienelemente an den Code des Blockdiagramms; die Frontpanel-Anzeigeelemente empfangen die Daten vom Blockdiagramm und geben sie an den Ausgabeanschlüssen aus.

Wenn Sie das Anschlussfeld zum ersten Mal anzeigen, sehen Sie ein Anschlussmuster, das Sie bei Bedarf ändern können. Das Anschlussfeld verfügt im Allgemeinen über einen Anschluss für jedes auf dem Frontpanel befindliche Bedien- oder Anzeigeelement. Sie können einem Anschlussfeld bis zu 28 Anschlüsse zuweisen. Wenn Sie mit Änderungen an einem VI rechnen, die neue Eingänge oder Ausgänge fordern, belassen Sie einige Anschlüsse ohne Zuweisung. Weitere Informationen zum Einrichten von Anschlussfeldern finden Sie in dem Abschnitt *Anschlussfeld einrichten* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Hinweis Sie sollten einem VI jedoch nicht mehr als 16 Anschlüsse zuweisen. Die Übersichtlichkeit und Brauchbarkeit des VIs wird durch zu viele Anschlüsse beeinträchtigt.

Verwenden und Anpassen von VIs und SubVIs

Nachdem Sie ein VI erstellt und dessen Symbol und Anschlussfeld definiert haben, können Sie es als SubVI verwenden. Weitere Informationen zu SubVIs finden Sie in Abschnitt *SubVIs* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Sie können VIs als einzelne Dateien speichern oder in einer VI-Bibliothek zusammenfassen. Weitere Informationen zum Speichern von VIs in Bibliotheken finden Sie in dem Abschnitt *Speichern von VIs* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Sie können auch das Erscheinungsbild und das Verhalten eines VIs Ihren Bedürfnissen anpassen. Darüber hinaus können Sie benutzerdefinierte Menüs für jedes von Ihnen erstellte VI anlegen und VIs so konfigurieren, dass die Menüleisten ein- oder ausgeblendet werden. Weitere Informationen zum Anpassen eines VIs finden Sie in dem Kapitel 16, *Anpassen von VIs*.

3

LabVIEW-Umgebung

Zum Erstellen der Frontpanels und Blockdiagramme von VIs stehen Ihnen die Paletten, Werkzeuge und Menüs von LabVIEW zur Verfügung. Die Paletten **Elemente** und **Funktionen** können benutzerspezifisch angepasst werden. Auch für die Arbeitsumgebung sind verschiedene Optionen einstellbar.

Weitere Informationen …

Weitere Informationen zum Einsatz von Paletten, Menüs und Symbolleisten sowie zum individuellen Anpassen der Arbeitsumgebung finden Sie in der *LabVIEW-Hilfe*.

Elementepalette

Die Palette **Elemente** steht nur auf dem Frontpanel zur Verfügung. Sie enthält die Bedien- und Anzeigeelemente zur Erstellung der Benutzeroberfläche eines VIs. Je nach Typ sind die Elemente in verschiedene Unterpaletten unterteilt. Weitere Informationen zu den Bedien- und Anzeigeelementen finden Sie im Abschnitt *Bedien- und Anzeigeelemente des Frontpanels* des Kapitels 4, *Erstellen des Frontpanels*. Je nach ausgewählter Palettenansicht können in der Palette **Elemente** unterschiedliche Bedien- und Anzeigeelemente zu sehen sein. Weitere Hinweise zu den möglichen Palettenansichten entnehmen Sie bitte dem Abschnitt *Erstellen und Bearbeiten von Palettenansichten* dieses Kapitels.

Klicken Sie zur Anzeige der Palette **Elemente** auf **Fenster»Elementepalette** oder klicken Sie mit der rechten Maustaste auf den Arbeitsbereich des Frontpanels. Die Palette **Elemente** kann beliebig auf dem Bildschirm verschoben werden. Die aktuelle Position der Palette wird beim Schließen von LabVIEW gespeichert, so dass diese beim nächsten Start wieder an derselben Stelle angezeigt wird.

Zur Darstellung der Palette **Elemente** sind verschiedene Optionen verfügbar. Lesen Sie dazu den Abschnitt *Anpassen der Elemente- und Funktionenpalette* dieses Kapitels.

Funktionenpalette

Die Palette **Funktionen** ist nur im Blockdiagramm verfügbar. Sie enthält die VIs und Funktionen zur Erstellung des Blockdiagramms. Diese sind ebenfalls je nach Typ in verschiedene Unterpaletten aufgeteilt. Weiterführende Informationen zu den VIs und Funktionen finden Sie im Abschnitt *Überblick über Funktionen* des Kapitels 5, *Erstellen des Blockdiagramms*. Für die Palette **Funktionen** können verschiedene Ansichten ausgewählt werden, so dass unterschiedliche Bedien- und Anzeigeelemente angezeigt werden können. Weitere Hinweise zu den möglichen Palettenansichten entnehmen Sie bitte dem Abschnitt *Erstellen und Bearbeiten von Palettenansichten* dieses Kapitels.

Zur Anzeige der Palette **Funktionen** wählen Sie **Fenster»Funktionenpalette** oder klicken Sie mit der rechten Maustaste in den Arbeitsbereich des Blockdiagramms. Sie kann beliebig auf dem Bildschirm verschoben werden. Die aktuelle Position der Palette wird beim Schließen von LabVIEW gespeichert, so dass diese beim nächsten Start wieder an derselben Stelle angezeigt wird.

Die Palette **Funktionen** kann in verschiedenen Ansichten dargestellt werden. Lesen Sie dazu den Abschnitt *Anpassen der Elemente- und Funktionenpalette* dieses Kapitels.

Auswahl von Elementen und Funktionen in den Paletten

Wenn Sie auf das Symbol einer Unterpalette klicken, wird die ausgewählte Unterpalette angezeigt. Um ein Palettenobjekt auf das Blockdiagramm zu platzieren, klicken Sie es an. Das Objekt wird dann zunächst dem Cursor unterlegt und kann an jeder beliebigen Stelle im Blockdiagramm abgelegt werden. Um ein VI zu öffnen, klicken Sie mit der rechten Maustaste auf das entsprechende Symbol und wählen Sie aus dem Kontextmenü die Option **Frontpanel öffnen**.

Um nach Bedienelementen, VIs und Funktionen zu suchen, verwenden Sie die Navigationsschaltflächen in den Paletten **Elemente** und **Funktionen**.

- **Nach oben**—Hiermit gelangen Sie in der Palettenhierarchie eine Ebene nach oben. Wenn Sie beim Anklicken dieser Schaltfläche die Maustaste gedrückt halten, wird ein Kontextmenü mit dem Pfad von der Hauptpalette bis hin zur aktuell geöffneten Unterpalette angezeigt. Über dieses Menü gelangen Sie auch zu einer bestimmten Unterpalette.

- **Suchen**—Bei Betätigung dieser Schaltfläche wechselt die Palette in den Suchmodus, so dass in den Paletten nach Bedienelementen, VIs oder Funktionen gesucht werden kann. Um den Suchmodus zu verlassen, klicken Sie auf die Schaltfläche **Zurück**.

- **Optionen**—Öffnet das Dialogfeld **Optionen: Funktionen durchsuchen**, in dem Ansicht und Format einer Palette festgelegt werden können.

- **Palettengröße wiederherstellen**—Stellt eine Palette auf die Standardgröße ein. Diese Schaltfläche wird nur angezeigt, wenn die Palette **Elemente** oder **Funktionen** in ihrer Größe verändert wurde.

Werkzeugpalette

Die Palette **Werkzeuge** ist sowohl auf dem Frontpanel als auch auf dem Blockdiagramm verfügbar. Ein Werkzeug ist eine spezielle Betriebsart des Mauszeigers. Bei Auswahl eines Werkzeugs nimmt der Cursor die in der Palette dargestellte Form an. Damit können Frontpanel- und Blockdiagrammobjekte bedient oder verändert werden.

Zur Anzeige der Palette **Werkzeuge** wählen Sie **Fenster»Werkzeugpalette**. Die Palette **Werkzeuge** kann beliebig auf dem Bildschirm verschoben werden. Die Position wird beim Schließen von LabVIEW gespeichert, so dass die Palette beim nächsten Start des Programms wieder an derselben Stelle angezeigt wird.

Tipp Zur vorübergehenden Anzeige der Palette **Werkzeuge** betätigen Sie bei gedrückter <Umschalttaste> die rechte Maustaste.

Wenn die automatische Werkzeugauswahl eingeschaltet ist und Sie den Cursor über ein Frontpanel- oder Blockdiagramm-Objekt bewegen, wird aus der Palette **Werkzeuge** automatisch das für den gewünschten Arbeitsvorgang passende Werkzeug ausgewählt. Die automatische Werkzeugauswahl kann jedoch auch deaktiviert werden. Klicken Sie dazu in der Palette **Werkzeuge** auf die links abgebildete Schaltfläche für die **automatische Werkzeugauswahl**. Um sie wieder zu aktivieren, klicken Sie entweder erneut auf die Schaltfläche oder drücken Sie die Tastenkombination <Umschalt-Tabulator>. Sobald Sie ein Werkzeug manuell auswählen, indem Sie es direkt anklicken, wird die automatische Werkzeugauswahl deaktiviert. Um sie dann wieder zu aktivieren, klicken Sie entweder erneut auf die Schaltfläche oder drücken Sie die Taste <Tabulator>.

Menüs und Symbolleiste

Mit Hilfe der Elemente aus den Menüs und der Symbolleiste können Frontpanel- und Blockdiagrammobjekte bedient und verändert werden. Die Schaltflächen der Symbolleiste dienen zum Ausführen von VIs.

Menüs

In den Menüs am oberen Rand eines VI-Fensters sind LabVIEW-spezifische und allgemeine Optionen wie **Öffnen**, **Speichern**, **Kopieren** oder **Einfügen** enthalten. Zu einigen Menüelementen sind auch Tastenkombinationen angegeben.

(Mac OS) Die Menüs werden am oberen Bildrand angezeigt.

(Windows und UNIX) Per Voreinstellung werden immer nur die zuletzt verwendeten Menüpunkte angezeigt. Um alle Menüpunkte sichtbar zu machen, klicken Sie auf die Pfeile am unteren Rand. Wenn die Menüs standardmäßig vollständig angezeigt werden sollen, wählen Sie dazu **Werkzeuge»Optionen** und nehmen Sie unter **Verschiedenes** die entsprechende Einstellung vor.

Hinweis Einige Menüpunkte stehen im Ausführungsmodus eines VIs nicht zur Verfügung.

Kontextmenüs

Das in LabVIEW am häufigsten anzutreffende Menü ist das “Objekt-Kontextmenü”. Ein solches Menü ist zu allen LabVIEW-Objekten und den Leerflächen auf dem Frontpanel verfügbar. Mit Hilfe dieser Menüs lassen sich Einstellungen zur Darstellung und zum Verhalten von Frontpanel- oder Blockdiagrammobjekten vornehmen. Zur Anzeige des Kontextmenüs klicken Sie mit der rechten Maustaste auf das entsprechende Objekt, Frontpanel oder Blockdiagramm.

Kontextmenüs im Ausführungsmodus

Wenn ein VI ausgeführt wird oder sich im Ausführungsmodus befindet, steht zu allen Frontpanel-Objekten nur ein gekürztes Kontextmenü zur Verfügung. Damit lässt sich der Inhalt des Objekts ausschneiden, kopieren oder einfügen, das Objekt auf den Standardwert zurücksetzen oder eine entsprechende Beschreibung anzeigen.

Zu einigen der komplexeren Bedienelemente sind zusätzliche Optionen vorhanden. So enthält das Array-Kontextmenü beispielsweise Befehle, um einen Wertebereich zu kopieren oder um zum letzten Element des Arrays zu wechseln.

Symbolleiste

Verwenden Sie die Schaltflächen der Symbolleiste, um ein VI auszuführen oder zu bearbeiten. Beim Ausführen eines VIs werden in der Symbolleiste Schaltflächen zur Fehlersuche angezeigt.

Kontexthilfe-Fenster

Wenn der Cursor über ein LabVIEW-Objekt bewegt wird, werden im Fenster der **Kontexthilfe** die wichtigsten Informationen dazu angezeigt. Diese Hilfe ist zu VIs, Funktionen, Konstanten, Strukturen, Paletten, Eigenschaften, Methoden, Ereignissen und Komponenten von Dialogfeldern verfügbar. Die **Kontexthilfe** hilft Ihnen auch beim Verdrahten von VIs oder Funktionen dabei, eng beieinander liegende Anschlüsse voneinander zu unterscheiden. Weitere Informationen zur Verwendung der **Kontexthilfe** finden Sie im Abschnitt *Manuelles Verbinden von Objekten* des Kapitels 5, *Erstellen des Blockdiagramms*.

Wählen Sie **Hilfe»Kontexthilfe anzeigen**, um das Fenster **Kontexthilfe** anzuzeigen. Das Fenster kann jedoch auch – wie links dargestellt – durch Anklicken der Schaltfläche **Kontexthilfe** in der Werkzeugleiste oder unter Windows durch Drücken der Tastenkombination <Strg-H> geöffnet werden. **(Mac OS)** Drücken Sie <Befehlstaste-H>. **(UNIX)** Drücken Sie <Alt-H>.

Die **Kontexthilfe** kann beliebig auf dem Bildschirm verschoben werden. Die Größe des Fensters passt sich jeweils automatisch dem Inhalt an, kann aber auch auf die maximale Größe eingestellt werden. Diese Einstellung wird dann zusammen mit der aktuellen Position der **Kontexthilfe** beim Schließen von LabVIEW gespeichert, so dass das Fenster beim nächsten Start von LabVIEW wieder an derselben Stelle und in derselben Größe angezeigt wird.

Des weiteren ist es möglich, die **Kontexthilfe** so einzustellen, dass sich der angezeigte Inhalt nicht ändert, wenn der Cursor über ein anderes Objekt bewegt wird. Wählen Sie dazu **Hilfe»Kontexthilfe fixieren**. Um diesen Zustand wieder aufzuheben, entfernen Sie das entsprechende Häkchen. Die **Kontexthilfe** arbeitet dann wieder wie gehabt. Dieselbe Einstellung kann aber auch über die Schaltfläche mit dem Vorhängeschloss im **Kontexthilfe**-Fenster selbst oder unter Windows über die Tastenkombination

<Strg-Umschalt-L> vorgenommen werden. **(Mac OS)** Drücken Sie <Befehl-Shift-L>. **(UNIX)** Drücken Sie <Alt-Shift-L>.

Zur Anzeige der optional zu verbindenden Anschlüsse klicken Sie in der **Kontexthilfe** auf die links dargestellt Schaltfläche **Ausführliche Kontexthilfe**. Weitere Informationen zu diesen Anschlüssen finden Sie in Abschnitt *Festlegen von erforderlichen, empfohlenen und optionalen Ein- und Ausgängen* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Wenn es zum Objekt in der **Kontexthilfe** ein *LabVIEW-Hilfethema* gibt, wird in blauer Schrift der Link `Klicken Sie hier, um mehr Hilfe zu erhalten` angezeigt. Außerdem ist dann die Schaltfläche **Weitere Hilfe** im Fenster der **Kontexthilfe** aktiviert (siehe Abbildung links). Um das entsprechende Thema in der *LabVIEW-Hilfe* zu öffnen, klicken Sie entweder auf den Link oder die Schaltfläche.

Hinweise zum Erstellen von Beschreibungen zur Anzeige in der **Kontexthilfe** finden Sie im Abschnitt *Erstellen von VI- und Objektbeschreibungen* des Kapitels 15, *Dokumentieren und Drucken von VIs*.

Anpassen der Arbeitsumgebung

Die Paletten “Elemente” und “Funktionen” können benutzerspezifisch angepasst werden. Eine Auswahl der Palettenansicht ist beispielsweise über das Dialogfeld “Optionen” möglich. Hier können auch weitere Einstellungen der Arbeitsumgebung vorgenommen werden.

Anpassen der Elemente- und Funktionenpalette

Um die Paletten **Elemente** und **Funktionen** benutzerspezifisch anzupassen, sind folgende Vorgehensweisen möglich:

- Sie können VIs und Bedienelemente zu den Paletten hinzufügen.
- Sie können für verschiedene Benutzer unterschiedliche Ansichten einrichten. So können zum Beispiel einige VIs und Funktionen ausgeblendet werden, um bestimmten Anwendern den Umgang mit LabVIEW zu vereinfachen.
- Sie können die Paletten neu ordnen, um häufig verwendete VIs und Funktionen leichter zugänglich zu machen.
- Sie können ActiveX-Bedienelemente in benutzerdefinierte Bedienelemente umwandeln und diese zu den Paletten hinzufügen.
- Sie können den Paletten Toolsets hinzufügen.

Vorsicht! Speichern Sie eigene VIs und Bedienelemente, die den Paletten **Funktionen** und **Elemente** hinzugefügt werden sollen, *nie* in das Verzeichnis `vi.lib`, da die Dateien sonst bei der Installation neuer Software-Versionen überschrieben werden, sondern immer in das Verzeichnis `user.lib`.

Hinzufügen von VIs und Bedienelementen zur Benutzerbibliothek und zur Instrumentenbibliothek

Am einfachsten lassen sich VIs und Bedienelemente zu den Paletten **Funktionen** und **Elemente** hinzufügen, indem sie in das Verzeichnis `labview/user.lib` gespeichert werden. Beim anschließenden Neustart von LabVIEW enthalten die Paletten **Eigene Bibliotheken** und **Eigene Elemente** Unterpaletten für jedes Verzeichnis, jede VI-Bibliotheksdatei (`.llb`) oder Menüdatei (`.mnu`) sowie ein Symbol für jede Datei in `labview\user.lib`. Die Paletten werden vor jedem Neustart von LabVIEW automatisch aktualisiert, wenn Sie bestimmten Verzeichnissen Dateien hinzufügen oder Dateien entfernen.

Die Dateien zur Palette **Instrumenten-I/O** befinden sich im Verzeichnis `labview\instr.lib`. Sie sollten daher alle Instrumententreiber in dieses Verzeichnis speichern, um über die Palette **Funktionen** darauf zugreifen zu können.

Wenn Sie der Palette **Funktionen** bzw. **Elemente** auf diese Weise VIs oder Bedienelemente hinzufügen, haben Sie jedoch keinen Einfluss darauf, wie jede Unterpalette bezeichnet wird und an welcher Stelle der Palette das VI bzw. Bedienelement eingefügt wird.

Erstellen und Bearbeiten von Palettenansichten

Um den Namen jeder Unterpalette bestimmen zu können und festzulegen, wo genau die erstellten Elemente und VIs in die Palette **Funktionen** bzw. **Elemente** eingefügt werden sollen, müssen Sie eine benutzerdefinierte Palettenansicht erstellen. Die Palettenansichten “Express” und “Fortgeschritten” sind in LabVIEW bereits voreingestellt. Um benutzerdefinierte Palettenansichten zu erstellen oder zu bearbeiten, wählen Sie **Werkzeuge»Fortgeschritten»Palettenansichten bearbeiten**.

Hinweis Voreingestellte Palettenansichten sind unveränderlich.

Alle Informationen zu den Paletten **Elemente** und **Funktionen** werden in das Verzeichnis `labview\menus` gespeichert. Unter `menus` befinden sich alle zugehörigen Verzeichnisse für neu erstellte oder installierte Ansichten. Wenn LabVIEW in einem Netzwerk installiert ist, können für jeden Benut-

zer individuelle menu-Verzeichnisse festgelegt werden, wodurch der Austausch von Palettenansichten zwischen den einzelnen Anwendern vereinfacht wird.

Beim Erstellen einer neuen Palettenansicht verwendet LabVIEW zunächst eine Kopie der originalen Palette, die sich unter `labview\menus` befindet, die Sie dann Ihren Bedürfnissen entsprechend verändern können. So können Sie mit den Paletten experimentieren und müssen nicht befürchten, dadurch möglicherweise die ursprüngliche Ansicht zu verändern.

Speichern von Palettenansichten in LabVIEW

Die `.mnu`- und `.llb`-Dateien enthalten jeweils eine **Elemente-** und eine **Funktionenpalette** und ein entsprechendes Symbol. Jede neu erstellte Unterpalette muss in einer eigenen `.mnu`-Datei gespeichert werden.

Wenn Sie eine Ansicht auswählen, überprüft LabVIEW das Verzeichnis `menus` auf Vorhandensein eines Verzeichnisses, das dieser Ansicht entspricht. In dem zusammen mit der benutzerdefinierten Palette automatisch erzeugten Verzeichnis werden nun die Hauptpaletten **Elemente** und **Funktionen** und die Unterpaletten aus der Datei `root.mnu` generiert.

LabVIEW erstellt für jedes VI oder Bedienelement ein Symbol in der Palette. Darüber hinaus wird für jede `.mnu`- und `.llb`-Datei auf der entsprechenden Palette eine Unterpalette erzeugt.

Erstellen von ActiveX-Unterpaletten

Wenn Sie im Frontpanel ActiveX-Elemente verwenden, wählen Sie **Werkzeuge»Fortgeschritten»Importiere ActiveX-Elemente**, um diese in benutzerdefinierte Elemente umzuwandeln und der Palette **Elemente** hinzuzufügen. LabVIEW speichert die Elemente per Voreinstellung in das Verzeichnis `user.lib`, da alle Dateien und Verzeichnisse der `user.lib` automatisch in den Paletten angezeigt werden.

Darstellen von Toolsets und Modulen in den Paletten

Toolsets und Module, zu denen VIs oder Bedienelemente in `vi.lib\addons` gehören, werden nach einem Neustart von LabVIEW in den Paletten **Elemente** und **Funktionen** angezeigt. In der voreingestellten Express-Palette werden für Toolsets und Module in den Paletten **Alle Bedienelemente** und **Alle Funktionen** Unterpaletten angelegt. In der voreingestellten Palette **Fortgeschritten** werden die Unterpaletten auf den Hauptpaletten **Elemente** und **Funktionen** erstellt.

Wenn Bedienelemente für Toolsets oder Module den Paletten hinzugefügt werden sollen, die nicht unter `vi.lib\addons` gespeichert sind, müssen die Bedienelemente einfach nur in dieses Verzeichnis verschoben werden.

Festlegen von Optionen für die Arbeitsumgebung

Über **Werkzeuge»Optionen** kann LabVIEW benutzerspezifisch angepasst werden. So können im Dialogfeld **Optionen** Einstellungen zum Frontpanel, Blockdiagramm, zu Pfaden, Leistung und Speicher, dem Ausrichtungsraster, den Paletten, Rückgängig-Schritten, Fehlersuchwerkzeugen, Farben, Schriftarten, zum Druckvorgang, zum **Historie-**Fenster und zu anderen LabVIEW-Komponenten vorgenommen werden.

Über das Pulldown-Menü im Fenster **Optionen** kann zwischen den verschiedenen Kategorien der Optionen ausgewählt werden.

Speichern von Optionen

Sie müssen die Optionen nicht manuell bearbeiten und auch deren exaktes Format nicht wissen, denn diese Aufgabe erledigt das Dialogfeld **Optionen** für Sie. LabVIEW speichert die Optionen je nach Plattform unterschiedlich.

Windows

Die Optionen werden in der Datei `labview.ini` im LabVIEW-Verzeichnis gespeichert. Das Dateiformat gleicht dem anderer `.ini`-Dateien. Es beginnt mit einem LabVIEW-Abschnittsmarker, gefolgt vom Optionsnamen und den Werten, wie beispielsweise `offscreenUpdates=True`.

Wenn Sie eine andere Optionen-Datei verwenden möchten, geben Sie diese in der Verknüpfung an, die Sie zum Starten von LabVIEW verwenden. Um für die Optionen beispielsweise anstelle von `labview.ini` eine auf dem Computer gespeicherte Datei mit dem Namen `lvrc` zu verwenden, klicken Sie auf dem Desktop mit der rechten Maustaste auf das LabVIEW-Symbol und wählen Sie aus dem Kontextmenü die Option **Eigenschaften**. Klicken Sie auf die Registerkarte **Verknüpfung** und geben Sie `labview -pref lvrc` in das Textfeld **Ziel** ein.

Mac OS

LabVIEW speichert die Optionen in die Textdatei `LabVIEW Preferences` unter **System»Preferences**.

Wenn Sie eine andere Optionsdatei verwenden möchten, kopieren Sie die Datei `LabVIEW Preferences` in den Ordner **LabVIEW** und nehmen Sie

dann im Dialogfeld **Options** die gewünschten Änderungen vor. Beim Start sucht LabVIEW zunächst im **LabVIEW**-Ordner nach einer Optionsdatei. Wird keine Datei gefunden, wird die Suche im Odner **System** fortgesetzt. Wenn hier ebenfalls keine Datei gefunden wird, erstell LabVIEW im Ordner **System** eine neue Datei. Änderungen im Dialogfenster **Options** werden von LabVIEW immer in der zuerst gefundenen `LabVIEW Preferences`-Datei abgespeichert.

UNIX

LabVIEW speichert die Optionen in der Datei `.labviewrc` im Stammverzeichnis. Änderungen im Dialogfeld **Options** werden in die Datei `.labviewrc` geschrieben. Es ist aber auch möglich, im Programmverzeichnis eine `labviewrc`-Datei für die Optionen zu erstellen, die für alle Benutzer gleich sind, wie zum Beispiel der VI-Suchpfad. In die Datei `.labviewrc` sollten Optionen gespeichert werden, die je nach Benutzer unterschiedlich sind, wie zum Beispiel Einstellungen der Schriftart oder Farbeinstellungen, da Einträge der Datei `.labviewrc` im Stammverzeichnis widersprüchliche Einträge im Programmverzeichnis überschreiben.

Wenn Sie die LabVIEW-Dateien zum Beispiel unter `/opt/labview` gespeichert haben, werden die Optionen zuerst aus `/opt/labview/labviewrc` entnommen. Wenn Sie im Dialogfenster **Options** eine Einstellung ändern, wie zum Beispiel die Schriftart einer Applikation, speichert LabVIEW die Änderung in der `.labviewrc`-Datei. Beim nächsten Start wird dann anstelle der voreingestellten Applikationsschriftart aus der Datei `/opt/labview/labviewrc` die unter `.labviewrc` vorhandene verwendet.

Optionseinträge bestehen aus einer Option gefolgt von einem Namen, einem Doppelpunkt und einem Wert. Der Name der Option ist der Programmcode gefolgt von einem Punkt (`.`) und einer Option. Bei der Suche nach Optionsnamen wird auch die Groß- und Kleinschreibung beachtet. Der Wert kann in doppelten (`"`) oder einfachen Anführungszeichen (`'`) angegeben werden. Um zum Beispiel als Voreinstellung die Genauigkeit “Double” zu verwenden, fügen Sie der `.labviewrc`-Datei im Stammverzeichnis folgenden Eintrag hinzu:

```
labview.defPrecision : double
```

Wenn Sie eine andere Optionen-Datei verwenden möchten, geben Sie diese beim Start von LabVIEW in der Befehlszeile an. Um zum Beispiel anstelle der Datei `.labviewrc` eine Datei mit dem Namen `lvrc` aus dem Verzeichnis `test` zu verwenden, geben Sie `labview -pref/test/lvrc` ein. Alle Änderungen im Dialogfenster **Optionen** werden nun in der Optionsdatei `lvrc` abgespeichert. Wenn Sie in der Befehlszeile eine Optionsdatei angeben, verwendet LabVIEW zwar die `labviewrc`-Datei im Programmverzeichnis, jedoch werden widersprüchliche Einträge in diesem Verzeichnis von der Datei überschrieben.

4

Erstellen des Frontpanels

Das Frontpanel ist die Benutzeroberfläche eines VIs. In der Regel entwerfen Sie zuerst das Frontpanel und dann das Blockdigramm, um den Ein- und Ausgabeelementen, die Sie auf dem Frontpanel zusammengestellt haben, Funktionen zuzuweisen. Weitere Informationen über das Blockdiagramm finden Sie in Kapitel 5, *Erstellen des Blockdiagramms*.

Das Frontpanel wird mit Hilfe von Bedien- und Anzeigeelementen erstellt, welche die interaktiven Ein- und Ausgangsanschlüsse eines VIs darstellen. Bedienelemente sind Knöpfe, Drucktasten, Drehregler und andere Eingabeelemente. Anzeigeelemente sind Graphen, LEDs und andere Anzeigen. Mit den Bedienelementen werden die Eingabegeräte simuliert und Daten an das Blockdiagramm des VIs übergeben. Mit Anzeigeelementen werden die Ausgabegeräte nachgeahmt und die Daten angezeigt, die vom Blockdiagramm erfasst oder erzeugt werden.

Wählen Sie **Fenster»Elementepalette**, um die Palette **Elemente** anzuzeigen. Klicken Sie dann auf ein Bedien- oder Anzeigeelement auf der Palette **Elemente** und platzieren Sie es auf das Frontpanel.

Weitere Informationen ...

Weitere Informationen zum Entwerfen und Konfigurieren des Frontpanels finden Sie in der *LabVIEW-Hilfe*.

Konfiguration von Frontpanel-Objekten

Darstellungsart und Funktionsmerkmale von Frontpanel-Objekten können entweder über das Dialogfeld “Eigenschaften” oder über die Kontextmenüs des Frontpanels eingestellt werden. Mit dem Dialogfenster “Eigenschaften” haben Sie die Möglichkeit, zu einem Bedien- oder Anzeigeelement mehrere Einstellungen auf einmal vorzunehmen. Ein weiterer Vorteil besteht darin, dass zu diesem Dialogfeld auch eine Kontext-Hilfe verfügbar ist. Kontextmenüs bieten sich hingegen zur schnellen Konfiguration einzelner Eigenschaften von Bedien- und Anzeigeelementen an. Die im Dialogfeld “Eigenschaften” und im Kontextmenü angezeigten Optionen variieren je nach Frontpanel-Objekt. Grundsätzlich gilt, dass durch Ände-

rungen, die über Kontextmenüs vorgenommen wurden, die entsprechenden Einstellungen im Dialogfeld überschrieben werden. Weitere Informationen zum Erstellen und zur Verwendung benutzerdefinierter Bedien- und Anzeigeelemente und zu Typdefinitionen finden Sie in der Application Note *LabVIEW Custom Controls, Indicators, and Type Definitions*.

Um zu einem bestimmten Objekt das Dialogfenster “Eigenschaften” aufzurufen, klicken Sie das Objekt auf dem Frontpanel mit der rechten Maustaste an und wählen Sie **Eigenschaften**. Das gilt jedoch nur, wenn sich das VI im Bearbeitungsmodus befindet. Während der Ausführung eines VIs sind diese Einstellungen nicht möglich.

Ein- und Ausblenden von optionalen Elementen

Bedien- und Anzeigeelemente auf dem Frontpanel verfügen über optionale Elemente, die Sie ein- und ausblenden können. Um festzulegen, welche Elemente des Bedien- oder Anzeigeelements auf dem Frontpanel angezeigt werden sollen und welche nicht, wählen Sie die Registerkarte **Erscheinungsbild** des Dialogfeldes “Eigenschaften”. Um dieselbe Einstellung im Kontextmenü vorzunehmen, klicken Sie das Frontpanel-Objekt mit der rechten Maustaste an und wählen Sie unter **Sichtbare Objekte** die anzuzeigenden Komponenten aus. Die meisten Objekte verfügen über eine Beschriftung und über einen Untertitel. Weitere Informationen zu Beschriftungen und Untertiteln finden Sie im Abschnitt *Beschriftungen* dieses Kapitels.

Umwandeln von Bedienelementen in Anzeigeelemente und umgekehrt

Die Objekte der Palette **Elemente** sind auf Grundlage ihres typischen Einsatzspektrums als Bedien- oder Anzeigeelemente aufgeführt. Wenn Sie beispielsweise einen Umschalter wählen, erscheint dieser auf dem Frontpanel als Bedienelement, da ein Umschalter in der Regel ein Eingabegerät ist. Wenn Sie jedoch eine LED wählen, erscheint diese auf dem Frontpanel als Anzeigeelement, da eine LED normalerweise als Ausgabeelement dient.

Einige Paletten enthalten ein Bedien- und ein Anzeigeelement für den gleichen Typ oder die gleiche Klasse von Objekten. So enthält beispielsweise die Palette **Numerisch** ein digitales Bedien- und ein digitales Anzeigeelement.

Sie haben jedoch auch die Möglichkeit, ein Bedien- in ein Anzeigeelement umzuwandeln und umgekehrt. Klicken Sie dazu ein Objekt mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **In Anzeigeelement umwandeln**, wenn es sich um ein Bedienelement handelt, bzw.

In Bedienelement umwandeln, wenn das Element ein Anzeigeelement ist.

Ersetzen von Frontpanel-Objekten

Frontpanel-Objekte lassen sich durch andere Bedien- oder Anzeigeelemente ersetzen. Wenn Sie mit der rechten Maustaste auf ein Objekt klicken und aus dem Kontextmenü **Ersetzen** wählen, wird eine temporäre **Elementepalette** angezeigt, auch wenn die Palette bereits geöffnet ist. Wählen Sie daraus das Bedien- oder Anzeigeelement, durch das das markierte Objekt ersetzt werden soll.

Mit der Auswahl von **Ersetzen** aus dem Kontextmenü werden so viele Informationen wie möglich über das ursprüngliche Objekt beibehalten, wie beispielsweise der Name, die Beschreibung, die Standarddaten, Datenflussrichtung (Bedien- oder Anzeigeelement), Farbe, Größe und so weiter. Das neue Objekt behält jedoch seinen eigenen Datentyp bei. Verbindungen vom Anschluss des Objekts oder lokale Variablen verbleiben im Blockdiagramm, können jedoch unterbrochen werden. Wenn Sie beispielsweise einen numerischen Anschluss durch einen String-Anschluss ersetzen, bleibt die ursprüngliche Verbindung im Blockdiagramm erhalten, wird jedoch unterbrochen dargestellt.

Je mehr das neue Objekt dem zu ersetzenden Objekt ähnelt, desto mehr der ursprünglichen Eigenschaften können beibehalten werden. Wenn Sie beispielsweise einen Schieber durch einen anders gearteten Schieber ersetzen, weist der neue Schieber die gleiche Höhe und Skala, den gleichen Wert und Namen, die gleiche Beschreibung und so weiter auf. Wenn Sie den Schieber jedoch durch ein String-Bedienelement ersetzen, behält LabVIEW nur den Namen, die Beschreibung und die Datenflussrichtung bei, da ein Schieber nicht viel mit einem String-Bedienelement gemeinsam hat.

Sie können auch Objekte aus der Zwischenablage einfügen, um vorhandene Frontpanel-Bedien- und Anzeigeelemente zu ersetzen. Bei dieser Methode werden keine Eigenschaften des alten Objekts beibehalten, jedoch bleibt die Verbindung zum Objekt bestehen.

Konfiguration des Frontpanels

Frontpanel-Objekte können auf vielfältige Weise benutzerdefiniert angepasst werden. So kann neben der Tabulatorreihenfolge für Frontpanel-Objekte beispielsweise festgelegt werden, dass sich Objekte automatisch der Größe des Frontpanels anpassen sollen.

Festlegen von Tastenkombinationen für Bedienelemente

Bedienelementen können Tastenkombinationen zugewiesen werden, so dass diese auch über die Tastatur steuerbar sind. Klicken Sie zur Anzeige des Dialogfeldes **Tastenbelegung** mit der rechten Maustaste auf das Bedienelement und wählen Sie aus dem Kontextmenü **Fortgeschritten» Tastenbelegung** aus.

Hinweis Tastenkombinationen funktionieren nur bei nicht verborgenen Elementen.

Wenn ein Benutzer die Tastenkombination eingibt, während das VI ausgeführt wird, erhält das entsprechende Bedienelement den Fokus. Wenn es sich bei dem Bedienelement um ein Text- oder ein numerisches Bedienelement handelt, hebt LabVIEW den Text hervor, damit Sie ihn bearbeiten können. Wenn das Bedienelement vom Typ “boolesch” ist, können Sie den Wert durch Betätigen der Leertaste bzw. der <Eingabe>-Taste verändern.

Bei Anzeigeelementen ist der Menüpunkt **Fortgeschritten»Tastenbelegung** grau hinterlegt, da in ein Anzeigeelement keine Daten eingegeben werden können.

Hinweis Mit einem Key-Down-Ereignis lässt sich durch Tastendruck des Benutzers ein Ereignis auslösen.

Weitere Informationen zum Festlegen von Tastenkombinationen in einer Benutzeroberfläche finden Sie in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*, unter der Überschrift *Key Navigation*.

Steuern des Schaltflächenverhaltens per Tastenbelegung

Sie können verschiedenen Schaltflächen, die das Verhalten eines Frontpanels steuern, Funktionstasten zuweisen. Eine Schaltfläche in einem VI kann so konfiguriert werden, dass sie sich wie die aktive Schaltfläche eines Dialogfeldes verhält. Ein Drücken der <Eingabe>-Taste wäre also gleichbedeutend mit einem Mausklick auf die Schaltfläche.

Wenn die <Eingabe>-Taste einer Dialogfeldschaltfläche zugewiesen wird, zieht LabVIEW um diese Schaltfläche automatisch einen dickeren Rahmen.

Wenn einem Bedienelement die <Eingabe>-Taste zugewiesen wird, kann in keines der String-Bedienelemente mehr ein Zeilenumbruch eingegeben werden. Dementsprechend sind alle Strings auf diesem Frontpanel auf eine

einzige Zeile beschränkt. Längere Strings können mit Hilfe der Bildlaufleisten angezeigt und bearbeitet werden.

Wenn der Tastaturfokus auf einem booleschen Bedienelement liegt und Sie die <Eingabe>-Taste drücken, ändert dieses immer seinen Zustand, auch wenn die <Eingabe>-Taste eigentlich einem anderen Element zugeordnet ist. Die Zuweisung der <Eingabe>-Taste ist nur gültig, wenn kein boolesches Bedienelement ausgewählt ist.

Festlegen der Tabulatorreihenfolge von Frontpanel-Objekten

Für das Aktivieren der Bedien- und Anzeigeelemente auf einem Frontpanel gibt es eine Reihenfolge, die als Tabulatorreihenfolge bezeichnet wird und die von der Position der Elemente auf dem Frontpanel unabhängig ist. Das erste Bedien- oder Anzeigeelement, das Sie auf dem Frontpanel erstellen, ist Element `0`, das zweite ist `1` und so weiter. Wenn Sie ein Bedien- oder Anzeigeelement löschen, wird die Tabulatorreihenfolge automatisch angepasst.

Die Tabulatorreihenfolge bestimmt die Reihenfolge, in der die Bedien- und Anzeigeelemente beim Drücken der <Tabulator>-Taste ausgewählt werden, während ein VI ausgeführt wird. Außerdem ist durch sie die Anordnung der Bedien- und Anzeigeelemente in den beim Protokollieren der Frontpanel-Daten erstellten Datenprotokolldateien festgelegt. Weitere Informationen zum Protokollieren von Daten finden Sie in Abschnitt *Protokollieren von Frontpanel-Daten* des Kapitels 14, *Datei-I/O*.

Die Tabulatorreihenfolge von Frontpanel-Objekten lässt sich über **Bearbeiten»Tab-Reihenfolge setzen** auswählen.

Um zu verhindern, dass Benutzer mit Hilfe der <Tabulator>-Taste auf ein Bedienelement zugreifen, während das VI ausgeführt wird, aktivieren Sie im Dialogfeld **Tastenbelegung** das Kontrollkästchen **Dieses Bedienelement bei Tabulator überspringen**.

Zuweisen von Farben zu Objekten

Die meisten Objekte können in ihrer Farbe verändert werden. Für Blockdiagrammanschlüsse von Frontpanel-Objekten und für Verbindungen werden zum Beispiel bestimmte Farben für Typ und Darstellung der Daten verwendet, die über diese Anschlüsse und Verbindungen übertragen werden. Aus diesem Grund ist eine Farbänderung hier nicht möglich.

Aktivieren Sie das Farbwerkzeug und klicken Sie mit der rechten Maustaste auf ein Objekt oder auf den Arbeitsbereich, um die Farbe von Frontpanel-Objekten oder dem Frontpanel- bzw. Blockdiagramm-Arbeitsbereich zu ändern. Um die Standardfarbe der meisten Objekte zu ändern, klicken Sie auf **Werkzeuge»Optionen** und wählen im Pulldown-Menü die Option **Farben**.

Weitere Informationen zur farblichen Gestaltung der Benutzeroberfläche finden Sie in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*, im Abschnitt *Colors*.

Verwendung importierter Grafiken

In LabVIEW können Grafiken aus anderen Applikationen importiert werden, um sie als Frontpanel-Hintergrund, Elemente in Ring-Bedienelementen oder als Bestandteil anderer Bedien- und Anzeigeelemente zu verwenden. Weitere Informationen zur Verwendung von Grafiken in Bedienelementen finden Sie in den Application Notes *LabVIEW Custom Controls, Indicators, and Type Definitions*.

In LabVIEW werden die meisten Standard-Grafikformate unterstützt. Dazu gehören GIF, MNG (beide auch animiert), BMP, JPEG und PNG. Außerdem ist es möglich, Objekte transparent zu gestalten.

Zum Importieren einer Grafik kopieren Sie diese in die Zwischenablage und fügen sie dann in das Frontpanel ein. Sie können auch **Bearbeiten» Bild aus Datei importieren** wählen.

Hinweis (Windows und Mac OS) Bilder, die über Kopieren und Einfügen importiert werden, verlieren ihre Transparenz.

Beispiele für Bedienelemente mit importierten Grafiken finden Sie in der Bibliothek `examples\general\controls\custom.llb`. Weitere Informationen zur Gestaltung der Benutzeroberfläche mit Grafiken finden Sie in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*, im Abschnitt *Graphics and Custom Controls*.

Ausrichten und Einteilen von Objekten

Wählen Sie **Ausführen»Ausrichtungsgitter des Panels aktivieren**, um Objekte beim Platzieren in das Frontpanel automatisch am Hilfsgitter auszurichten. Zum Deaktivieren dieser Funktion wählen Sie **Ausführen» Ausrichtungsgitter des Panels deaktivieren**. Die Objekte müssen dann manuell ausgerichtet werden. Zum Umschalten zwischen beiden Optionen

kann auch die Tastenkombination <Strg-#> verwendet werden. Auf französischen Tastaturen verwenden Sie bitte die Tastenkombination <Strg-”>.

(Mac OS) Drücken Sie <Befehlstaste-*>. **(Sun)** Drücken Sie <Meta-#>. **(Linux)** Drücken Sie <Alt-#>.

Auch für das Blockdiagramm ist ein Ausrichtungsgitter verfügbar.

Wählen Sie zur Anzeige des Gitters **Werkzeuge»Optionen** und wählen Sie aus dem oberen Pulldown-Menü die Option **Ausrichtungsgitter** aus. Hier können auch benutzerspezifische Einstellungen für das Gitter vorgenommen werden.

Um in das Frontpanel oder Blockdiagramm platzierte Objekte auszurichten, markieren Sie sie und wählen Sie in der Symbolleiste die Schaltfläche **Objekte ausrichten**. Daraufhin öffnet sich ein Pulldown-Menü. Um gleichmäßige Abstände zwischen den Objekten zu erzielen, markieren Sie die Objekte und wählen Sie in der Symbolleiste das Pulldown-Menü **Objekte einteilen** aus.

Gruppieren und Sperren von Objekten

Verwenden Sie das Positionierwerkzeug, um die Frontpanel-Objekte auszuwählen, die gruppiert und gesperrt werden sollen. Klicken Sie in der Symbolleiste auf die Schaltfläche **Neuordnen** und wählen Sie aus dem Kontextmenü **Gruppe** oder **Sperre**. Gruppierte Objekte behalten ihre relative Anordnung und Größe bei, wenn Sie sie mit dem Positionierwerkzeug verschieben oder deren Größe ändern. Gesperrte Objekte behalten ihre Position auf dem Frontpanel bei. Zum Löschen gesperrter Objekte müssen Sie erst die Sperre entfernen. Objekte können gleichzeitig gruppiert und in ihrer Position fixiert werden. Mit Ausnahme des Positionierwerkzeugs können in der Regel alle Werkzeuge auf gruppierte oder gesperrte Objekte angewendet werden.

Ändern der Größe von Objekten

Die meisten Frontpanel-Objekte sind in ihrer Größe veränderlich. Wenn das Positionierwerkzeug über ein solches Objekt bewegt wird, erscheinen Ziehpunkte oder Kreise, mit denen das Objekt auf die gewünschte Größe gezogen werden kann. Die Schriftgröße bleibt unabhängig von der Größe des Objekts immer gleich. Wenn die Größe einer Objektgruppe geändert wird, ändert sich die Größe aller Objekte in der Gruppe.

Einige Objekte lassen sich nur in eine Richtung erweitern, zum Beispiel digitale numerische Bedien- und Anzeigeelemente. Andere behalten ihre

Proportionen bei, wenn sie in der Größe geändert werden, wie beispielsweise Drehknöpfe. Das Positionierwerkzeug bleibt gleich, jedoch lässt sich die gestrichelte Linie um das Objekt nur in eine Richtung bewegen.

Beim Ändern der Größe eines Objekts können Sie die Vergrößerungsrichtung manuell begrenzen. Um ein Objekt entweder nur in horizontale oder vertikale Richtung aufzuziehen oder die aktuellen Proportionen beizubehalten, drücken Sie während des Vorgangs die <Umschalt>-Taste. Um beim Vergrößern oder Verkleinern den Mittelpunkt des Objekts beizubehalten, halten Sie während des Vorgangs die <Strg>-Taste gedrückt.

(Mac OS) Drücken Sie <Befehlstaste>. **(Sun)** Drücken Sie die <Meta>-Taste. **(Linux)** Drücken Sie die <Alt>-Taste.

Um mehrere Frontpanel-Elemente auf die gleiche Größe zu bringen, markieren Sie diese und klicken Sie in der Menüleiste auf die Schaltfläche **Objektgröße verändern**. Hier können entweder alle ausgewählten Objekte auf die Höhe/Breite des größten/kleinsten Objekts oder auf eine bestimmte Größe in Pixeln eingestellt werden.

Skalieren von Frontpanel-Objekten

Frontpanel-Objekte können so eingestellt werden, dass sie sich in der Größe stets automatisch dem Frontpanel-Fenster anpassen. Diese Einstellung ist sowohl für einzelne Objekte als auch für den gesamten Inhalt des Frontpanels verfügbar. Mehrere Objekte auf dem Frontpanel können Sie jedoch erst dann gemeinsam skalieren, wenn Sie für alle die Skalierung aktiviert oder die Objekte vorher gruppiert haben. Zum Skalieren eines Objekts markieren Sie es und wählen dann **Bearbeiten»Objekt mit Panel skalieren**.

Wenn Sie für ein einzelnes Objekt die Skalierung aktiviert haben, ändert das Objekt automatisch seine Größe in Relation zu jeder beliebigen Änderung der Größe des Frontpanel-Fensters. Alle Frontpanel-Objekte, die sich nicht der Fenstergröße anpassen, ordnen sich dabei jedesmal so an, dass ihre ursprüngliche Position zum skalierten Objekt gleich bleibt.

Wenn für ein einzelnes Objekt die automatische Skalierung aktiviert wurde, werden, wie in Abbildung 4-1 dargestellt, auf dem Frontpanel in bestimmten Bereichen graue Linien angezeigt. Diese Bereiche kennzeichnen die Positionen der anderen Frontpanel-Objekte in Relation zum Objekt, das Sie skalieren möchten. Wenn Sie die Größe des Frontpanel-Fensters ändern, positioniert sich das zu skalierende Objekt automatisch neu in Relation zu seiner ursprünglichen Position, wobei seine Größe angepasst wird. Wenn das VI ausgeführt wird, verschwinden die grauen Linien.

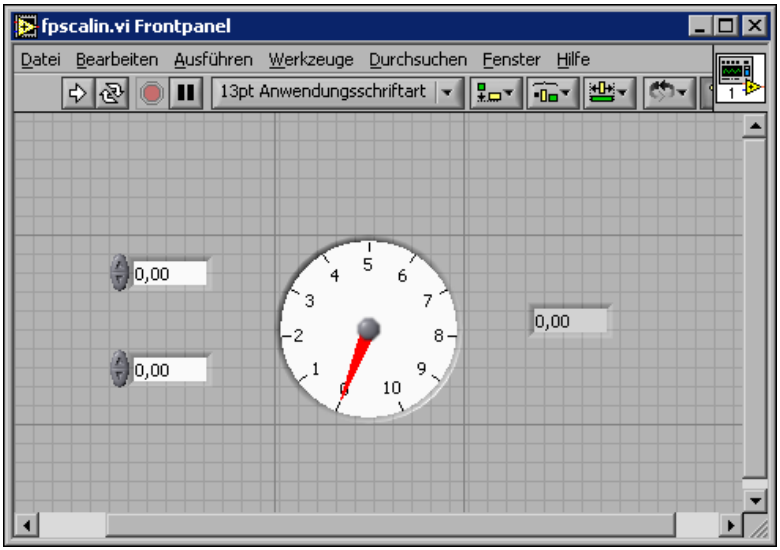


Abbildung 4-1. Frontpanel mit Objekt, das sich der Fenstergröße anpasst

Wenn LabVIEW Objekte automatisch skaliert, gelten die gleichen Konventionen wie beim manuellen Ändern der Größe von Objekten. Beispielsweise können einige Objekte nur horizontal oder vertikal in der Größe geändert werden, und der Schriftgrad bleibt unverändert, wenn Sie die Größe eines Objekts ändern.

Nachdem LabVIEW ein Objekt automatisch skaliert hat, kann es vorkommen, dass das Objekt nicht wieder seine exakte ursprüngliche Größe annimmt, wenn Sie das Fenster in die ursprüngliche Größe und Position bringen. Bevor Sie das VI speichern, wählen Sie daher **Bearbeiten»Rückgängig**, um die Originalgröße von Frontpanel-Fenster und Objekten wiederherzustellen.

Sie können die Skalierung für ein Array aktivieren oder die Objekte innerhalb eines Arrays skalieren. Wenn Sie die Skalierung für ein Array aktivieren, passen Sie die Anzahl der Zeilen und Spalten an, die im Array angezeigt werden. Wenn Sie die Skalierung für die Objekte im Array aktivieren, wird immer die gleiche Anzahl Zeilen und Spalten angezeigt, allerdings in unterschiedlicher Größe.

Sie können ferner die Skalierung für einen Cluster aktivieren oder die Objekte innerhalb eines Clusters skalieren. Wenn Objekte innerhalb eines Clusters skaliert werden, passt sich der Cluster-Container automatisch an.

Hinzufügen einer Freifläche auf dem Frontpanel ohne Größenänderung des Fensters

Sie können dem Frontpanel Leerraum hinzufügen, ohne die Größe des Fensters zu ändern. Um den Abstand zwischen zu nahe beieinander liegenden Objekten zu vergrößern, drücken Sie die <Strg>-Taste und ziehen Sie mit dem Positionierwerkzeug im Frontpanel-Arbeitsbereich zwischen den Elementen einen Bereich auf. Ziehen Sie den Bereich in der Größe auf, wie Sie Freiraum einfügen möchten.

(Mac OS) Drücken Sie <Befehlstaste>. **(Sun)** Drücken Sie die <Meta>-Taste. **(Linux)** Drücken Sie die <Alt>-Taste.

Ein von einem gestrichelten Rahmen umschlossenes Rechteck zeigt an, wo die Freifläche eingefügt wird. Lassen Sie die Taste los, um den Freiraum einzufügen.

Bedien- und Anzeigeelemente des Frontpanels

Verwenden Sie zur Erstellung des Frontpanels die Bedien- und Anzeigeelemente aus der Palette **Elemente**. Bedienelemente sind Knöpfe, Drucktasten, Drehregler und andere Eingabeelemente. Anzeigeelemente sind Graphen, LEDs und sonstige Anzeigen. Mit den Bedienelementen werden Eingabegeräte simuliert und Daten an das Blockdiagramm des VIs übergeben. Mit Anzeigeelementen werden Ausgabegeräte nachgeahmt und die Daten angezeigt, die vom Blockdiagramm erfasst oder erzeugt werden.

3D- sowie klassische Bedien- und Anzeigeelemente

Viele Frontpanel-Objekte werden dreidimensional und in High-Color-Auflösung dargestellt. Stellen Sie Ihren Monitor zur optimalen Anzeige der Elemente auf mindestens 16-Bit-Farben ein.

Anstelle der 3D-Frontpanel-Objekte sind auch zweidimensionale Objekte mit geringerer Farbtiefe verfügbar. Die 2D-Bedien- und Anzeigeelemente unter **Klassische Elemente** sind für Monitore gedacht, die auf High Color (16 Bit) bzw. 256 Farben eingestellt sind.

Um die Darstellung der Elemente festzulegen, die bei einem Rechtsklick auf einen Anschluss erzeugt werden, wenn aus dem Kontextmenü **Erstelle»Bedienelement** oder **Erstelle»Anzeigeelement** ausgewählt wird, wählen Sie **Datei»VI-Einstellungen** und klicken Sie im Pulldown-Menü **Kategorie** auf **Editor-Optionen**. Weitere Informationen zur Konfiguration des Aussehens und des Verhaltens von VIs finden Sie im Abschnitt *Verhal-*

ten und Erscheinungsbild von VIs konfigurieren des Kapitels 16, *Anpassen von VIs*. Um die Darstellungsart von Elementen in neuen VIs festzulegen, die über einen Rechtsklick auf einen Anschluss und **Erstelle»Bedienelement** bzw. **Erstelle»Anzeigeelement** erzeugt werden, wählen Sie **Werkzeuge»Optionen** und klicken Sie im Pulldown-Menü auf **Frontpanel**.

Schieberegler, Drehschalter, Drehregler, numerische Anzeigen und Zeitstempel

Mit den numerischen Bedien- und Anzeigeelementen auf den Paletten **Numerisch** und **Klassische Elemente** können Schieberegler, Drehknöpfe, Drehregler und numerische Anzeigen nachgebildet werden. Hier finden Sie neben Farbboxen und Farbrampen zum Festlegen von Farbwerten auch Zeitstempel, mit deren Hilfe Messdaten mit einer Zeit und einem Datum verknüpft werden können. Numerische Bedien- und Anzeigeelemente dienen zur Eingabe bzw. Anzeige numerischer Daten.

Schieberegler und Anzeigen

In LabVIEW sind neben vertikalen und horizontalen Schiebereglern auch Verlaufsanzeigen, ein Tank und ein Thermometer verfügbar. Um den Wert eines Schiebereglers oder einer Verlaufsanzeige zu ändern, ziehen Sie entweder den Balken oder Zeiger an eine neue Position, klicken mit dem Bedienwerkzeug an eine bestimmte Stelle oder verwenden die optionale numerische Anzeige. Wenn Sie den Schieber an eine neue Position ziehen und das VI während der Änderung aktiv ist, übergibt das Bedienelement Zwischenwerte an das VI, und zwar abhängig davon, wie häufig das VI das Bedienelement liest.

Schieberegler oder Anzeigen können mehr als einen Wert anzeigen. Klikken Sie mit der rechten Maustaste auf das Objekt, und wählen Sie **Schieber hinzufügen** aus dem Kontextmenü, um weitere Schieber hinzuzufügen. Der Datentyp eines Bedienelements mit mehreren Schiebern ist ein Cluster, der jeden der numerischen Werte enthält. Weitere Informationen zu Clustern finden Sie im Abschnitt *Cluster* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Drehbare Bedienelemente und Anzeigen

Zu den drehbaren Bedienelementen und Anzeigen gehören Drehknöpfe, Drehregler, Runduminstrumente und Drehspulinstrumente. Drehbare Objekte funktionieren ähnlich wie Schieber oder Verlaufsanzeigen. Sie ändern den Wert eines drehbaren Bedien- oder Anzeigeelements,

indem Sie auf den Zeiger bzw. auf den Drehknopf klicken und diesen in die gewünschte Position ziehen oder direkt den gewünschten Wert in das digitale Display eingeben.

Drehbare Bedien- oder Anzeigeelemente können mehr als einen Wert anzeigen. Klicken Sie zum Hinzufügen neuer Zeiger mit der rechten Maustaste auf das Objekt und wählen Sie aus dem Kontextmenü **Zeiger hinzufügen**. Der Datentyp eines Bedienelements mit mehreren Zeigern ist ein Cluster, der jeden der numerischen Werte enthält. Weitere Informationen zu Clustern finden Sie im Abschnitt *Cluster* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Numerische Bedien- und Anzeigeelemente

Mit numerischen Bedien- und Anzeigeelementen lassen sich numerische Daten am einfachsten ein- und ausgeben. Sie können diese Frontpanel-Objekte horizontal in der Größe ändern, um weitere Stellen anzuzeigen. Sie können den Wert eines numerischen Bedienelements oder einer numerischen Anzeige ändern, indem Sie wie nachstehend beschrieben vorgehen:

- Verwenden Sie das Bedienwerkzeug oder das Beschriftungswerkzeug, klicken Sie hiermit in das Fenster der numerischen Anzeige und geben Sie die Werte über die Tastatur ein.
- Klicken Sie mit dem Bedienwerkzeug auf die Pfeiltasten eines digitalen Bedienelements.
- Platzieren Sie den Cursor mit Hilfe des Bedien- oder des Beschriftungswerkzeugs rechts neben die zu ändernde Ziffer, und drücken Sie dann die Taste “Nach oben” oder “Nach unten”.

Festlegen des numerischen Formats

Per Voreinstellung erfolgt die Anzeige und Speicherung von Zahlen in LabVIEW wie in einem Taschenrechner. Das heißt, mit einem numerischen Bedien- bzw. Anzeigeelement können maximal sechs Stellen angezeigt werden. Wenn diese Anzahl überschritten wird, erfolgt eine automatische Umschaltung in die Exponentialdarstellung. Die Einstellung, ab wie vielen Stellen die Exponentialschreibweise verwendet werden soll, kann jedoch auch verändert werden. Klicken Sie dazu im Dialogfeld **Eigenschaften numerischer Elemente** auf die Registerkarte **Format und Genauigkeit**.

Die ausgewählte Genauigkeit wirkt sich lediglich auf die Anzeige des Wertes aus. Die interne Genauigkeit richtet sich weiterhin nach dem Datentyp.

Zeitstempel-Element und -Anzeige

Mit dem Zeitstempel-Element werden Daten im Blockdiagramm mit einem Datum und einer Zeit versehen, und mit der Zeitstempel-Anzeige ist es möglich, diese Informationen anzeigen zu lassen. Um den Wert eines Zeitstempel-Elements zu verändern, gibt es mehrere Möglichkeiten:

- Klicken Sie mit der rechten Maustaste auf das Element und wählen Sie im Kontextmenü die Option **Format & Genauigkeit** aus.

- Klicken Sie auf die **Zeit/Datum**-Suchtaste, die links abgebildet ist. Sie sehen dann das Dialogfeld **Datum und Zeit setzen**.
- Wählen Sie aus dem Kontextmenü **Datenoperationen»Datum und Zeit setzen**. Sie sehen dann das Dialogfeld **Datum und Zeit setzen**. Sie können das Element aber auch mit der rechten Maustaste anklicken und aus dem Kontextmenü **Datenoperationen»Aktuelle Zeit verwenden** wählen.

Farbboxen

In einer Farbbox wird die Farbe angezeigt, die einem festgelegten Wert entspricht. Mit Farbboxen können auch verschiedene Bedingungen angezeigt werden, wie zum Beispiel, wenn Werte einen zulässigen Bereich überschreiten. Der Farbwert wird als Hexadezimalzahl im Format RRGGBB ausgedrückt. Die ersten beiden Ziffern steuern den Farbwert für Rot. Mit dem zweiten Ziffernpaar wird der Farbwert für Grün festgelegt. Mit den letzten zwei Stellen wird der Wert für Blau bestimmt.

Um die Farbe einer Farbbox festzulegen, klicken Sie diese mit dem Bedien- oder Farbwerkzeug an. Daraufhin wird eine Farbwahltabelle angezeigt.

Farbrampen

Bei einer Farbrampe wird ein numerischer Wert durch eine Farbe dargestellt. Eine Farbrampe besteht aus wenigstens zwei beliebigen Markern, von denen jeder über einen numerischen Wert und eine entsprechende Anzeigefarbe verfügt, und muss entsprechend konfiguriert werden. Wenn sich der Eingangswert ändert, ändert sich die Farbe in die Farbe, die diesem neuen Wert entspricht. Farbrampen eignen sich für die visuelle Anzeige von Datenbereichen, wie beispielsweise einem Warnungsbereich für den Fall, dass ein Runduminstrument in einen gefahrenträchtigen Wertebereich eintritt. Eine Farbrampe eignet sich auch beispielsweise zur Festlegung einer Farbskala für Intensitätsdiagramme und -graphen. Im Abschnitt *Intensitätsgraphen und -diagramme* des Kapitels 12, *Graphen und Diagramme*, erhalten Sie weiterführende Informationen zu Intensitätsdiagrammen und -graphen.

Über die Optionen des Kontextmenüs, das über einen Klick mit der rechten Maustaste geöffnet wird, können die Darstellung, Größe, Anzahl der Farben und die verwendeten Farben eingestellt werden.

Farbrampen können zusammen mit Drehschaltern, Drehreglern und Rundinstrumenten eingesetzt werden. Bei Drehspulinstrumenten ist standardmäßig eine Farbrampe vorhanden.

Graphen und Diagramme

Mit Graphen und Diagrammen können Daten in grafischer Form dargestellt werden.

Weitere Informationen über die Verwendung von Graphen und Diagrammen in LabVIEW finden Sie in Kapitel 12, *Graphen und Diagramme*.

Tasten, Schalter und LEDs

Die booleschen Bedien- und Anzeigeelemente umfassen Nachbildungen von Tastern, Schaltern und LEDs. Sie dienen zur Eingabe bzw. Anzeige boolescher Werte (TRUE/FALSE). Wenn Sie beispielsweise die Temperatur eines Experiments überwachen, können Sie eine boolesche Warnleuchte auf das Frontpanel platzieren, um anzuzeigen, wann die Temperatur ein bestimmtes Niveau übersteigt.

Über das Kontextmenü können Sie das Erscheinungsbild eines booleschen Objekts Ihren Bedürfnissen anpassen und festlegen, wie sich das Objekt verhalten soll, wenn es angeklickt wird.

Texteingabefelder, Beschriftungen und Pfadanzeigen

Die Stringelemente in LabVIEW sind Felder zur Eingabe von Text und Beschriftungen. In die Pfadelemente kann der Pfad zu einem Verzeichnis oder einer Datei eingegeben werden.

String-Bedien- und Anzeigeelemente

Mit Hilfe des Bedienwerkzeugs oder des Beschriftungswerkzeugs geben Sie Text in ein String-Bedienelement auf dem Frontpanel ein oder ändern den vorhandenen Text. Standardmäßig wird neuer oder geänderter Text erst dann an das Blockdiagramm übergeben, wenn Sie die Texteingabe beenden. Um den Bearbeitungsvorgang zu beenden, klicken Sie entweder an beliebiger Stelle auf das Frontpanel, wechseln zu einem anderen Fenster, klicken auf die Schaltfläche **Eingabe** in der Symbolleiste oder drücken die <Eingabe>-Taste auf dem numerischen Eingabeblock. Durch

Drücken der <Eingabe>-Taste der Tastatur wird ein Zeilenumbruch erzeugt.

Weitere Informationen zu den String-Bedien- und Anzeigeelementen entnehmen Sie bitte dem Abschnitt *Strings auf dem Frontpanel* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Kombinationsfelder

In Kombinationsfelder kann eine Liste von Strings eingegeben werden, durch die im Frontpanel gescrollt werden kann. Ein Kombinationsfeld funktioniert ähnlich wie ein Text- oder Menü-Ringelement. Der Unterschied besteht darin, dass das Kombinationsfeld den Datentyp “String” hat, während Ringelemente den Datentyp “numerisch” verwenden. Nähere Erläuterungen zu Ringelementen finden Sie im Abschnitt *Ring- und Enum-Bedien- und Anzeigeelemente* dieses Kapitels.

Ein Kombinationsfeld kann zum Beispiel auch für die Auswahl der Rahmen in einer Case-Struktur eingesetzt werden. Weitere Informationen zu Case-Strukturen finden Sie im Abschnitt *Case-Strukturen* des Kapitels 8, *Schleifen und Strukturen*.

Um Einträge in das Kombinationsfeld vorzunehmen, klicken Sie dieses mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü **Objekte bearbeiten**. Die Einträge werden im Kombinationsfeld in derselben Reihenfolge angezeigt wie auf der Registerkarte **Objekte bearbeiten** des Dialogfeldes **Kombinationsfeld-Eigenschaften**. Per Voreinstellung ist es möglich, Strings in das Kombinationsfeld einzugeben, die in der entsprechenden Liste aufgeführt sind. Wenn Sie möchten, dass nur die Strings in der Liste eingegeben werden können, deaktivieren Sie die Option **Undefinierte Strings zulassen**.

Bei der Eingabe eines Strings während der Ausführung des VIs springt LabVIEW zum ersten kürzesten String, der mit der eingegebenen Zeichenfolge beginnt. Wenn keine Übereinstimmung gefunden wird und keine undefinierten Strings zulässig sind, werden die eingegebenen Zeichen nicht angenommen.

Beim Erstellen der Liste ist es möglich, einem Eintrag einen benutzerdefinierten Wert zuzuweisen, wenn Sie möchten, dass im Blockdiagramm ein anderer String verwendet wird als der im Frontpanel angezeigte. Klicken Sie dazu mit der rechten Maustaste auf das Kombinationsfeld, wählen Sie aus dem Kontextmenü **Objekte bearbeiten** und deaktivieren Sie im Dialogfeld **Kombinationsfeld-Eigenschaften** auf der Registerkarte **Objekte**

bearbeiten die Option **Werte entsprechen Beschriftungen**. In der Spalte **Werte** ist es nun möglich, den Wert zu jedem Eintrag zu ändern.

Pfad-Bedien- und Anzeigeelemente

Mit Hilfe der Pfad-Bedien- und Anzeigeelemente ist es möglich, den Speicherort einer Datei oder eines Verzeichnisses zu bestimmen oder anzuzeigen. Diese Elemente funktionieren ähnlich wie String-Bedien- und Anzeigeelemente. Allerdings werden Pfade entsprechend der Standardsyntax der von Ihnen verwendeten Plattform formatiert.

Ungültige Pfade

Wenn eine Funktion, die einen Pfad zurückgibt, fehlschlägt, gibt die Funktion im Anzeigeelement einen ungültigen Wert für den Pfad, nämlich die Angabe `<Kein Pfad>`, zurück. Es empfiehlt sich, den Wert `<Kein Pfad>` als Standardwert für ein Pfad-Bedienelement zu verwenden. So kann festgestellt werden, wenn der Benutzer fälschlicherweise keinen Pfad angibt, und es kann für diesen Fall ein Dateidialogfeld zur Verfügung gestellt werden. Verwenden Sie dazu die Funktion “Dateidialog”.

Leere Pfade

Ein leerer Pfad in einem Pfad-Bedienelement wird unter Windows und Mac OS als leerer String und unter UNIX als Schrägstrich (`/`) angezeigt. Verwenden Sie leere Pfade, um den Benutzer zur Eingabe eines Pfades aufzufordern. Wenn Sie einen leeren Pfad mit einer Funktion zur Datei-I/O verbinden, bezieht sich der leere Pfad auf die Liste der dem Computer zugeordneten Laufwerke.

(Mac OS) Der leere Pfad verweist auf die gemounteten Datenträger.
(UNIX) Der leere Pfad verweist auf das Stammverzeichnis.

Array- und Cluster-Bedien- und Anzeigeelemente

Arrays und Cluster, mit denen Bedien- und Anzeigeelemente zusammengefasst werden können, befinden sich unter **Array & Cluster** und **Array & Cluster (klassisch)**. Für weitere Informationen zu Arrays und Clustern lesen Sie bitte den Abschnitt *Gruppieren von Daten mit Arrays und Clustern* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Außerdem sind in den Paletten **Array & Cluster** ein Variant-Element und je ein Standard-Cluster für Fehlerein- und -ausgang eisfellt enthalten. Im Abschnitt *Fehler-Cluster* des Kapitels 6, *Ausführen von VIs und Fehlersu-*

che, sind weitere Erläuterungen zu Fehler-Clustern verfügbar. Nähere Einzelheiten zum Variant-Bedienelement entnehmen Sie bitte dem Abschnitt *Verarbeiten von Variant-Daten* des Kapitels 5, *Erstellen des Blockdiagramms*.

Listenfelder, Baumstruktur-Elemente und Tabellen

Mit den Listenfeld-Bedienelementen, die sich auf den Paletten **Liste & Tabelle** und **Liste & Tabelle (klassisch)** befinden, können Einträge aufgelistet werden, aus denen der Anwender wählen kann.

Listenfelder

Listenfelder können auf Einfach- oder Mehrfachauswahl konfiguriert werden. Mit einem Listenfeld können Sie auch erweiterte Informationen über ein Element oder Ereignis, wie zum Beispiel Größe und Erstellungsdatum, darstellen.

Wird während der VI-Ausführung in das Listenfeld geschrieben, wird immer der erste Eintrag angezeigt, der mit den eingegebenen Zeichen beginnt. Mit den Tasten “Nach rechts” und “Nach links” kann zum nächsten bzw. vorhergehenden Element gewechselt werden.

Neben den Listeneintrag kann auch ein Symbol eingefügt werden, ähnlich wie im Dialogfeld **Speichern**, in dem Verzeichnisse und Dateien durch unterschiedliche Symbole gekennzeichnet werden. Die Listeneinträge können durch Trennlinien voneinander separiert werden.

Zum Ändern von Listeneinträgen und zum Abrufen bestimmter Informationen, wie zum Beispiel aktuell ausgewählter Einträge oder solcher, die doppelt angeklickt wurden sind, können auch Eigenschaftsknoten verwendet werden. Nähere Einzelheiten zu Eigenschaftsknoten entnehmen Sie bitte dem Abschnitt *Eigenschaftsknoten* des Kapitels 17, *Programmatische Steuerung von VIs*.

Baumstruktur-Elemente

Mit den Baumstruktur-Elementen können dem Anwender hierarchische Listen zur Verfügung gestellt werden, aus denen Elemente ausgewählt werden können. Dazu werden die Elemente, die in die Struktur eingegeben werden, gruppiert (zu Knoten zusammengefasst). Wenn alle in einem Knoten enthaltenen Elemente angezeigt werden sollen, ist das Symbol zum Erweitern neben dem Knoten anzuklicken. Dasselbe Symbol ist auch anzuklicken, um die Elemente wieder auszublenden.

Hinweis Das Erstellen und Bearbeiten von Baumstruktur-Elementen ist nur im LabVIEW Full und Professional Development System möglich. Wenn in einem VI ein Baumstruktur-Element vorhanden ist, kann das VI zwar auch im Base Package ausgeführt werden, jedoch ist es nicht möglich, das Element zu konfigurieren.

Wird während der VI-Ausführung in das Listenfeld geschrieben, wird immer der erste Eintrag angezeigt, der mit den eingegebenen Zeichen beginnt. Die Hierarchieebene eines Elements kann auch verändert werden. Markieren Sie es und drücken Sie die Punkttaste (`.`), um es einzurücken, oder die Kommataste (`,`), um es nach links zu verschieben.

Die Elemente in einem Verzeichnisstruktur-Element werden genauso konfiguriert wie die Elemente in einem Listenfeld. Sie können das Symbol neben jedem Knoten verändern und festlegen, ob der Anwender Elemente in das Bedienelement hineinziehen kann.

Um die Elemente in einem Verzeichnisstruktur-Element zu verändern oder Informationen wie zuletzt gelöschte oder doppelt angeklickte Elemente zu ermitteln, können Methodenknoten verwendet werden. Wenn dem Bedienelement ein Eintrag hinzugefügt wird, erstellt LabVIEW dafür ein Tag. Es wird nie dasselbe Tag mehrfach vergeben. Mit ihm können Elemente verändert oder Informationen zu Elementen programmatisch abgerufen werden. Um den Tag zu bearbeiten, klicken Sie das Baumstruktur-Element mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü **Objekte bearbeiten**. Weiterführende Informationen zu Methodenknoten finden Sie im Abschnitt *Methodenknoten* des Kapitels 17, *Programmatische Steuerung von VIs*.

Die Anwendung eines Baumstruktur-Elements wird im VI “Directory Hierarchy in Tree Control” demonstriert, das sich unter `examples\general\controls\Directory Tree Control.llb` befindet.

Tabellen

Verwenden Sie das Tabellen-Bedienelement aus der Palette **Liste & Tabelle** und **Liste & Tabelle (klassisch)**, um auf dem Frontpanel eine Tabelle zu erstellen.

Weitere Informationen zum Einsatz von Tabellenelementen finden Sie im Abschnitt *Tabellen* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Ring- und Enum-Bedien- und Anzeigeelemente

Mit Hilfe der Ring- und Enum-Bedien- und Anzeigeelemente, die sich in den Paletten **Ring & Enum** und **Ring & Enum (klassisch)** befinden, kann eine Liste von Strings erstellt werden. Beim Durchklicken der Einträge über die Pfeiltasten wird nach dem letzten Eintrag wieder der erste angezeigt.

Ring-Bedienelemente

Ring-Bedienelemente sind numerische Objekte, bei denen numerische Werte mit Strings oder Grafiken verbunden werden. Ring-Bedienelemente erscheinen als Pulldown-Menüs, in denen der Benutzer eine Auswahl treffen kann.

Ring-Bedienelemente eignen sich für die Auswahl von sich gegenseitig ausschließenden Elementen wie Triggermodi. Beispielsweise können Sie für die Benutzer ein Ring-Bedienelement bereitstellen, in dem sie zwischen kontinuierlichen, einzelnen und externen Triggern auswählen können.

Um einem Ring-Element Einträge hinzuzufügen, die beim Klicken auf die Pfeiltasten angezeigt werden sollen, klicken Sie das Element mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü **Objekte bearbeiten**. Die Reihenfolge der Einträge im Bedienelement ist dieselbe wie auf der Registerkarte **Objekte bearbeiten** des Dialogfeldes **Ringelement-Eigenschaften**. Ein Ringelement kann auch so konfiguriert werden, dass der Benutzer numerische Werte eingeben kann, die sich noch nicht in der im Fenster “Eigenschaften” festgelegten Liste befinden. Klicken Sie das Element dazu mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Undefinierte Werte zulassen**.

Zur Eingabe eines nicht voreingestellten Wertes während der Ausführung des VIs klicken Sie das Bedienelement an und wählen Sie aus dem Menü **<Andere>**. Geben Sie dann eine Zahl ein und drücken Sie die <Eingabe>-Taste. Der neue Wert erscheint dann im Bedienelement in spitzen Klammern. Jedoch wird er nicht in die Liste aufgenommen, aus der über die Pfeiltasten ein Element ausgesucht werden kann.

Beim Erstellen einer Liste für ein Ring-Bedienelement können Sie jedem Eintrag einen bestimmten numerischen Wert zuweisen. Ansonsten weist LabVIEW den Einträgen automatisch Werte zu, wobei entsprechend der Position des Eintrags mit 0 beginnend (für das erste Element) aufwärts gezählt wird. Für die Zuweisung bestimmter numerischer Werte klicken Sie das Ring-Bedienelement mit der rechten Maustaste an, wählen Sie aus dem Kontextmenü die Option **Objekte bearbeiten** und deaktivieren Sie im

Dialogfeld **Ringelement-Eigenschaften** auf der Registerkarte **Objekte bearbeiten** die Option **Sequenzielle Werte**. In der Spalte **Werte** ist es nun möglich, den Wert zu jedem Element zu verändern. Jedem Eintrag im Ring-Bedienelement muss ein einmaliger numerischer Wert zugewiesen sein.

Bedienelemente vom Typ Enum

Mit Bedienelementen des Enum-Typs lassen sich Pulldown-Menüs für Mehrfachauswahlen erzeugen. Ein Bedienelement vom Enum-Typ ist einem Text- oder Menü-Ringelement ähnlich. Der Unterschied besteht darin, dass der Datentyp bei Enum-Bedienelementen auch Informationen zu den numerischen Werten und den Beschriftungen (Strings) des Elements enthält. Dagegen ist der Datentyp von Ring-Bedienelementen “numerisch”.

Hinweis Bei Ringelementen können keine Werte eingegeben werden, die nicht vorher für das Element definiert wurden. Ebenso wenig können den Einträgen benutzerdefinierte Werte zugewiesen werden. Wenn Sie dieses Funktionsmerkmal benötigen, sollten Sie statt dessen ein Ring-Bedienelement verwenden. Nähere Erläuterungen zu Ringelementen finden Sie im Abschnitt *Ring-Bedienelemente* dieses Kapitels.

Bedienelemente vom Enum-Typ können auch dazu verwendet werden, um die Rahmen einer Case-Struktur auszuwählen. Weitere Informationen zu Case-Strukturen finden Sie im Abschnitt *Case-Strukturen* des Kapitels 8, *Schleifen und Strukturen*.

Als numerische Darstellung eines Bedienelements vom Enum-Typ kann ein 8-, 16- oder 32-Bit-Integer verwendet werden. Um die numerische Darstellung für das Element zu ändern, klicken Sie es mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Darstellung** aus.

Erweiterte Bedien- und Anzeigeelemente vom Typ Enum

Bedienelemente vom Enum-Typ werden von allen arithmetischen Funktionen außer “Inkrement” und “Dekrement” wie ein vorzeichenloser Integer-Wert behandelt. “Inkrement” erhöht den letzten aufgelisteten Wert auf den ersten, und “Dekrement” verringert den ersten aufgelisteten Wert auf den letzten aufgelisteten Wert. Wenn eine mit einem Vorzeichen versehene Ganzzahl zwangsweise in einen Enum-Typ umgewandelt wird, werden negative Zahlen gleich dem ersten Enum-Wert und außerhalb des Bereichs befindliche positive Zahlen gleich dem letzten Enum-Wert

gesetzt. Außerhalb des Bereichs befindliche vorzeichenlose Ganzzahlen werden immer dem letzten Enum-Wert gleichgesetzt.

Wenn ein Anzeigeelement vom Enum-Typ mit einem Fließkommawert verbunden wird, wandelt LabVIEW diesen in den nächsthöheren oder -tieferen numerischen Wert des Elements um. LabVIEW behandelt außerhalb des Bereichs befindliche Zahlen wie bereits beschrieben. Wenn ein Enum-Bedienelement mit einem numerischen Wert verbunden wird, dann wird der Enum-Wert in einen numerischen Wert umgewandelt. Damit ein Bedien- und ein Anzeigeelement vom Enum-Typ miteinander verbunden werden können, müssen die in beiden enthaltenen Elemente übereinstimmen. Das Anzeigeelement kann jedoch mehr Einträge enthalten als das Bedienelement.

Container-Bedienelemente

Mit den Container-Bedienelementen, die sich unter **Container** und **Container (klassisch)** befinden, können Bedien- und Anzeigeelemente gruppiert werden. Daneben ist es möglich, das Frontpanel eines anderen VIs im aktuellen VI anzuzeigen und (unter **Windows**) ActiveX-Objekte auf dem Frontpanel darzustellen. Weitere Informationen zur Verwendung von ActiveX finden Sie in Kapitel 19, *Windows-Konnektivität*, dieses Handbuchs.

Register-Bedienelemente

Mit Register-Bedienelementen können Bedien- und Anzeigeelemente auf dem Frontpanel in einem kleineren Bereich überlappt werden. Ein Register-Bedienelement besteht aus Seiten und Registern. Setzen Sie die Frontpanel-Objekte auf die jeweiligen Seiten eines Register-Bedienelements und verwenden Sie die Register zum Auswählen der verschiedenen Seiten.

Register-Bedienelemente sind bei mehreren Frontpanel-Objekten nützlich, die zusammen oder in einer spezifischen Betriebsphase verwendet werden. So könnten Sie beispielsweise mit einem VI arbeiten, bei dem der Benutzer zunächst eine Reihe von Einstellungen konfigurieren muss, bevor ein Testlauf erfolgen kann, und das es dem Benutzer ermöglicht, bestimmte Aspekte im Verlauf des Tests zu ändern. Schließlich hat der Benutzer dann die Möglichkeit, nur die relevanten Daten anzuzeigen und zu speichern.

Im Blockdiagramm wird das Register-Bedienelement standardmäßig als Bedienelement vom Typ Enum dargestellt. Anschlüsse für Bedien- und Anzeigeelemente, die in das Register-Bedienelement platziert wurden, erscheinen im Blockdiagramm als normale Anschlüsse. Weitere Informa-

tionen zu Bedienelementen vom Typ Enum finden Sie im Abschnitt *Bedienelemente vom Typ Enum* dieses Kapitels.

Unterpanel-Bedienelemente

Unterpanel-Bedienelemente dienen dazu, das Frontpanel eines anderen VIs in dem des aktuellen VIs darzustellen. Auf diese Weise können zum Beispiel Benutzeroberflächen erstellt werden, die wie Assistenten funktionieren. Platzieren Sie dazu eine **Zurück**- und eine **Vor**-Schaltfläche in das Frontpanel des übergeordneten VIs. Die Frontpanels für jeden Schritt des Assistenten können dann über ein Unterpanel-Bedienelement geladen werden.

Hinweis Das Erstellen und Bearbeiten von Unterpanel-Bedienelementen ist nur im LabVIEW Full und Professional Development System möglich. Wenn in einem VI ein Baumstruktur-Element vorhanden ist, kann das VI zwar auch im Base Package ausgeführt werden, jedoch ist es nicht möglich, das Element zu konfigurieren.

Wenn ein Unterpanel-Bedienelement in das Frontpanel platziert wird, dann enthält das entsprechende Blockdiagrammobjekt keinen Frontpanel-Anschluss. Statt dessen wird auf dem Blockdiagramm ein Methodenknoten erstellt, der auf die Methode “VI einfügen” konfiguriert ist. Um ein VI in das Unterpanel-Bedienelement zu laden, verbinden Sie den Knoten mit der entsprechenden VI-Referenz. Nähere Hinweise zur Anwendung von VI-Referenzen und Methodenknoten entnehmen Sie bitte Kapitel 17, *Programmatische Steuerung von VIs*, dieses Handbuchs.

Hinweis Da bei Unterpanel-Bedienelementen kein Anschluss vorhanden ist, können sie weder in Arrays zusammengefasst werden noch kann der Typ eines solchen Elements definiert werden. Ein Unterpanel-Bedienelement kann zwar zusammen mit anderen Elementen in einen Cluster platziert werden, jedoch dürfen sich in diesem nicht ausschließlich Unterpanel-Elemente befinden.

Im Fall, dass das zu ladende Frontpanel geöffnet ist oder bereits in ein weiteres Unterpanel-Bedienelement auf demselben Frontpanel geladen wurde, gibt LabVIEW einen Fehler aus und der Ladevorgang schlägt fehl. Frontpanel können nicht rekursiv oder aus LabVIEW-Umgebungen im Netzwerk geladen werden.

Ebenso wenig können zur Steuerung des eingebetteten Frontpanels Tastenkombinationen verwendet werden.

Aktuell nicht ausgeführte VIs werden vom VI, in dem sich das Unterpanel-Element befindet, im Ausführungsmodus geladen.

Im Unterpanel-Bedienelement wird immer nur der sichtbare Teil des geladenen Frontpanels angezeigt. Nach dem Stoppen des übergeordneten VIs wird das eingebettete Frontpanel aus dem Element entfernt. Zum Entfernen des VIs aus dem Unterpanel-Element kann aber auch die Methode “VI entfernen” verwendet werden.

Beispiele für die Verwendung von Unterpanel-Bedienelementen finden Sie unter `examples\general\controls\subpanel.llb`.

I/O-Namen-Bedien- und Anzeigeelemente

Verwenden Sie die konfigurierten I/O-Namensbedien- und -anzeigeelemente, um DAQ-Kanalnamen, VISA-Ressourcennamen und logische IVI-Namen an I/O-VIs zur Kommunikation mit einem Instrument oder DAQ-Gerät zu übergeben.

Die I/O-Namenskonstanten befinden sich auf der Palette **Funktionen**.

Hinweis Die I/O-Namensbedienelemente und -konstanten sind auf allen Plattformen komplett verfügbar. Das heißt, I/O-VIs für die Kommunikation mit plattformspezifischen Geräten können auf jedem System entwickelt werden. Eingesetzt werden können die VIs jedoch immer nur auf der Plattform, auf der das entsprechende Gerät unterstützt wird. Ansonsten wird ein Fehler ausgegeben.

(Windows) Verwenden Sie zur Konfiguration von DAQ-Kanal-, VISA-Ressourcen- und logischen IVI-Namen den im Menü **Werkzeuge** befindlichen Measurement & Automation Explorer.

(Mac OS) Zur Konfiguration der DAQ-Hardware von National Instruments steht Ihnen das NI-DAQ-Konfigurationsprogramm aus dem Menü **Werkzeuge** zur Verfügung. Mit Hilfe des DAQ-Kanal-Assistenten aus dem Menü **Werkzeuge** können Sie DAQ-Kanalnamen konfigurieren.

(Mac OS und UNIX) Verwenden Sie von VISA-Ressourcen- und logischen IVI-Namen die Konfigurationsprogramme Ihres Instruments. Weitere Informationen zu den Konfigurationsprogrammen finden Sie in der Dokumentation zum jeweiligen Instrument.

Das Bedienelement “IMAQ-Sitzung” ist ein spezieller Bezeichner, der die Verbindung zur Hardware repräsentiert.

Signalverlauf-Bedienelement

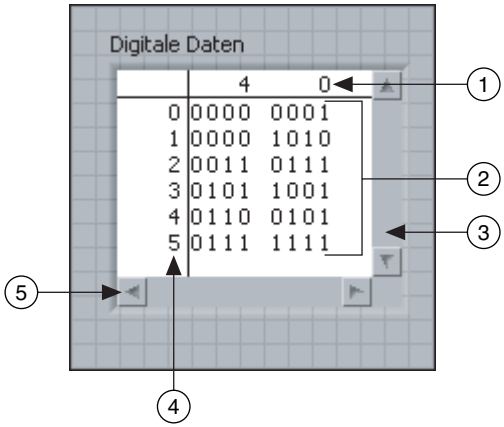
Mit dem Signalverlauf-Bedienelement können einzelne Datenpunkte eines Signals manipuliert werden. Weitere Informationen zum Datentyp “Signalverlauf” finden Sie im Abschnitt *Datentyp “Signalverlauf”* des Kapitels 12, *Graphen und Diagramme*.

Bedienelement für digitale Signalverläufe

Mit dem Bedienelement “Digitaler Signalverlauf” können einzelne Elemente eines digitalen Signals manipuliert werden. Weitere Hinweise zu diesem Element entnehmen Sie bitte dem Abschnitt *Datentyp “digitaler Signalverlauf”* des Kapitels 12, *Graphen und Diagramme*.

Bedienelement für digitale Daten

Im Bedienelement für digitale Daten sind Digitalwerte in Zeilen und Spalten angeordnet. Der Datentyp “digital” dient zur Erzeugung digitaler Signalverläufe und zur Anzeige von Daten, die aus einem digitalen Signalverlauf entnommen wurden. Mit dem Datentyp “digitaler Signalverlauf” können Daten eines digitalen Signalverlaufs mit einem entsprechenden Anzeigeelement verbunden werden. Auf diese Weise lassen sich die einzelnen Abtastungen und Signale darstellen. Im Bedienelement für Digitaldaten in Abbildung 4-2 sind fünf Abtastungen dargestellt, von denen jedes acht Signale beinhaltet.



1	Signale	4	Abtastwerte
2	Daten	5	horizontale Bildlaufleiste
3	vertikale Bildlaufleiste		

Abbildung 4-2. Bedienelement für digitale Daten

Bei Bedienelementen für digitale Daten ist es auch möglich, Zeilen einzufügen oder zu löschen. Wenn Sie eine Zeile hinzufügen möchten, klicken Sie eine Abtastung in der entsprechenden Spalte mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Zeile zuvor einfügen**. Zum Löschen einer Zeile klicken Sie eine Abtastung mit der rechten Maustaste an und wählen Sie **Zeile entfernen**. Zum Hinzufügen einer Spalte klicken Sie ein Signal in der entsprechenden Spalte mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Spalte zuvor einfügen**. Zum Entfernen einer Spalte verfahren Sie analog, wählen jedoch aus dem Kontextmenü die Option **Spalte löschen**.

In einem Bedienelement für digitale Daten ist auch das Ausschneiden, Kopieren und Einfügen von Daten möglich. Wählen Sie zum Ausschneiden die entsprechende Zeile oder Spalte aus, klicken Sie mit der rechten Maustaste und wählen Sie **Datenoperationen»Daten ausschneiden** aus dem Kontextmenü. Es können immer nur ganze Zeilen oder Spalten ausgeschnitten werden. Mit den ausgeschnittenen Daten kann keine neue Zeile oder Spalte erstellt werden. Zum Kopieren wählen Sie den entsprechenden Bereich aus, klicken Sie mit der rechten Maustaste und wählen Sie **Datenoperationen»Daten kopieren**. Wenn Sie Digitaldaten einfügen möchten, wählen Sie einen entsprechenden Bereich aus, klicken Sie mit der rechten Maustaste und wählen Sie **Datenoperationen»Daten einfügen**. Der Bereich, in den die Daten eingefügt werden sollen, muss dieselbe Dimension haben wie der Bereich, aus dem sie kopiert wurden. Wenn Sie also beispielsweise vier Datenbits aus einer Reihe kopieren, müssen Sie vier vorhandene Datenbits in derselben oder einer anderen Reihe damit überschreiben. Wenn vier Bits aus einem Bereich von zwei Spalten mal zwei Zeilen kopiert werden sollen, muss der Bereich, in den sie eingefügt werden sollen, auch aus zwei Zeilen mal zwei Spalten bestehen.

Für die Elemente “Digitale Daten” und “Digitaler Signalverlauf” sind die Werte 0, 1, L, H, Z, X, T und V zulässig. Im Element “Digitale Daten” kann darüber hinaus zwischen Binär-, Hexadezimal-, Oktal- und Dezimalformat ausgewählt werden. Die digitalen Zustände L, H, Z, X, T und V, die bei manchen Messgeräten verwendet werden, werden bei Anzeige im Hexadezimal-, Oktal- oder Dezimalformat als Fragezeichen dargestellt. Um das Datenformat festzulegen, klicken Sie das Element mit der rechten Maustaste an, wählen im Kontextmenü die Option **Datenformat** und klikken dann auf das gewünschte Format.

Umwandlung in digitale Daten

In den meisten Fällen wird ein erfasstes Signal in Form von Rohdaten ausgegeben. Um es in einem digitalen Signalverlaufsgraphen darstellen zu können, müssen die erfassten Rohdaten jedoch zunächst in den Datentyp des Graphen umgewandelt werden. Verwenden Sie dazu das VI “Analoger nach digitaler Signalverlauf”. Um aus den Daten des Typs “Digitaler Signalverlauf” die eigentlichen Digitaldaten zu extrahieren, empfiehlt es sich, die Funktion “Signalverlaufskomponenten lesen” zu verwenden.

Auf dem Blockdiagramm in Abbildung 4-3 wird die Erfassung eines sinusförmigen Signals mit einer Amplitude von 5, also einem Amplitudenbereich von -5 bis 5, simuliert. Im VI “Analoger nach digitaler Signalverlauf” in diesem Blockdiagramm wird jeder Wert durch 8 Bit dargestellt. Diese 8 Bit können sowohl den negativen Spitzenwert von -5 als auch den positiven Spitzenwert von 5 darstellen. Mit der Sonde für digitale Signalverläufe lässt sich jeweils ein Teil der ausgegebenen Werte im Binärformat anzeigen.

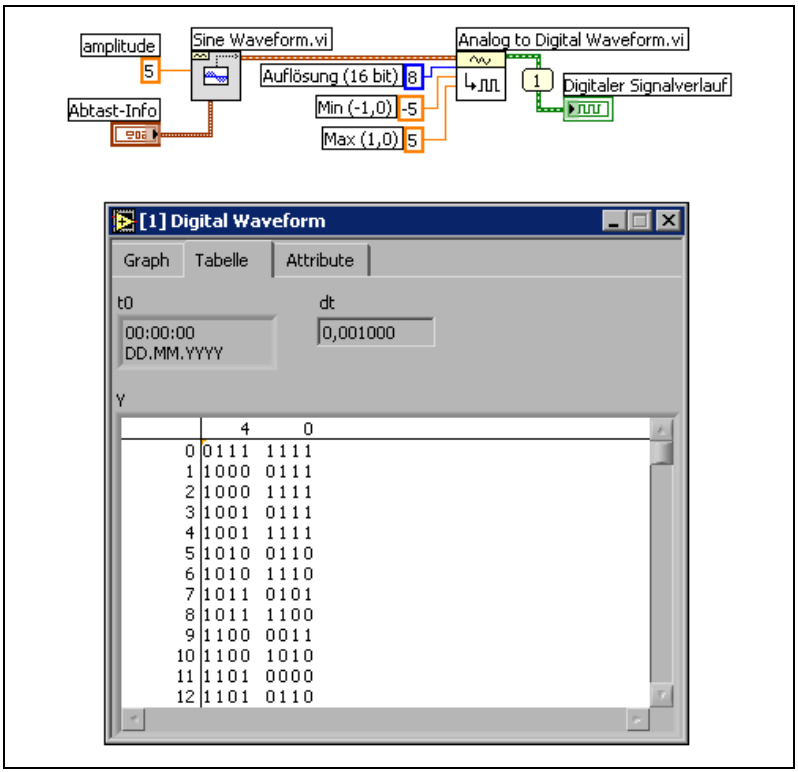


Abbildung 4-3. Umwandlung analoger in digitale Signalverläufe

Erfassen von Ausschnitten digitaler Signale

Mit dem VI “Digital: Signalausschnitt” lassen sich aus einem digitalen Signalverlauf einzelne Signale entnehmen. Im Blockdiagramm in Abbildung 4-4 sehen Sie, wie die entnommenen Einzelsignale mit anderen digitalen Daten kombiniert und so neue Signale erzeugt werden können.

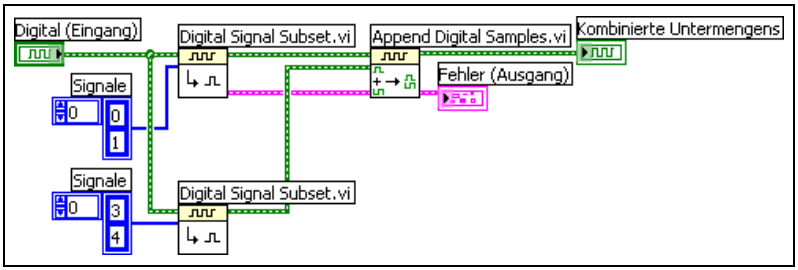


Abbildung 4-4. Anhängen eines digitalen Signalverlaufs an einen anderen

Mit dem oberen VI “Digital: Signalausschnitt” wird das erste und zweite Signal, und mit dem unteren das vierte und fünfte Signal entnommen. Mit dem VI “Digitale Abtastwerte anfügen” wird anschließend das erste Signal an das vierte und das zweite Signal an das fünfte angehängt. Die entstandenen Signale werden dann in einem Signalverlaufsgraphen für Digitalsignale dargestellt.

Anhängen digitaler Abtastungen und Signale

Mit dem VI “Digitale Signale anfügen” können Abtastungen aus einem Digitalsignal an das Ende eines anderen Signals angehängt werden. Die Anzahl der Abtastungen ist dabei unerheblich. Wenn beispielsweise zwei Signale vorliegen, die jeweils aus zwei Reihen von je 8 Bit bestehen, hat das Endsignal zwei Reihen von 16 Bit. Wenn das eine Signal aus zwei 8-Bit-Reihen und das andere aus einer 8-Bit-Reihe besteht, hat das Endsignal ebenfalls zwei 16-Bit-Reihen. Die übrigen Spalten im zweiten Signal werden dann mit dem Wert aufgefüllt, der am Eingang **Standardwert** angegeben wurde.

Kompression digitaler Daten

Mit dem VI “Digital: Komprimieren” können Digitaldaten so komprimiert werden, dass zwei oder mehr aufeinander folgende Signale mit denselben Bitfolgen in der gleichen Reihe angezeigt werden, um die Daten besser zu visualisieren. Wenn zum Beispiel zehn Abtastungen von Digitalsignalen vorliegen, von denen neun identisch sind, kann durch ein Komprimieren der Daten leichter herausgefunden werden, welcher der Signalverläufe sich

von den anderen unterscheidet. Durch das Komprimieren wird auch weniger Speicherplatz benötigt. Um die Daten wieder zu dekomprimieren, wird das VI “Digital: Dekomprimieren” verwendet.

Mustererkennung

Mit dem VI “Nach Digitalmustern suchen” kann ein bestimmtes digitales Muster erkannt werden. Wenn beispielsweise ein langer Signalverlauf vorliegt und Sie sehen möchten, ob ein Teil davon ein bestimmtes Muster aufweist, verbinden Sie das Muster mit dem Eingang **Digitales Muster** dieses VIs.

Referenzen auf Objekte oder Applikationen

Verwenden Sie die Referenznummer-Bedien- und Anzeigeelemente, die sich in den Paletten **RefNum** und **RefNum (klassisch)** befinden, um mit Dateien, Verzeichnissen, Geräten und Netzwerkverbindungen zu arbeiten. Mit dem RefNum-Bedienelement können Sie Informationen von Frontpanel-Objekten an SubVIs übergeben. Weitere Informationen zu RefNum-Bedienelementen finden Sie im Abschnitt *Steuern von Frontpanel-Objekten* des Kapitels 17, *Programmatische Steuerung von VIs*.

Eine Referenznummer (RefNum) ist ein eindeutiger Bezeichner für ein Objekt wie beispielsweise eine Datei, ein Gerät oder eine Netzwerkverbindung. Wenn Sie eine Datei, ein Gerät oder eine Netzwerkverbindung öffnen, erstellt LabVIEW eine RefNum, die mit dieser Datei, diesem Gerät oder dieser Netzwerkverbindung verknüpft ist. Bei allen Operationen, die Sie mit geöffneten Dateien, Geräten oder Netzwerkverbindungen ausführen, werden diese RefNums verwendet, um das jeweilige Objekt zu identifizieren. Mit RefNum-Bedien- oder -Anzeigeelementen können RefNums an ein VI übergeben oder von einem solchen empfangen werden. Beispielsweise können Sie ein RefNum-Bedien- oder Anzeigeelement verwenden, um den Inhalt der Datei zu ändern, auf welche die RefNum verweist, ohne die Datei schließen und erneut öffnen zu müssen.

Da eine RefNum ein temporärer Verweis auf ein geöffnetes Objekt ist, gilt sie nur so lange, bis es geschlossen wird. Wenn Sie das Objekt schließen, hebt LabVIEW die Verknüpfung zwischen RefNum und Objekt auf, und die RefNum wird freigegeben. Wenn Sie das Objekt erneut öffnen, erstellt LabVIEW eine neue RefNum, die jedoch nicht mit der ersten RefNum übereinstimmt.

LabVIEW speichert die mit jeder RefNum verbundenen Informationen, wie beispielsweise die aktuelle Position für das Auslesen oder das Beschreiben des Objekts sowie den Grad des Benutzerzugriffs, so dass Sie zwar gleichlaufende, jedoch unabhängige Operationen an einem einzigen Objekt durchführen können. Wenn ein Objekt in einem VI mehrfach geöffnet wird, gibt jede “Öffnen”-Operation eine andere RefNum zurück.

Dialogelemente

Dialogelemente können in selbst erstellten Dialogfeldern eingesetzt werden. Sie sind speziell für die Verwendung in Dialogfeldern konzipiert. Zu den Elementen gehören Ring-, Drehbedienelemente, numerische Schieber, Verlaufsanzeigen, Listenfelder, Tabellen, String- und Pfadbedienelemente, Registerkarten- und Baumstrukturelemente, Schaltflächen, Auswahl- und Optionsfelder sowie eine Beschriftung, die sich automatisch an die Hintergrundfarbe anpasst und das Darunterliegende verdeckt. Diese Bedienelemente unterscheiden sich von denjenigen auf dem Frontpanel lediglich im Hinblick auf die Darstellung. Die Bedienelemente werden in den Farben dargestellt, die Sie für den Desktop definiert haben.

Da sich das Erscheinungsbild der Dialogelemente abhängig von der Plattform ändert, auf der Sie das VI ausführen, ist die Darstellungsweise von Bedienelementen in von Ihnen erstellten VIs mit allen LabVIEW-Plattformen kompatibel. Wenn Sie ein VI auf einer anderen Plattform ausführen, passen sich die Dialogelemente in Farbe und Erscheinungsbild an die standardmäßigen Dialogfeld-Steuerelemente dieser Plattform an.

Um die Menüleiste und die Bildlaufleisten auszublenden sowie VIs zu erstellen, die wie die Standarddialogfelder der jeweiligen Plattform aussehen und sich ebenso verhalten, wählen Sie **Datei»VI-Einstellungen** und klicken Sie wählen Sie im Pulldown-Menü **Kategorie** auf die Option **Fenstererscheinungsbild**. Über die **Editor-Optionen** im Pulldown-Menü **Kategorie** lässt sich die Darstellungsart eines Bedien- oder Anzeigeelements festlegen, das von LabVIEW bei einem Rechtsklick auf einen VI-Anschluss und der Kontextmenü-Auswahl **Erstelle»Bedienelement** bzw. **Erstelle»Anzeigeelement** erzeugt wird. Um die Darstellungsart von Elementen in neuen VIs bei einem Rechtsklick auf einen Anschluss und die Option **Erstelle»Bedienelement** bzw. **Erstelle»Anzeigeelement** festzulegen, verwenden Sie **Werkzeuge»Optionen** und klicken dann im Pulldown-Menü auf **Frontpanel**. Weitere Informationen zum Konfigurieren des Aussehens und des Verhaltens von VIs finden Sie im Abschnitt *Verhalten und Erscheinungsbild von VIs konfigurieren* des Kapitels 16, *Anpassen von VIs*.

Beschriftungen

Verwenden Sie Beschriftungen, um Objekte auf dem Frontpanel und im Blockdiagramm zu kennzeichnen.

In LabVIEW gibt es zwei Arten von Beschriftungen – mit Objekten verknüpfte und freie Beschriftungen. Verknüpfte Beschriftungen gehören zu einem speziellen Objekt, werden mit diesem verschoben und kennzeichnen nur dieses Objekt. Sie können eine verknüpfte Beschriftung zwar unabhängig verschieben, wenn Sie jedoch das mit der Beschriftung verknüpfte Objekt verschieben, wird die Beschriftung zusammen mit dem Objekt verschoben. Verknüpfte Beschriftungen können ausgeblendet werden. Jedoch ist es nicht möglich, sie unabhängig vom zugehörigen Objekt zu kopieren oder zu löschen. Zur Verwendung von Einheitenbeschriftungen für numerische Elemente wählen Sie im Kontextmenü die Option **Sichtbare Objekte»Einheitenbeschriftung**. Weitere Informationen zu numerischen Einheiten finden Sie in Abschnitt *Numerische Einheiten und strikte Typenüberprüfung* des Kapitels 5, *Erstellen des Blockdiagramms*.

Freie Beschriftungen sind nicht objektgebunden. Sie können unabhängig erstellt, verschoben, gedreht oder gelöscht werden. Freie Beschriftungen bieten sich daher zum Beispiel für Anmerkungen in Frontpanels und Blockdiagrammen an.

Daneben eignen sie sich auch zur Dokumentation von Programmabschnitten im Blockdiagramm oder für Benutzeranweisungen auf dem Frontpanel. Zum Erstellen von freien Beschriftungen oder zum Bearbeiten von jeglicher Art von Beschriftung verwenden Sie das Beschriftungswerkzeug. Freie Beschriftungen können auch durch einen Doppelklick auf eine freie Fläche im Blockdiagramm oder Frontpanel erstellt werden.

Weitere Einzelheiten zum Erstellen von Beschriftungen für Beschreibungen auf der Benutzeroberfläche finden Sie auch in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*, unter *Labels*.

Untertitel

Zu Frontpanel-Objekten können auch Untertitel angezeigt werden. Zur Anzeige eines Untertitels klicken Sie mit der rechten Maustaste auf das Objekt und wählen aus dem Kontextmenü die Option **Sichtbare Objekte» Untertitel** aus. Im Gegensatz zu einer Beschriftung hat ein Untertitel keinen Einfluss auf den Namen des Objekts, und Sie können einen Untertitel für eher beschreibende Zwecke verwenden. Der Untertitel erscheint nicht im Blockdiagramm.

Wenn Sie das Objekt einem Anschluss eines Anschlussfeldes zuweisen, dann erscheint der Untertitel, wenn Sie das Verbindungswerkzeug über den Anschluss im Blockdiagramm bewegen. Der Untertitel erscheint auch an diesem Anschluss im **Kontext-Hilfe**-Fenster. Nähere Informationen zu Anschlussfeldern finden Sie in Abschnitt *Anschlussfeld einrichten* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Texteigenschaften

LabVIEW verwendet die Schriftarten, die bereits auf dem Computer installiert sind. Die Textattribute lassen sich über das Kontextmenü **Texteinstellungen** in der Symbolleiste festlegen. Wenn Sie Objekte oder Text auswählen, bevor Sie eine Auswahl aus dem Kontextmenü **Texteinstellungen** treffen, wirken sich die Änderungen auf alle ausgewählten Objekte aus. Wenn keine Auswahl getroffen wurde, werden die Änderungen an der Standardschriftart vorgenommen. Ein Ändern der Standardschriftart hat keine Auswirkung auf die Schriftart bestehender Beschriftungen. Es betrifft nur die Beschriftungen, die Sie ab diesem Zeitpunkt erstellen.

Um markiertem Text einen bestimmten Schriftstil zuzuweisen, wählen Sie auf dem Frontpanel im Pulldown-Menü für die **Texteinstellungen** die Option **Schriftsatzdialog** aus. Wenn Sie keinen Text markiert haben, ist die Option **Standard für Panel** aktiviert. Wenn Sie **Texteinstellungen» Schriftsatzdialog** vom Blockdiagramm aus wählen, ohne dass irgendwelche Objekte ausgewählt sind, ist die Option **Standard für Diagramm** aktiviert. Sie können für das Frontpanel und das Blockdiagramm jeweils verschiedene Schriftarten festlegen. Beispielsweise können Sie im Blockdiagramm eine kleine und auf dem Frontpanel eine große Schriftart verwenden.

Über das Pulldown-Menü **Texteinstellungen** sind folgende in LabVIEW integrierte Schriftarten verfügbar:

- **Applikationsschriftart**—Standardschriftart, die für die Paletten **Elemente** und **Funktionen** und für Text in neuen Bedienelementen verwendet wird.
- **Systemschriftart**—Wird in Menüs verwendet.
- **Dialogschriftart**—Wird für den Text in Dialogfeldern verwendet.

Wenn Sie ein VI, das eine dieser integrierten Schriftarten enthält, auf eine andere Plattform übertragen, werden Schriftarten verwendet, die der ursprünglichen so ähnlich wie möglich sind.

Das Pulldown-Menü **Texteinstellungen** verfügt auch über die Untermenüelemente **Anpassen**, **Stil**, **Ausrichten** und **Farbe**.

Die Auswahl, die Sie in irgendeinem dieser Untermenüs vornehmen, gilt für die jeweils ausgewählten Objekte. Wenn Sie beispielsweise eine neue Schriftart wählen, während ein Drehknopf oder ein Graph ausgewählt ist, werden alle Beschriftungen, Skalen und die numerischen Anzeigen in der neuen Schriftart formatiert.

Bei einer Änderung werden so viele Schriftattribute wie möglich bewahrt. Bei der Schriftart "Courier" werden Schriftgröße und die verwendeten Schriftstile nach Möglichkeit beibehalten. Wenn Sie das Dialogfeld **Schriftart** verwenden, ändert LabVIEW für die ausgewählten Objekte die ausgewählten Texteigenschaften. Bei Verwendung einer LabVIEW -Schriftart oder Auswahl der aktuellen Schriftart, ändert LabVIEW die Schriftart der ausgewählten Objekte und verwendet die entsprechende Größe.

Bei Objekten, die über mehrere Textbereiche verfügen, wie beispielsweise Schieberegler, betreffen die Schriftänderungen die aktuell ausgewählten Objekte oder den markierten Text. Wenn Sie beispielsweise den gesamten Schieberegler markieren und **Stil»Fett** aus dem Kontextmenü **Texteinstellungen** wählen, wird die Schrift der Skala, der numerischen Anzeige und der Beschriftung fett gesetzt. Bei Markierung der Beschriftung und Auswahl der Option **Fett** wird nur der Text der Beschriftung fett gesetzt. Wenn Sie den Text einer Achsenmarkierung markieren und **Fett** wählen, wird der Text aller Markierungen fett formatiert.

Weitere Informationen zur Gestaltung der Benutzeroberfläche mit Hilfe verschiedener Schriftarten und -attribute finden Sie in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*, unter *Fonts and Text Characteristics*.

Gestalten von Benutzeroberflächen

Erscheinungsbild und Layout des Frontpanels sind immer dann von Bedeutung, wenn ein VI als Benutzeroberfläche oder Dialogfeld dient. Gestalten Sie das Frontpanel so, dass die Benutzer einfach erkennen können, welche Aktionen durchgeführt werden. Sie können Frontpanels erstellen, die ähnlich wie Messinstrumente oder andere Geräte aussehen.

Weitere Informationen über die Gestaltung von Benutzeroberflächen finden Sie unter *LabVIEW Development Guidelines*.

Informationen zum Verwenden von Ereignissen zur Verbesserung der Funktionalität von Benutzeroberflächen finden Sie unter *Case- und Sequenzstrukturen* im Kapitel 8, *Schleifen und Strukturen*.

Verwenden von Frontpanel-Bedien- und Anzeigeelementen

Bedien- und Anzeigeelemente sind die Hauptkomponenten des Frontpanels. Bei der Gestaltung des Frontpanels sollten Sie berücksichtigen, wie die Benutzer mit den VIs arbeiten, und Bedien- und Anzeigeelemente logisch gruppieren. Wenn mehrere Bedienelemente zusammengehören, versehen Sie diese mit einem dekorativen Rahmen oder fassen Sie sie zu einem Cluster zusammen. Mit den Gestaltungselementen der Palette **Gestaltung** können Sie verschiedene Frontpanel-Objekte mit Hilfe von Feldern, Linien und Pfeilen gruppieren oder separieren. Diese Objekte dienen lediglich zur Gestaltung und zeigen keine Daten an.

Sorgen Sie für ausreichend Zwischenraum zwischen den Frontpanel-Objekten, um das Frontpanel übersichtlich zu gestalten. Darüber hinaus wird durch Leerräume verhindert, dass der Benutzer versehentlich auf das falsche Bedienelement oder die falsche Schaltfläche klickt.

Weisen Sie Schaltflächen bestimmte Namen zu und verwenden Sie die übliche Terminologie. Nutzen Sie anstelle von **OK** Beschriftungen wie **Start**, **Stopp** oder **Speichern**. Die Verwendung aussagekräftiger Beschriftungen erleichtert den Benutzern die Arbeit mit dem VI.

Setzen Sie die voreingestellten LabVIEW-Schriftarten und -Farben ein. Beim Portieren auf andere Plattformen ersetzt LabVIEW die integrierten Schriftarten durch vergleichbare Schriftartenfamilien. Wenn Sie eine andere Schriftart verwenden, ersetzt LabVIEW die Schrift durch eine möglichst ähnliche Schrift, wenn diese Schriftart auf dem Computer nicht vorhanden ist. LabVIEW behandelt Farben ähnlich wie Schriftarten. Wenn eine Farbe auf einem Computer nicht verfügbar ist, ersetzt LabVIEW diese durch eine möglichst ähnliche Farbe. Mit Hilfe der Systemfarben kann das Erscheinungsbild des Frontpanels an die Systemfarben des Computers, auf dem das VI läuft, angepasst werden.

Vermeiden Sie es, Objekte übereinander zu platzieren. Wenn Sie eine Beschriftung oder ein anderes Objekt so platzieren, dass es Bedien- oder Anzeigeelemente ganz oder teilweise überdeckt, wird die Bildschirmaktualisierung verlangsamt, was dazu führen kann, dass das Bedien- oder Anzeigeelement flimmert.

Zusätzliche Hinweise zur Verwendung von Layouts, Schriftarten und Farben zur Gestaltung der Benutzeroberfläche finden Sie in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*.

Gestalten von Dialogfeldern

Wenn ein VI aufeinander folgende Dialogfelder enthält, die an der gleichen Bildschirmposition angezeigt werden, ordnen Sie diese so an, dass die Schaltflächen im ersten Dialogfeld nicht direkt über den Schaltflächen im nächsten Dialogfeld liegen. Der Benutzer klickt möglicherweise eine Schaltfläche im ersten Dialogfeld doppelt an und klickt damit unwissentlich auf eine Schaltfläche im darauf folgenden Dialogfeld.

Weitere Informationen zu Dialogelementen finden Sie in Abschnitt *Dialogelemente* dieses Kapitels. Hinweise zur Verwendung von Dialogfeldern in der Benutzeroberfläche entnehmen Sie bitte dem Abschnitt *Colors* des Kapitels 6, *LabVIEW Style Guide*, der *LabVIEW Development Guidelines*.

Auswahl der Bildschirmgröße

Berücksichtigen Sie beim Entwurf eines VIs, dass das Frontpanel möglicherweise auch auf Computern mit unterschiedlichen Bildschirmauflösungen angezeigt wird. Um die Fensterproportionen des Frontpanels in Relation zur Bildschirmauflösung beizubehalten, wählen Sie **Datei» VI-Einstellungen**, klicken Sie im Pulldown-Menü **Kategorie** auf **Fenstergröße** und aktivieren Sie dann das Kontrollkästchen **Fensterproportionen für verschiedene Bildschirmauflösungen beibehalten**.

Weitere Einzelheiten zur Auswahl der Bildschirmgröße für eine Benutzeroberfläche finden Sie auch in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*, im Abschnitt *Sizing and Positioning*.

5

Erstellen des Blockdiagramms

Nachdem Sie das Frontpanel erstellt haben, können Sie mit Hilfe grafisch dargestellter Funktionen Code hinzufügen, um die Frontpanel-Objekte zu steuern. Dieser grafische Quellcode ist im Blockdiagramm enthalten.

Weitere Informationen …

Weitere Informationen zur Gestaltung und Konfiguration des Blockdiagramms befinden sich in der *LabVIEW-Hilfe*.

Beziehung zwischen Frontpanel-Objekten und Blockdiagrammanschlüssen

Frontpanel-Objekte werden im Blockdiagramm als Anschlüsse dargestellt. Wenn Sie wissen möchten, für welches Frontpanel-Element ein bestimmtes Blockdiagrammobjekt steht, klicken Sie dieses doppelt an. Das entsprechende Element wird dann im Frontpanel hervorgehoben.

Anschlüsse sind Eingangs- und Ausgangsports, über die Informationen zwischen dem Frontpanel und dem Blockdiagramm ausgetauscht werden. Daten, die Sie in den Frontpanel-Bedienelementen eingeben, werden über die Bedienelementanschlüsse an das Blockdiagramm übergeben. Die Daten fließen bei der Ausführung eines VIs zu den Anzeigeelementanschlüssen, wo sie das Blockdiagramm verlassen, um erneut an das Frontpanel übergeben und dann mit Hilfe der Frontpanel-Anzeigeelemente angezeigt zu werden.

Blockdiagrammobjekte

Zu den Objekten im Blockdiagramm gehören Anschlüsse, Knoten und Funktionen. Blockdiagramme werden erstellt, indem Objekte mit Hilfe von Verbindungen verknüpft werden.

Blockdiagrammanschlüsse

Die Frontpanel-Elemente können so konfiguriert werden, dass sie im Blockdiagramm entweder als Symbol oder als Anschluss eines bestimmten Datentyps dargestellt werden. Per Voreinstellung werden Frontpanel-Objekte als Symbole dargestellt. So wird zum Beispiel, wie links dargestellt, mit einem Drehknopfsymbol ein auf dem Frontpanel befindlicher Drehknopf dargestellt. Das “DBL” am unteren Rand zeigt den Datentyp an. Das heißt, dieses Element arbeitet mit Fließkommazahlen mit doppelter Genauigkeit. Dagegen ist bei Darstellung als DBL-Symbol, wie links abgebildet, nur ersichtlich, dass es sich um ein numerisches Bedien- oder Anzeigeelement handelt, das mit diesem Datentyp arbeitet. Damit als VI-Symbol nur der Datentyp zu sehen ist, klicken Sie dieses an und wählen Sie aus dem Kontextmenü die Option **Als Symbol anzeigen**. Die Anschlüsse mit den VI-Symbolen haben den Vorteil, dass neben dem Datentyp im Blockdiagramm auch ersichtlich wird, um was für ein Frontpanel-Objekt es sich handelt. Die Anschlüsse, die nur den Datentyp anzeigen, sind dagegen platzsparender.

Hinweis Wenn Datentypsymbole in Symbole umgewandelt werden, die die entsprechenden Elemente darstellen, sollte beachtet werden, dass diese größer sind und dadurch andere Blockdiagrammobjekte verdecken können.

Ein Anschluss ist ein Punkt, an den eine Verbindungsleitung angeschlossen werden kann, die allerdings nicht an einer anderen Verbindung enden darf. LabVIEW verfügt über Anschlüsse für Bedien- und Anzeigeelemente, Knotenanschlüsse, Konstanten und spezielle Anschlüsse an Strukturen, wie beispielsweise Eingabe- und Ausgabeanschlüsse am Formelknoten. Zum Herstellen der Verbindungen zu den Anschlüssen und zur Weitergabe von Daten an andere Anschlüsse verwenden Sie Verbindungen. Klicken Sie mit der rechten Maustaste auf ein Blockdiagrammobjekt und wählen Sie aus dem Kontextmenü **Sichtbare Elemente»Anschlüsse**, um die Anschlüsse anzuzeigen. Klicken Sie mit der rechten Maustaste auf ein Objekt, und wählen Sie erneut **Sichtbare Elemente»Anschlüsse**, um die Anschlüsse wieder auszublenden. Dieses Kontextmenü steht nicht für erweiterbare VIs und Funktionen zur Verfügung

Datentypen für Bedien- und Anzeigeelemente

Tabelle 5-1 enthält die Symbole für die unterschiedlichen Anschlusstypen bei Bedien- und Anzeigeelementen. Die Farbe und das Symbol des jeweiligen Anschlusses repräsentieren den Datentyp des Bedien- bzw. Anzeigeelements. Bedienelementanschlüsse weisen einen dickeren Rahmen als Anzeigeelementanschlüsse auf. Außerdem wird bei den

Anschlüssen der Frontpanel-Elemente durch schwarze Pfeile angezeigt, ob es sich um ein Bedien- oder Anzeigeelement handelt. Befindet sich der Pfeil auf der rechten Seite des Anschlusses, handelt es sich um ein Bedien-, ansonsten um ein Anzeigeelement.

Tabelle 5-1. Anschlüsse von Bedien- und Anzeigeelementen

Bedien-element	Anzeige-element	Datentyp	Farbe	Standard werte
SGL	SGL	Fließkommazahl mit einfacher Genauigkeit	orange	0.0
DBL	DBL	Fließkommazahl mit doppelter Genauigkeit	orange	0.0
EXT	EXT	Fließkommazahl mit erweiterter Genauigkeit	orange	0.0
CSG	CSG	komplexe Fließkommazahl mit einfacher Genauigkeit	orange	0,0 + i0,0
CDB	CDB	komplexe Fließkommazahl mit doppelter Genauigkeit	orange	0,0 + i0,0
CXT	CXT	komplexe Fließkommazahl mit erweiterter Genauigkeit	orange	0,0 + i0,0
I8	I8	8-Bit-Ganzzahl mit Vorzeichen	blau	0
I16	I16	16-Bit-Ganzzahl mit Vorzeichen	blau	0
I32	I32	32-Bit-Ganzzahl mit Vorzeichen	blau	0
U8	U8	8-Bit-Ganzzahl ohne Vorzeichen	blau	0
U16	U16	16-Bit-Ganzzahl ohne Vorzeichen	blau	0
U32	U32	32-Bit-Ganzzahl ohne Vorzeichen	blau	0
		<64,64>-Bit Zeitstempel	braun	Zeit und Datum (lokal)
		Enum-Typ	blau	—
TF	TF	boolesch	grün	FALSE
abc	abc	String	rosa	leerer String
[ ]	[ ]	Array—der Datentyp der Elemente wird in eckige Klammern eingeschlossen, und die Farbe des Datentyps wird übernommen.	unter-schiedlich	—

Tabelle 5-1. Anschlüsse von Bedien- und Anzeigeelementen (Fortsetzung)

Bedien-element	Anzeige-element	Datentyp	Farbe	Standard werte
		Cluster—umfasst mehrere Datentypen. Wenn die Elemente des Clusters vom gleichen Typ sind, werden Cluster-Datentypen braun dargestellt, ansonsten rosa.	braun oder rosa	—
		Pfad	hellblau	`<Kein Pfad>`
		dynamisch	blau	—
		Signalverlauf—ein Cluster mit Elementen, die Daten, Startzeit und Δt eines Signalverlaufs enthalten. Weitere Informationen zum Datentyp “Signalverlauf” befinden sich in Abschnitt *Datentyp “Signalverlauf”* des Kapitels 12, *Graphen und Diagramme*.	braun	—
		Digitaler Signalverlauf	dunkel grün	—
		Digitaldaten	dunkel grün	—
		Referenznummer (RefNum)	hellblau	—
		Variant—speichert den Namen des Bedien- oder Anzeigeelements, Angaben zum Datentyp und die Daten selbst. Weitere Informationen zu Daten vom Typ “Variant” befinden sich in Abschnitt *Verarbeiten von Variant-Daten* dieses Kapitels.	lila	—

Tabelle 5-1. Anschlüsse von Bedien- und Anzeigeelementen (Fortsetzung)

Bedien-element	Anzeige-element	Datentyp	Farbe	Standard werte
		I/O-Name—übergibt benutzerseitig konfigurierte Datenerfassungs-, VISA-Ressourcen- und logische IVI-Namen an I/O-VIs, damit diese mit einem Instrument oder einem DAQ-Gerät kommunizieren können. Weitere Informationen zum Datentyp “I/O-Name” befinden sich in Abschnitt *I/O-Namen-Bedien- und Anzeigeelemente* des Kapitels 4, *Erstellen des Frontpanels*.	lila	—
		Bild—dient der Anzeige von Bildern, die Linien, Kreise, Text und andere Arten von Grafikformen enthalten. Weitere Informationen zum Datentyp “Bild” befinden sich in Abschnitt *Verwenden des Bildanzeigeelements* des Kapitels 13, *VIs für Grafiken und Sound*.	blau	—

Zu einigen Datentypen gibt es Funktionssätze, mit denen speziell Daten dieser Typen bearbeitet werden können. Weitere Informationen zu den Funktionen, die mit dem jeweiligen Datentyp eingesetzt werden können, befinden sich in Abschnitt *Überblick über Funktionen* dieses Kapitels.

Weitere Einzelheiten dazu, wie durch Auswahl des richtigen Datentyps die Speichernutzung optimiert werden kann, finden Sie auch in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, im Abschnitt *Memory and Speed Optimization*.

Konstanten

Konstanten sind Anschlüsse im Blockdiagramm, mit denen konstante Datenwerte an das Blockdiagramm übergeben werden, wie beispielsweise Pi (π) oder Unendlich (∞). Benutzerdefinierte Konstanten sind Konstanten, die Sie vor der VI-Ausführung selbst festlegen und bearbeiten.

Sie beschriften eine Konstante, indem Sie mit der rechten Maustaste auf die Konstante klicken und dann aus dem Kontextmenü **Sichtbare Elemente» Beschriftung** wählen. Konstanten enthalten Standardbeschriftungen, die Sie mit Hilfe des Bedienwerkzeugs oder des Beschriftungswerkzeugs bearbeiten können.

Die meisten Konstanten befinden sich im oberen oder unteren Bereich der jeweiligen Palette.

Konstanten

Konstanten werden für mathematische Berechnungen und zum Formatieren von Strings oder Pfaden eingesetzt. LabVIEW umfasst die folgenden Arten von Konstanten:

- **Zusätzliche numerische Konstanten**—Häufig verwendete mathematische und physikalische Werte mit hoher Genauigkeit, beispielsweise der natürliche Logarithmus der Basis (e) oder die Lichtgeschwindigkeit. Die zusätzlichen numerischen Konstanten befinden sich auf der Palette **Zusätzliche numerische Konstanten**.
- **Stringkonstanten**—Häufig verwendete, nicht darstellbare Stringzeichen wie “Zeilenvorschub” (Line Feed, LF) oder “Zeilenumbruch” (Carriage Return, CR). Die Stringkonstanten befinden sich auf der Palette **String**.
- **Dateikonstanten**—Häufig verwendete Pfadwerte wie beispielsweise “Kein Pfad”, “Keine RefNum” und “Standardverzeichnis”. Die Dateikonstanten befinden sich auf der Palette **Dateikonstanten**.

Benutzerdefinierte Konstanten

Die Palette **Funktionen** enthält nach Typ geordnete Konstanten wie beispielsweise “Boolesch”, “Zahl”, “Ring”, “Enum”, “Farbfeld”, “Listenfeld”, “String”, “Array”, “Cluster” und “Pfad”.

Um eine benutzerdefinierte Konstante zu erstellen, ist mit der rechten Maustaste auf den jeweiligen Anschluss einer Funktion zu klicken und **Erstelle Konstante** aus dem Kontextmenü auszuwählen. Sie können den Wert von benutzerdefinierten Konstanten nicht ändern, während das VI ausgeführt wird.

Sie können eine Konstante auch erstellen, indem Sie ein Bedienelement des Frontpanels auf das Blockdiagramm ziehen. Die erstellte Konstante enthält den Wert des Frontpanel-Bedienelements zu dem Zeitpunkt, zu dem Sie es auf das Blockdiagramm gezogen haben. Das Frontpanel-Bedienelement verbleibt auf dem Frontpanel. Änderungen am Inhalt des Bedienelements haben keinen Einfluss auf die Konstante und umgekehrt.

Um den Wert der Konstanten zu ändern, ist das Bedien- oder das Beschriftungswerkzeug zu wählen. Ist die automatische Werkzeugauswahl aktiv, genügt es, einen Doppelklick auf die Konstante auszuführen, um diese bearbeiten zu können.

Blockdiagrammknoten

Knoten sind Objekte im Blockdiagramm, die über Ein- und/oder Ausgänge verfügen und in einem laufenden VI bestimmte Funktionen ausführen. Knoten entsprechen Anweisungen, Operatoren, Funktionen und Unterprogrammen in textbasierten Programmiersprachen. LabVIEW umfasst die folgenden Arten von Knoten:

- **Funktionen**—Integrierte- Ausführungselemente, die mit einem Operator, einer Funktion oder einer Anweisung vergleichbar sind. Weitere Informationen zu den in LabVIEW verfügbaren Funktionen finden Sie im Abschnitt *Überblick über Funktionen* dieses Kapitels.
- **SubVIs**—VIs, die in einem Blockdiagramm von einem anderen VI verwendet werden, vergleichbar mit Unterprogrammen. Der Einsatz von SubVIs im Blockdiagramm ist im Abschnitt *SubVIs* des Kapitels 7, *Erstellen von VIs und SubVIs*, beschrieben.
- **Strukturen**—Prozesssteuerungselemente wie beispielsweise Sequenzstrukturen, Case-Strukturen, For- oder While-Schleifen. Lesen Sie zu diesem Thema auch Kapitel 8, *Schleifen und Strukturen*.
- **Formelknoten**—In der Größe veränderbare Strukturen, mit denen Gleichungen direkt in ein Blockdiagramm eingegeben werden können. Die Verwendung von Formelknoten ist im Abschnitt *Formelknoten* des Kapitels 21, *Formeln und Gleichungen*, erläutert.
- **Ausdrucksknoten**—Strukturen zur Berechnung von Ausdrücken oder Gleichungen mit einer einzelnen Variablen (siehe auch Abschnitt *Ausdrucksknoten* des Kapitels 21, *Formeln und Gleichungen*).
- **Eigenschaftsknoten**—Strukturen, mit denen die Eigenschaften einer Klasse festgelegt oder ermittelt werden können. Nähere Einzelheiten zum Einsatz von Eigenschaftsknoten finden Sie im Abschnitt *Eigenschaftsknoten* des Kapitels 17, *Programmatische Steuerung von VIs*.
- **Methodenknoten**—Struktur zur Ausführung von Methoden einer bestimmten Klasse. Lesen Sie hierzu auch den Abschnitt *Methodenknoten* des Kapitels 17, *Programmatische Steuerung von VIs*.
- **Code-Interface-Knoten (CINs)**—Strukturen zum Aufrufen von Programmcode in einer textbasierten Programmiersprache. Wie Code aus textbasierten Programmiersprachen aufgerufen werden kann, erfahren Sie im Abschnitt *Code-Interface-Knoten* des Kapitels 20, *Aufrufen von Code aus textbasierten Programmiersprachen*.

- **Knoten zum Aufruf über Referenz**—Strukturen zum Aufrufen eines VIs, das dynamisch geladen wird. Weitere Hinweise zum Einsatz des Knotens zum Aufruf über Referenz befinden sich in Abschnitt *Der Knoten "Aufruf über Referenz" und strikt typisierte VI-RefNums* des Kapitels 17, *Programmatische Steuerung von VIs*.
- **Knoten zum Aufruf externer Bibliotheken**—Strukturen, mit denen die meisten Standardbibliotheken (DLLs) aufgerufen werden können. Lesen Sie zu diesem Thema auch den Abschnitt *Knoten zum Aufruf externer Bibliotheken* des Kapitels 20, *Aufrufen von Code aus textbasierten Programmiersprachen*.

Überblick über Funktionen

Funktionen sind die grundlegenden Betriebselemente von LabVIEW. Die Elemente der **Funktionen**-Palette, die mit einem hellgelben Hintergrund und einem schwarzen Vordergrund dargestellt werden, sind die Symbole der Basisfunktionen. Funktionen verfügen nicht über Frontpanels oder Blockdiagramme, weisen jedoch Anschlussfelder auf. Sie können weder geöffnet noch bearbeitet werden.

Die Palette **Funktionen** beinhaltet auch die VIs, die zum Lieferumfang von LabVIEW gehören. Verwenden Sie diese VIs als SubVIs, wenn Sie VIs für die Datenerfassung, die Instrumentensteuerung, die Kommunikation oder andere VIs erstellen. Weitere Informationen zum Einsatz der integrierten VIs befinden sich im Abschnitt *Verwendung der in LabVIEW integrierten VIs und Funktionen* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Numerische Funktionen

Mit den numerischen Funktionen können arithmetische, trigonometrische, logarithmische und komplexe mathematische Operationen durchgeführt und Zahlen in andere Datentypen konvertiert werden.

Boolesche Funktionen

Mit Hilfe der booleschen Funktionen können logische Operationen für einzelne boolesche Werte oder Arrays mit booleschen Werten durchgeführt werden:

- Ändern eines Wertes von TRUE zu FALSE und umgekehrt.
- Festlegen, welcher boolesche Wert zurückgegeben werden soll, wenn Sie zwei oder mehr boolesche Werte erhalten.

- Umwandeln eines booleschen Wertes in eine Zahl (entweder 1 oder 0).
- Ausführen einer mehrfacharithmetischen Funktion für zwei oder mehr boolesche Werte.

String-Funktionen

Mit den String-Funktionen sind folgende Operationen möglich:

- Verknüpfen von zwei oder mehreren Strings.
- Extrahieren eines Teilstrings aus einem String.
- Suchen und ersetzen von Zeichen oder Teilstrings in einem String.
- Umwandeln numerischer Daten in Strings.
- Formatieren eines Strings zur Verwendung in einem Textverarbeitungs- oder Tabellenkalkulationsprogramm.

Nähere Informationen zum Einsatz der String-Funktionen befinden sich in Abschnitt *Strings* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Array-Funktionen

Mit den Array-Funktionen können Arrays erstellt und verändert werden, zum Beispiel durch folgende Operationen:

- Extrahieren von einzelnen Datenelementen aus einem Array.
- Hinzufügen von einzelnen Datenelementen zu einem Array.
- Teilen von Arrays.

Weitere Informationen zum Einsatz der Array-Funktionen befinden sich in Abschnitt *Arrays* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Cluster-Funktionen

Mit den Cluster-Funktionen können Cluster erstellt und verändert werden. So sind zum Beispiel folgende Operationen möglich:

- Extrahieren von einzelnen Datenelementen aus einem Cluster.
- Hinzufügen von einzelnen Datenelementen zu einem Cluster.
- Aufteilen eines Clusters in die einzelnen Datenelemente.

Weitere Informationen zum Einsatz der Cluster-Funktionen befinden sich in Abschnitt *Cluster* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Vergleichsfunktionen

Die Vergleichsfunktionen eignen sich sowohl zum Vergleichen von booleschen Werten als auch von Strings, numerischen Werten, Arrays und Clustern.

Weitere Informationen zum Einsatz der Vergleichsfunktionen finden Sie in Anhang C, *Vergleichsfunktionen*.

Zeit- und Dialogfunktionen

Mit den Zeit- und Dialogfunktionen sind folgende Operationen möglich:

- Manipulieren der Geschwindigkeit, mit der eine Operation ausgeführt wird.
- Abrufen von Uhrzeit und Datum der systemeigenen Uhr.
- Erstellen von Dialogfeldern für Anweisungen an den Benutzer.

Die Palette **Zeit & Dialog** enthält auch VIs zur Fehlersuche, zu denen Sie weitere Informationen im Abschnitt *Fehlerprüfung und Fehlerbehandlung* des Kapitels 6, *Ausführen von VIs und Fehlersuche*, finden.

Datei-I/O-Funktionen

Mit den Datei-I/O-Funktionen sind folgende Operationen möglich:

- Öffnen und Schließen von Dateien.
- Lesen aus und Schreiben in Dateien.
- Erstellen der Verzeichnisse und Dateien, die Sie im Pfad-Bedienelement angeben.
- Abrufen von Verzeichnisinformationen.
- Schreiben von Strings, Zahlen, Arrays und Clustern in Dateien.

Die Palette **Datei-I/O** enthält auch VIs zum Ausführen von häufig anstehenden Datei-I/O-Aufgaben. Weitere Hinweise zur Verwendung von Datei-I/O-VIs und -Funktionen befinden sich in Kapitel 14, *Datei-I/O*.

Signalverlaufsfunktionen

Mit den Signalverlaufsfunktionen sind folgende Operationen möglich:

- Erstellen von Signalverläufen, die Datenwerte sowie Informationen über Kanäle und Timing enthalten.
- Extrahieren von einzelnen Datenelementen aus einem Signalverlauf.
- Bearbeiten von einzelnen Datenelementen aus einem Signalverlauf.

Nähere Einzelheiten zum Erstellen und Verwenden von Signalverläufen in VIs befinden sich in Kapitel 5, *Creating a Typical Measurement Application*, des *LabVIEW Measurements Manuals*.

Applikationsteuerungsfunktionen

Die VIs und Funktionen zur Applikationssteuerung dienen zur Steuerung der VIs und LabVIEW-Applikationen auf Ihrem lokalen Computer oder über ein Netzwerk. Weitere Hinweise zum Einsatz der Funktionen zur Applikationssteuerung befinden sich in Kapitel 17, *Programmatische Steuerung von VIs*.

Erweiterte Funktionen

Mit den erweiterten Funktionen können Programmcode aus Bibliotheken wie DLLs (Dynamic Link Libraries) bzw. mit textbasierten Programmiersprachen erstellte Programmteile aufgerufen werden. Außerdem können damit LabVIEW-Daten zum Einsatz in anderen Applikationen sowie Windows-Registry-Einträge bearbeitet werden. Weitere Informationen zur Verwendung der erweiterten Funktionen befinden sich im Handbuch *Using External Code in LabVIEW*.

Hinzufügen von Anschlüssen zu Blockdiagrammfunktionen

Bei einigen Funktionen kann die Anzahl der Anschlüsse verändert werden. Wenn Sie beispielsweise ein Array mit zehn Elementen erstellen möchten, müssen Sie der Funktion “Array erstellen” zehn Anschlüsse hinzufügen.

Um erweiterbaren VIs Anschlüsse hinzuzufügen, ziehen Sie die Funktion mit dem Positionierwerkzeug nach oben oder unten auf. Mit dem Positionierwerkzeug lassen sich auch Anschlüsse wieder entfernen. Dazu dürfen diese jedoch nicht verbunden sein. In diesem Fall müssen Sie zuerst die vorhandene Verbindung löschen, bevor Sie den Anschluss entfernen können.

Sie können Anschlüsse auch hinzufügen oder entfernen, indem Sie mit der rechten Maustaste auf einen der Anschlüsse der Funktion klicken und dann aus dem Kontextmenü **Eingang hinzufügen**, **Ausgang hinzufügen**, **Eingang entfernen** oder **Ausgang entfernen** wählen. Abhängig von der Funktion können dann Anschlüsse für Eingänge, Ausgänge oder Ref-Num-Bedienelemente hinzugefügt werden. Mit den Menüelementen **Eingang hinzufügen** und **Ausgang hinzufügen** aus dem Kontextmenü wird ein Anschluss unmittelbar an den Anschluss angefügt, auf den Sie mit der rechten Maustaste geklickt haben. Mit den Menüelementen **Eingang entfernen** und **Ausgang entfernen** wird der Anschluss entfernt, auf den

Sie mit der rechten Maustaste geklickt haben. Wenn Sie über das Kontextmenü einen bereits verbundenen Anschluss entfernen, wird der Anschluss gelöscht und die Verbindung getrennt.

Verbindung von Blockdiagrammobjekten

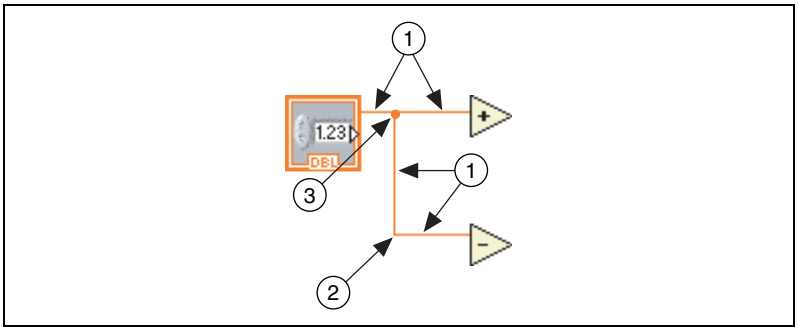
Die Datenübertragung zwischen den Blockdiagrammobjekten erfolgt über Verbindungsleitungen. Jede Verbindung besteht aus einer einzigen Datenquelle, die mit einer oder mehreren Datensenken, wie z. B. VIs oder Funktionen, verbunden werden kann. Hierbei müssen alle notwendigen Anschlüsse verbunden werden. Ansonsten ist das VI nicht ausführbar. Die Anschlüsse, die verbunden werden müssen, um die Funktionstüchtigkeit eines Knotens zu gewährleisten, können über die **Kontext-Hilfe** eingesehen werden. Die entsprechenden Beschriftungen sind darin fett gedruckt dargestellt. Weitere Hinweise zu den zu verbindenden Anschlüssen sind dem Abschnitt *Festlegen von erforderlichen, empfohlenen und optionalen Ein- und Ausgängen* des Kapitels 7, *Erstellen von VIs und SubVIs* zu entnehmen. Informationen über fehlerhafte VIs befinden sich in Abschnitt *Fehlerbeseitigung in nicht ausführbaren VIs* des Kapitels 6, *Ausführen von VIs und Fehlersuche*.

Je nach Datentyp haben die Verbindungen unterschiedliche Farben, Formate und Linienstärken. Eine unterbrochene Verbindung wird als gestrichelte schwarze Linie mit einem roten `X` in der Mitte dargestellt. Die Pfeile an beiden Seiten des roten `X` zeigen die Richtung des Datenstroms an und die Farbe der Pfeile den jeweiligen Datentyp. Weitere Informationen zu den Datentypen von Verbindungen befinden sich auf der *LabVIEW-Referenzkarte*.

Verbindungsstümpfe sind die abgeschnittenen Leitungen, die an einem unverbundenen VI- oder Funktionssymbol angezeigt werden, wenn Sie das Verbindungswerkzeug über das Symbol bewegen. Sie zeigen den Datentyp des jeweiligen Anschlusses an. Darüber hinaus erscheint auch ein Hinweisstreifen mit dem Namen des Anschlusses. Ist der Anschluss bereits verbunden, erscheint der Verbindungsstumpf für diesen Anschluss nicht mehr, wenn Sie das Verbindungswerkzeug über die Funktion bewegen.

Ein Verbindungssegment ist ein einziges, horizontal oder vertikal verlegtes Verbindungsstück. An der Stelle, an der zwei Segmente verbunden sind, entsteht eine Abzweigung. Der Punkt, an dem zwei oder mehr Verbindungssegmente zusammenlaufen, wird als Knotenpunkt bezeichnet. Ein Verbindungszweig enthält alle Verbindungssegmente von Knotenpunkt zu Knotenpunkt, von Anschluss zu Knotenpunkt oder von Anschluss zu Anschluss, wenn sich keine Knotenpunkte dazwischen befinden.

Abbildung 5-1 zeigt ein Verbindungssegment, einen Bogen und einen Knotenpunkt.



1	Segment	2	Bogen	3	Knotenpunkt

Abbildung 5-1. Verbindungssegment, Bogen und Knotenpunkt

Beim Verbinden eines Anschlusses fügen Sie einmal einen 90°-Bogen ein, indem Sie den Cursor entweder vertikal oder horizontal bewegen. Um eine Verbindung um 90° abzuwinkeln, ist an der entsprechenden Stelle mit der Maustaste zu klicken und der Mauszeiger in die neue Richtung zu ziehen. Auf diese Weise kann ein Verbindungsstrang beliebig stark verwinkelt werden.

Um eine Verbindung von dem letzten Punkt, an dem sie fixiert wurde, wieder zu lösen, ist die <Umschalt>-Taste zu drücken und an einer beliebigen Stelle ins Blockdiagramm zu klicken.

(Mac OS) Drücken Sie beim Klicken die <Befehlstaste>.
(UNIX und Linux) Klicken Sie mit der mittleren Maustaste.

Wenn Sie Verbindungen kreuzweise übereinander legen, zeigt ein kleiner Spalt in der ersten von Ihnen gezogenen Verbindung an, dass die erste Verbindung unter der zweiten liegt.

Vorsicht! Sich kreuzende Verbindungen können ein Blockdiagramm unübersichtlich werden lassen und dadurch die Fehlersuche erschweren.

Verdrahtungstechniken und weitere Tipps zu diesem Thema befinden sich auch in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter *Wiring Techniques*.

Automatisches Verbinden von Objekten

Objekte können in LabVIEW beim Ablegen in das Blockdiagramm automatisch miteinander verbunden werden. Dasselbe gilt für bereits auf dem Blockdiagramm befindliche Objekte. LabVIEW verbindet die Anschlüsse, die am besten zusammenpassen, und lässt Anschlüsse, die nicht zusammenpassen, unverbunden.

Wenn Sie ein markiertes Objekt nahe an andere Objekte im Blockdiagramm verschieben, zieht LabVIEW temporäre Verbindungen, um gültige Verbindungen anzuzeigen. Wenn Sie die Maustaste loslassen, um das Objekt auf dem Blockdiagramm zu platzieren, stellt LabVIEW automatisch die Verbindungen her.

Um die Funktion “Automatische Verbindung” zu deaktivieren, halten Sie die Leertaste gedrückt, während Sie ein Objekt mit Hilfe des Positionierwerkzeugs bewegen.

Die Funktion “Automatische Verbindung” ist standardmäßig aktiviert, wenn Sie ein Objekt aus der Palette **Funktionen** wählen oder wenn Sie ein Objekt kopieren, das sich bereits im Blockdiagramm befindet, indem Sie die <Strg>-Taste drücken und das Objekt ziehen. Die automatische Verbindungsfunktion ist standardmäßig deaktiviert, wenn Sie das Positionierwerkzeug verwenden, um ein Objekt zu verschieben, das sich bereits im Blockdiagramm befindet.

(Mac OS) Drücken Sie <Befehlstaste>. **(Sun)** Drücken Sie die <Meta>-Taste. **(Linux)** Drücken Sie die <Alt>-Taste.

Zum Deaktivieren der Funktion zum automatischen Verbinden wählen Sie **Werkzeuge»Optionen** und klicken anschließend im Pulldown-Menü auf die Option **Blockdiagramm**.

Manuelles Verbinden von Objekten

Verwenden Sie das Verbindungswerkzeug, um Anschlüsse an einem Blockdiagrammknoten mit den Anschlüssen an einem anderen Blockdiagrammknoten zu verbinden. Die Cursorspitze des Werkzeugs ist die Spitze der abgewickelten Leitungsspule. Wenn Sie das Verbindungswerkzeug über einen Anschluss bewegen, blinkt dieser. Bei Anschlüssen von VIs oder Funktionen erscheint daneben auch ein Hinweisstreifen mit dem Namen des Anschlusses. Wenn Sie zwei inkompatible Anschlüsse miteinander verbinden wollen, erscheint am Verbindungswerkzeug ein Warnzeichen. Die fehlerhafte Verbindung kann zwar dennoch erstellt, muss jedoch vor Ausführung des VIs entfernt werden, damit es funktions-

tüchtig ist. Wie sich Verbindungsfehler beseitigen lassen, ist im Abschnitt *Beseitigen von Verbindungsfehlern* dieses Kapitels näher erläutert.

Um beim Verbinden eng beieinander liegende Anschlüsse voneinander zu unterscheiden, ist die **Kontext-Hilfe** zu verwenden. Wenn der Cursor über ein VI oder eine Funktion bewegt wird, werden in der **Kontext-Hilfe** alle dazugehörigen Anschlüsse angezeigt. Bei erweiterbaren Funktionen, wie beispielsweise “Array erstellen”, werden jedoch nicht alle Anschlüsse angezeigt. Damit auch die optional zu verbindenden Anschlüsse im Anschlussfeld angezeigt werden, klicken Sie in der **Kontext-Hilfe** auf die Schaltfläche **Optionale Anschlüsse und kompletten Pfad anzeigen**. Weitere Informationen zu diesen Anschlüssen finden Sie in Abschnitt *Festlegen von erforderlichen, empfohlenen und optionalen Ein- und Ausgängen* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Erstellen von Verbindungen

Beim Verdrahten sucht LabVIEW automatisch nach einem günstigen Verbindungsweg. So werden Verbindungen zum Beispiel um vorhandene Objekte wie Schleifen oder Strukturen herumgelegt. Außerdem werden alle Verbindungen soweit wie möglich begradigt, also optimiert. Dabei wird nach Möglichkeit so verfahren, dass alle Verbindungen aus Bedienelementen an der rechten Seite des Symbols austreten und bei Anzeigeelementen an der linken Seite eintreten.

Um einen vorhandenen Verdrahtungsweg automatisch zu optimieren, klicken Sie diesen mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Verbindungen ausrichten** aus.

Wenn Sie, nachdem Sie mit dem Verbinden von VIs begonnen haben, das Autorouting kurzfristig ausschalten und die Verbindung manuell erstellen möchten, drücken Sie die Taste <A>. Durch erneutes Drücken der <A>-Taste wird das Autorouting wieder aktiviert. Beim Anschließen der Verbindung wird das Autorouting automatisch wieder aktiviert. Eine weitere Möglichkeit, das eingeschaltete Autorouting zu deaktivieren, besteht darin, einfach die gedrückte Maustaste wieder loszulassen. Nach dem Loslassen der Maustaste wird das Autorouting automatisch erneut aktiviert.

Zum Deaktivieren der Funktion zum Autorouting für alle neuen Verbindungen wählen Sie **Werkzeuge»Optionen** und klicken anschließend im Pulldown-Menü auf die Option **Blockdiagramm**.

Wenn das Autorouting deaktiviert ist, können die Verbindungen vertikal oder horizontal an die Anschlüsse herangeführt werden, je nachdem, in welche Richtung das Verbindungswerkzeug zuerst bewegt wird. Die Ver-

bindung wird zum Mittelpunkt des Anschlusses hergestellt, ungeachtet, an welche Stelle des Anschlusses Sie klicken. Nachdem Sie auf den Anschluss geklickt haben, können Sie zwischen der horizontalen und vertikalen Verbindungsrichtung wechseln, indem Sie die Leertaste drücken.

Diese Funktion ist auch bei aktiviertem Autorouting möglich. Sobald LabVIEW eine neue Verbindungsrichtung erkennt, nimmt der Draht diese Richtung ein.

Markieren von Verbindungen

Zum Markieren von Verbindungen klicken Sie diese mit dem Positionierwerkzeug entweder ein-, zwei- oder dreimal an. Mit einem einfachen Klick auf eine Verbindung wird ein Segment der Verbindung markiert. Mit einem Doppelklick wird ein Verbindungszweig markiert. Wenn Sie dreimal auf eine Verbindung klicken, wird die gesamte Verbindung markiert.

Beseitigen von Verbindungsfehlern

Eine unterbrochene Verbindung wird als gestrichelte schwarze Linie mit einem roten `X` in der Mitte dargestellt. Für unterbrochene Verbindungen gibt es eine Vielzahl von Gründen, beispielsweise, wenn Sie versuchen, zwei Objekte mit inkompatiblen Datentypen miteinander zu verbinden. Wenn Sie das Verbindungswerkzeug über eine unterbrochene Verbindung bewegen, wird ein Hinweisstreifen angezeigt, in dem erläutert wird, warum die Verbindung fehlerhaft ist. Gleichzeitig erscheint diese Information auch im Fenster **Kontext-Hilfe**. Klicken Sie mit der rechten Maustaste auf die Verbindung, und wählen Sie im Kontextmenü **Fehler auflisten** aus, um das Fenster **Fehlerliste** anzuzeigen. Weitere Informationen darüber, warum die jeweilige Verbindung unterbrochen ist, erhalten Sie über einen Klick auf die Schaltfläche **LabVIEW-Hilfe**.

Klicken Sie dreimal mit dem Positionierwerkzeug auf die Verbindung, und drücken Sie die <Entf>-Taste, um eine unterbrochene Verbindung zu entfernen. Sie können die Verbindung auch mit der rechten Maustaste anklicken und aus dem Kontextmenü zwischen Optionen wie **Verzweigung löschen**, **Verzweigung erstellen**, **Offene Enden entfernen**, **Verbindungen ausrichten**, **In Bedienelement umwandeln**, **In Anzeigeelement umwandeln**, **Indizierung an der Quelle aktivieren** oder **Indizierung an der Quelle deaktivieren** auswählen. Welche dieser Optionen jeweils angezeigt werden, hängt von der Ursache für die fehlerhafte Verbindung ab.

Um alle fehlerhaften Verbindungen zu löschen, wählen Sie **Bearbeiten» Ungültige Verbindungen entfernen** oder drücken Sie <Strg-B>.

(Mac OS) Drücken Sie <Befehlstaste-B>. **(UNIX)** Drücken Sie Tastenkombination <Meta-B>.

Vorsicht! Seien Sie beim Entfernen aller ungültigen Verbindungen vorsichtig. Manchmal scheint eine Verbindung ungültig zu sein, wenn das Blockdiagramm noch nicht fertig erstellt worden ist.

Typumwandlungspunkte

Wenn Sie zwei unterschiedliche Datentypen miteinander verbinden, werden auf dem Blockdiagrammknoten Typumwandlungspunkte angezeigt, mit denen auf die unterschiedliche Darstellung hingewiesen werden soll. Der Punkt bedeutet, dass der an den Knoten weitergeleitete Wert von LabVIEW in eine andere Darstellung konvertiert wurde. So sind zum Beispiel für die Additionsfunktion Gleitkommazahlen mit doppelter Genauigkeit als Eingangswerte zulässig. Wenn Sie einen der Eingänge für ganzzahlige Werte konfigurieren, erscheint an der Funktion ein Typumwandlungspunkt.

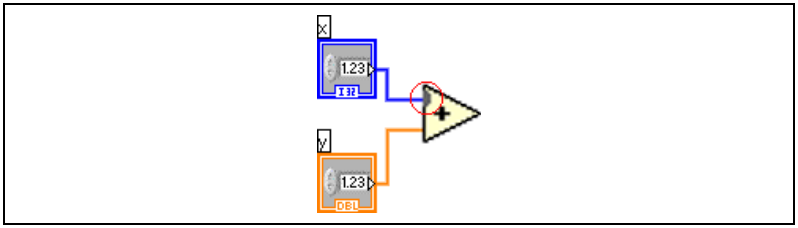


Abbildung 5-2. Typumwandlungspunkt

Im Blockdiagramm wird am Rand des Anschlusses ein Typumwandlungspunkt angezeigt, um darauf hinzuweisen, dass eine automatische numerische Umwandlung erfolgt ist. Da VIs und Funktionen über viele Anschlüsse verfügen können, kann ein Typumwandlungspunkt auch im Inneren des Symbols angezeigt werden, wenn Sie einen Anschluss mit einem anderen verbinden.

Typumwandlungspunkte werden auch angezeigt, wenn ein Anschluss des Typs “Variant” mit einem beliebigen Datentyp (außer einem anderen Variant) verbunden wird. Weitere Informationen zu Daten vom Typ “Variant” finden Sie in Abschnitt *Verarbeiten von Variant-Daten* dieses Kapitels.

Wenn ein VI Typumwandlungspunkte enthält, kann es langsamer und speicheraufwendiger werden. Daher sollten in VIs möglichst immer einheitliche Datentypen verwendet werden.

Polymorphe VIs und Funktionen

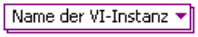
Polymorphe VIs und Funktionen können an die Eingabewerte unterschiedlicher Datentypen angepasst werden. Die meisten LabVIEW-Strukturen sind polymorph, so auch einige VIs und Funktionen.

Polymorphe VIs

Bei polymorphen VIs können an einen Ein- oder Ausgangsanschluss unterschiedliche Datentypen übergeben werden. Bei einem polymorphen VI handelt es sich um eine Anzahl von SubVIs mit denselben Anschlussfeldmustern. Jedes SubVI ist eine Instanz des polymorphen VIs.

Beispielsweise ist das VI “Schlüssel lesen” polymorph. An den Eingang **Standardwert** können boolesche Werte, numerische Fließkommawerte mit doppelter Genauigkeit, vorzeichenbehaftete 32-Bit-Ganzzahlen, Pfade, Strings und vorzeichenlose 32-Bit-Ganzzahlen übergeben werden.

Bei den meisten polymorphen VIs wird die zu verwendende Instanz bereits durch die Datentypen festgelegt, die an die Eingänge des VIs angelegt werden. Wenn das polymorphe VI kein SubVI für den entsprechenden Datentyp enthält, erscheint eine unterbrochene Verbindung. Wenn das SubVI durch den angelegten Datentyp nicht bestimmt wird, muss diese Einstellung manuell vorgenommen werden. In diesem Fall verhält sich das VI jedoch nicht mehr wie ein polymorphes VI, da es nur noch die Datentypen annimmt und ausgibt, die von Ihnen vorgegeben wurden.



Um manuell auszuwählen, welchen Typ das polymorphe VI annehmen soll, klicken Sie es mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Typ auswählen**. Stattdessen können Sie aber auch mit dem Bedienwerkzeug auf den Selektor des polymorphen VIs klicken (siehe Abbildung links) und aus dem Kontextmenü eine Instanz auswählen. Zur Anzeige des Selektors ist mit der rechten Maustaste auf das polymorphe VI zu klicken und **Sichtbare Objekte»Selektor** zu wählen. Wenn das polymorphe VI wieder mit allen Datentypen kompatibel sein soll, ist es entweder mit der rechten Maustaste anzuklicken und aus dem Kontextmenü **Typ auswählen»Automatisch** auszuwählen oder es ist mit dem Bedienwerkzeug der Selektor anzuklicken und aus dem Kontextmenü **Automatisch** auszuwählen.

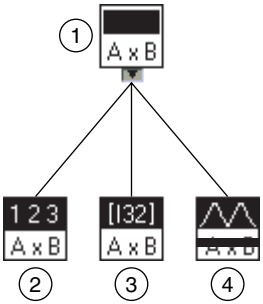
Erstellen polymorpher VIs

Polymorphe VIs sind dazu geeignet, gleiche Operationen mit unterschiedlichen Datentypen durchzuführen.

Hinweis Das Erstellen und Bearbeiten polymorpher VIs ist nur mit dem LabVIEW Professional Developnemt System möglich.

Wenn Sie beispielsweise die gleiche mathematische Operation mit einem Fließkommawert mit einfacher Genauigkeit, mit einem aus Zahlen bestehenden Array oder einem Signalverlauf durchführen, könnten Sie drei unterschiedliche VIs erstellen, nämlich “Zahl berechnen”, “Array berechnen” und “Signalverlauf berechnen”. Wenn die Operation dann durchgeführt werden soll, setzen Sie eines dieser VIs in das Blockdiagramm, und zwar in Abhängigkeit von dem bei der Eingabe verwendeten Datentyp.

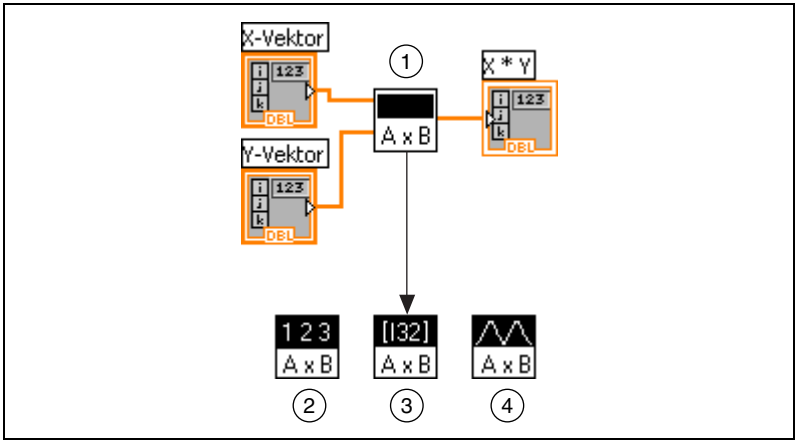
Anstatt nun jeweils manuell eine Version des VIs in das Blockdiagramm zu setzen, können Sie auch ein einziges polymorphes VI erstellen und verwenden. Das polymorphe VI “Berechnen” enthält drei Instanzen des VIs, wie in Abbildung 5-3 dargestellt.



1	Berechnen	3	Array berechnen
2	Zahl berechnen	4	Signalverlauf berechnen

Abbildung 5-3. Beispiel für ein polymorphes VI

Je nachdem, für welchen Datentyp das SubVI des VIs “Berechnen” vorgesehen ist, verknüpft dieses die entsprechende Instanz statisch, wie in Abbildung 5-4 dargestellt.



1	Berechnen	3	Array berechnen
2	Zahl berechnen	4	Signalverlauf berechnen

Abbildung 5-4. Polymorphes VI, das ein SubVI aufruft

Polymorphe VIs unterscheiden sich von den meisten VIs dadurch, dass sie kein Blockdiagramm oder Frontpanel haben.

Berücksichtigen Sie beim Erstellen von polymorphen VIs die folgenden Aspekte:

- Alle VIs, die in das polymorphe VI eingeschlossen werden, müssen über das gleiche Anschlussfeldmuster verfügen, da das Anschlussfeld des polymorphen VIs mit denen der Instanzen identisch ist.
- Die Eingänge und Ausgänge am Anschlussfeld jeder Instanz des VIs müssen den Eingängen und Ausgängen am Anschlussfeld des polymorphen VIs entsprechen.
- Die VIs, die zum Erstellen von polymorphen VIs verwendet werden, müssen nicht aus den gleichen SubVIs und Funktionen bestehen.
- Die jeweiligen Frontpanels der VIs müssen nicht über die gleiche Anzahl Objekte verfügen. Allerdings muss jedes Frontpanel zumindest die gleiche Anzahl Bedien- und Anzeigeelemente haben, aus denen sich das Anschlussfeld des polymorphen VIs zusammensetzt.
- Sie können ein Symbol für ein polymorphes VI erstellen.
- Polymorphe VIs können nicht in anderen polymorphen VIs verwendet werden.

Wenn Sie eine vollständige Dokumentation für ein VI erstellen, das ein polymorphes SubVI enthält, werden das polymorphe VI und alle darin enthaltenen VIs in der Liste der SubVIs aufgeführt.

Polymorphe Funktionen

Funktionen können auf unterschiedliche Weise polymorph sein: keine, einige oder alle Eingänge können polymorph sein. Einige Funktionseingänge akzeptieren Zahlen oder boolesche Werte. Andere akzeptieren Zahlen oder Strings. Daneben sind für einige polymorphe VIs nicht nur skalare Zahlen, sondern beispielsweise auch Arrays und Cluster mit numerischen Werten oder Arrays aus Clustern mit numerischen Werten zulässig. An einige polymorphe VIs können nur eindimensionale Arrays übergeben werden, wobei die Array-Elemente jedoch jeden Datentyp haben können. Einige Funktionen lassen alle Datentypen zu, komplexe Zahlen eingeschlossen. Weitere Informationen zu polymorphen Funktionen finden Sie in Anhang B, *Polymorphe Funktionen*.

Express-VIs

Express-VIs sind für allgemeine Arten von Messungen geeignet. Es handelt sich hierbei um Funktionsknoten, für die nur ein geringer Verbindungsaufwand betrieben werden muss, da sie über Dialogfelder konfiguriert werden. Ein- und Ausgänge von Express-VIs variieren je nach vorgenommenen Einstellungen. Express-VIs werden auf dem Blockdiagramm als erweiterbare Knoten dargestellt. Das jeweilige Symbol befindet sich auf einem blauen Hintergrund.

Nähere Informationen zu erweiterbaren Knoten befinden sich im Abschnitt *Darstellung von SubVIs und Express-VIs in Form von Symbolen und erweiterbaren Knoten* des Kapitels 7, *Erstellen von VIs und SubVIs*. Weitere Hinweise zu Express-VIs entnehmen Sie bitte dem Handbuch *Erste Schritte mit LabVIEW*.

Verwenden von Express-VIs als SubVIs

Es ist möglich, aus einem konfigurierten Express-VI ein SubVI zu erstellen. So kann zum Beispiel die Konfiguration des Express-VIs “LabVIEW-Messdaten in Datei schreiben” als SubVI gespeichert werden, so dass dieses in andere VIs integriert werden kann und das Express-VI nicht jedesmal neu konfiguriert werden muss. Um ein Express-VI in ein SubVI umzuwandeln, ist dieses mit der rechten Maustaste anzuklicken und aus dem Kontextmenü die Option **Frontpanel öffnen** zu wählen.

Wenn ein SubVI aus einem Express-VI erzeugt wird, wird das Frontpanel des SubVIs angezeigt. Dieses kann dann bearbeitet und gespeichert werden. Zum Speichern der über die Bedienelemente eingegebenen Werte wählen Sie **Ausführen»Aktuelle Werte als Standard** oder Sie klicken

jedes Bedienelement mit der rechten Maustaste an und wählen dann **Datenoperationen»Aktuellen Wert als Standard** aus dem Kontextmenü. Das Express-VI wird dann durch ein neues SubVI ersetzt, das auf dem Blockdiagramm als erweiterbarer Knoten dargestellt wird.

Es ist nicht möglich, ein VI, das aus einem Express-VI erstellt worden ist, wieder in ein solches umzuwandeln.

Dynamische Daten

Anschlüsse des Datentyps “dynamisch” werden—wie links abgebildet—auf dem Blockdiagramm dunkelblau dargestellt. Dieser Datentyp wird auch von den meisten Express-VIs angenommen und/oder ausgegeben. Verbindungen dieses Typs können zu jedem Anzeigeelement oder Eingang erstellt werden, der für numerische, boolesche oder Signalverlaufsdaten geeignet ist. Dabei sollte immer ein Anzeigeelement gewählt werden, mit dem sich die Werte bestmöglich darstellen lassen. Zu den Anzeigeelementen gehören Graphen, Diagramme und numerische Anzeigen.

Der Datentyp “dynamisch” ist speziell für Express-VIs entwickelt worden und ist daher für die meisten sonstigen Funktionen, mit denen LabVIEW ausgeliefert wird, ungeeignet. Um dynamische Daten mit einem in LabVIEW integrierten VI zu analysieren oder zu verarbeiten, müssen diese vorher in einen anderen Datentyp umgewandelt werden. Weitere Informationen hierzu finden Sie im Abschnitt *Umwandeln dynamischer Daten in andere Datentypen* dieses Kapitels.

Außer den Daten des Signals selbst sind in dynamischen Daten auch Informationen zum Signal gespeichert, wie zum Beispiel Datum und Erfassungszeit. Dadurch wird bestimmt, wie das Signal in einem Graphen oder Diagramm dargestellt wird. Wenn beispielsweise mit dem Express-VI “DAQ-Assistent” ein Signal erfasst und in einem Graphen dargestellt wird, erscheint die Bezeichnung des Signals in der Legende des Graphen und die x-Achse passt sich automatisch an die Zeitwerte des Signals an, die je nach Signalattributen relativ oder absolut angegeben sein können. Bei einer FFT-Analyse eines Signals mit Hilfe des Express-VIs “Spektrummessung” und Darstellung der Ergebnisse in einem Graphen passt sich die x-Achse so an, dass das Signal im Frequenzbereich dargestellt wird, der über die Signalattribute bestimmt wird. Zur Anzeige der ausgegebenen Werte in einem Graphen klicken Sie den Ausgang für dynamische Daten eines VIs oder einer Funktion mit der rechten Maustaste an und wählen aus dem Kontextmenü die Option **Erstelle»Graph-Anzeige**. Ansonsten wählen Sie **Erstelle»Numerische Ausgabe**.

Tabelle 5-2 zeigt alle Anzeigeelemente, die für den dynamischen Datentyp und die darin enthaltenen Daten geeignet sind, und es wird beschrieben, wie diese jeweils verarbeitet werden.

Tabelle 5-2. Anzeigeelemente für dynamische Daten

Messdaten in Signalen dynamischen Datentyps	Anzeige-element	Ergebnis
einzelner numerischer Wert	Graph	Stellt den einzelnen Wert mit Zeitstempel und Attributen dar.
einzelner Kanal		Stellt den gesamten Signalverlauf unter Berücksichtigung der Zeitstempel und Attribute dar.
mehrere Kanäle		Stellt alle Daten einschließlich der Zeitstempel und Attribute dar.
einzelner numerischer Wert	numerische Ausgabe	Zeigt den einzelnen Wert an.
einzelner Kanal		Zeigt den letzten Datenwert des Kanals an.
mehrere Kanäle		Zeigt den letzten Datenwert des ersten Kanals an.
einzelner numerischer Wert	boolesche Anzeige	Zeigt “TRUE” an, wenn der numerische Wert größer oder gleich 0,5 ist.
einzelner Kanal		Zeigt “TRUE” an, wenn der letzte Datenwert des Kanals größer gleich 0,5 ist.
mehrere Kanäle		Zeigt “TRUE” an, wenn der letzte Datenwert des ersten Kanals größer oder gleich 0,5 ist.

Umwandeln dynamischer Daten in andere Datentypen

Mit dem Express-VI “Umwandeln dynamischer Daten” können dynamische Daten in einen anderen Datentyp, wie beispielsweise “numerisch”, “Signalverlauf” oder “Array”, konvertiert werden, so dass sie mit anderen VIs und Funktionen verwendet werden können. Wenn Sie dieses Express-VI in das Blockdiagramm einfügen, wird das Dialogfeld **Einstellungen zur Umwandlung dynamischer Daten** angezeigt. Hier finden Sie Optionen, mit denen bestimmt werden kann, wie die Daten, die von diesem Express-VI ausgegeben werden, formatiert sein sollen.

Wenn zum Beispiel von einem Datenerfassungsgerät ein sinusförmiges Signal vorliegt, sollte die Option **Einzelner Signalverlauf** gewählt werden. Anschließend ist der Ausgang **Signalverlauf** des Express-VIs "Umwandeln dynamischer Daten" mit einer Funktion oder einem für diesen Datentyp geeigneten VI zu verbinden. Wenn stattdessen mit einem DAQ-Gerät Temperaturwerte von verschiedenen Kanälen erfasst worden sind, sollten die Optionen **Letzter Wert jedes Kanals** und **Fließkommazahlen (DOUBLE)** verwendet werden. Anschließend ist der **Array**-Ausgang des Express-VIs mit einer Funktion oder einem VI zu verbinden, für das dieser Datentyp zulässig ist.

Wenn dynamische Daten in Array-Anzeigeelementen dargestellt werden sollen, wird das Express-VI "Umwandeln dynamischer Daten" automatisch in das Blockdiagramm platziert. Um festzulegen, wie die Daten im Array dargestellt werden sollen, klicken Sie das Express-VI doppelt an, um das Dialogfeld **Einstellungen zur Umwandlung dynamischer Daten** zu öffnen.

Umwandeln in dynamische Daten

Mit Hilfe des Express-VIs "Umwandeln in dynamische Daten" können numerische, boolesche, Signalverlaufs- und Array-Daten in dynamische Daten umgewandelt werden, so dass sie auch mit Express-VIs verwendet werden können. Wenn Sie dieses Express-VI in das Blockdiagramm einfügen, wird das Dialogfeld **Einstellungen zur Umwandlung in dynamische Daten** angezeigt. Hier kann ausgewählt werden, welcher Datentyp in den dynamischen Datentyp konvertiert werden soll.

Wenn beispielsweise ein sinusförmiges Signal vorliegt, das mit den Analogeingangs-VIs für den herkömmlichen NI-DAQ-Treiber erfasst wurde und mit dem Express-VI "Signalanalyse" analysiert werden soll, wählen Sie im Dialogfeld **Einstellungen zur Umwandlung in dynamische Daten** die Option "Einzelner Signalverlauf". Verbinden Sie dann den Ausgang **Dynamische Daten** mit einem Express-VI, das für die Eingangssignale dieses Datentyps geeignet ist.

Verarbeiten von Variant-Daten

Variant-Daten können keinem bestimmten Datentyp zugeordnet werden. Sie können Attribute enthalten. Diese Daten werden in LabVIEW als Datentyp "Variant" dargestellt. Variant-Daten unterscheiden sich insofern von anderen Datentypen, als dass bei diesem Datentyp neben den eigentlichen Messwerten auch der Name des Bedien- oder Anzeigeelements und Informationen zum Datentyp, aus dem konvertiert worden ist, enthalten

sind. Dadurch kann LabVIEW die Variant-Daten in den korrekten Datentyp umwandeln. Wenn beispielsweise ein String in Daten des Typs “Variant” konvertiert wird, speichert der Datentyp “Variant” neben dem Text auch die Information, dass es sich um einen String handelt.

Die Variant-Funktionen dienen zur Erzeugung und Manipulation von Variant-Daten. Grundsätzlich kann jeder LabVIEW-Datentyp für die Verwendung in anderen VIs und Funktionen in den Typ “Variant” umgewandelt werden. Ausgegeben wird dieser Datentyp von verschiedenen polymorphen VIs.

Die Verwendung des Variant-Datentyps empfiehlt sich immer dann, wenn es darauf ankommt, Daten unabhängig vom Typ zu manipulieren, wie zum Beispiel bei der Übertragung oder Speicherung von Daten, beim Auslesen und/oder Beschreiben unbekannter Geräte, beim Speichern von Daten in einen Stapel, eine Queue bzw. einen Melder, oder wenn an den Daten Operationen mit den unterschiedlichsten Steuerungselementen durchgeführt werden sollen.

Um Daten unabhängig vom Typ darzustellen, kann auch die Funktion “In String konvertieren” verwendet werden. Diese Umwandlung ist sinnvoll, wenn Daten mit TCP/IP übertragen werden, da dieses Protokoll nur mit Strings arbeitet.

Jedoch gibt es bei der Verwendung von in Strings konvertierten Daten auch Einschränkungen. So kann beispielsweise keine Umwandlung erfolgen, wenn der ursprüngliche Datentyp nicht mit dem übereinstimmt, in den die Daten konvertiert werden sollen. Darüber hinaus ist es auch nicht möglich, eine in einen String umgewandelte Ganzzahl als Fließkommazahl mit erweiterter Genauigkeit darzustellen. Weitere Informationen zum Umwandeln von Daten in String-Daten und umgekehrt finden Sie in Abschnitt *Flattened Data* der Application Note *LabVIEW Data Storage*.

Ein weiterer Vorteil des Variant-Datentyps besteht darin, dass auch Attribute zu den Daten gespeichert werden können, also Informationen über die Art der Daten. Wenn Sie zum Beispiel wissen möchten, wann ein Datenelement erzeugt worden ist, kann beim Variant-Typ zu den Daten das Attribut **Zeit** hinzugefügt werden, in dem der entsprechende Zeit-String gespeichert ist. Die Daten mit den Attributen können von beliebigem Datentyp sein. Mit Hilfe dieser Attribute können Daten nach bestimmten Eigenschaften sortiert oder gefiltert werden, und es kann das Gerät oder die Anwendung ermittelt werden, von dem/der die Daten stammen.

Numerische Einheiten und strikte Typenüberprüfung

Grundsätzlich können jedem numerischen Bedien- oder Anzeigeelement, das mit Fließkommazahlen arbeitet, physikalische Maßeinheiten wie Meter oder Sekunde zugeordnet werden.

Die Einheiten für ein Bedienelement werden in einer separaten verknüpften Beschriftung, der so genannten Einheitenbeschriftung, angezeigt. Zur Anzeige der Einheitenbeschriftung klicken Sie mit der rechten Maustaste auf das Bedienelement und wählen Sie aus dem Kontextmenü die Option **Sichtbare Objekte»Einheitenbeschriftung** aus. Um eine Einheit zu bearbeiten, klicken Sie diese mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Einheitenstring erstellen** aus.

Wenn LabVIEW die Einheitenbeschriftung anzeigt, können die Einheiten unter Verwendung der Standardabkürzungen wie beispielsweise `m` für Meter oder `s` für Sekunde eingegeben werden.

Hinweis In Formelknoten können keine Einheiten verwendet werden.

Einheiten und strikte Typenüberprüfung

Wenn Sie einem Objekt Einheiten zuweisen, können Sie nur Objekte verbinden, die über kompatible Einheiten verfügen. LabVIEW verwendet die strikte Typenüberprüfung, um sicherzustellen, dass die Einheiten kompatibel sind. Wenn Sie zwei Objekte mit inkompatiblen Einheiten verbinden, gibt LabVIEW einen Fehler aus. Dies geschieht z. B. dann, wenn Sie ein Objekt mit der Einheit Meilen mit einem Objekt mit der Einheit Liter verbinden, da Meilen ein Längenmaß und Liter ein Raummaß ist.

In Abbildung 5-5 sehen Sie, wie Objekte mit kompatiblen Einheiten verbunden werden. In dieser Abbildung skaliert LabVIEW automatisch den **Entfernungs**-Anzeiger so, dass Kilometer anstelle von Metern angezeigt werden, da das Anzeigeelement auf Kilometer eingestellt ist.

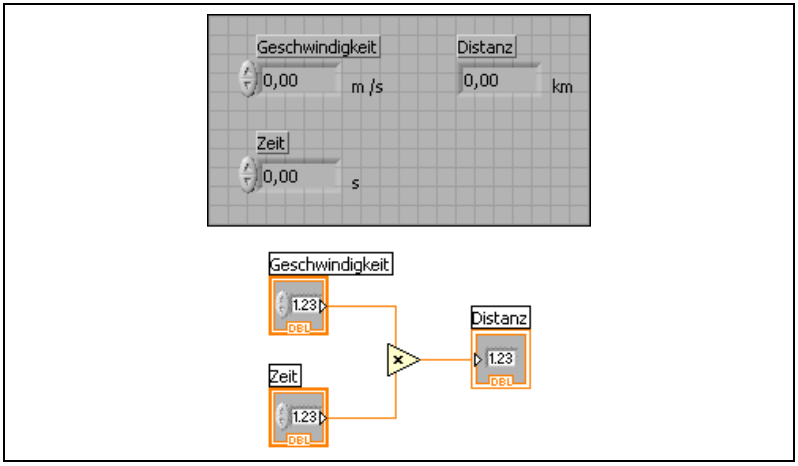


Abbildung 5-5. Verbinden von Objekten mit kompatiblen Einheiten

In Abbildung 5-6 wird ein Fehler ausgegeben, weil für die **Distanz** die Einheit “Sekunde” angegeben wurde. Zum Beheben dieses Fehlers ändern Sie die Sekunden in ein Längenmaß wie beispielsweise Kilometer, wie in Abbildung 5-5 dargestellt.

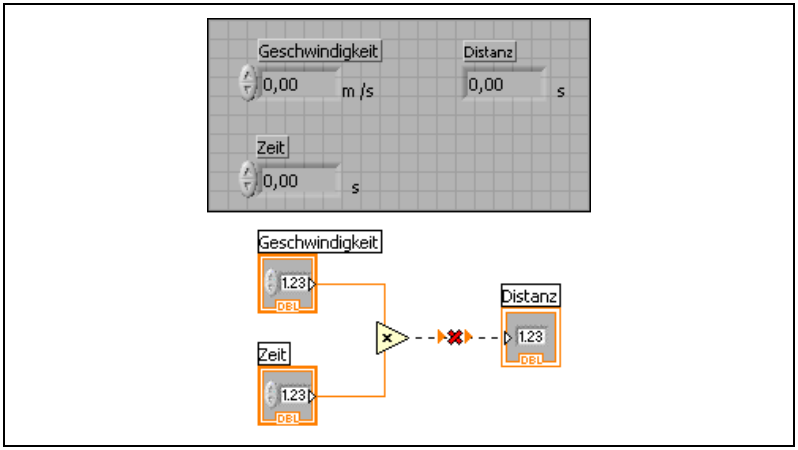


Abbildung 5-6. Wenn Objekte mit inkompatiblen Einheiten verbunden werden, entstehen fehlerhafte Verbindungen

Einige VIs und Funktionen sind im Hinblick auf Einheiten doppeldeutig. Diese VIs und Funktionen können Sie nicht mit anderen Anschlüssen verwenden, die Einheiten aufweisen. Beispielsweise ist die Funktion “Inkrement” doppeldeutig im Hinblick auf Einheiten. Wenn Sie Längenmaße verwenden, kann die Funktion “Inkrement” nicht feststellen, ob ein

Meter, ein Kilometer oder ein Fuß hinzugefügt werden soll. Aufgrund dieser Doppeldeutigkeit können die Funktion “Inkrement” und andere Funktionen, mit denen Werte erhöht oder verringert werden, nicht mit Daten verwendet werden, denen Einheiten zugewiesen wurden.

Zur Vermeidung der Doppeldeutigkeit in diesem Beispiel ist eine numerische Konstante mit der korrekten Einheit zu verwenden sowie die Additionsfunktion, um eine eigene Inkrement-Funktion zum Erhöhen von Werten mit Einheiten zu erstellen, wie in Abbildung 5-7 dargestellt.

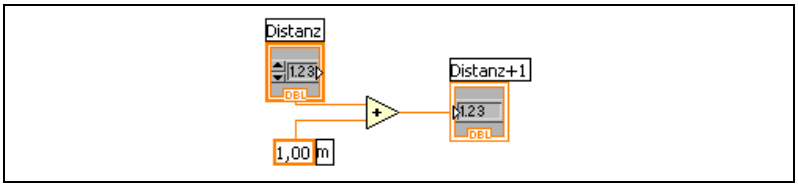


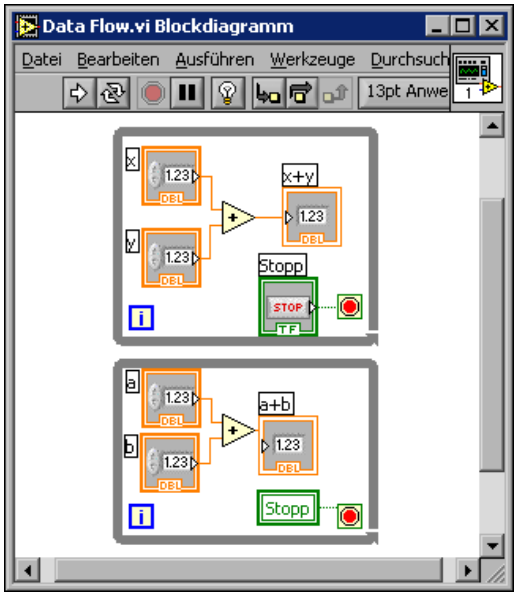
Abbildung 5-7. Erstellen einer Inkrement-Funktion mit Einheiten

Datenflussprinzip Blockdiagramm

Für das Ausführen von VIs folgt LabVIEW einem Datenflussmodell. Ein Blockdiagrammknoten wird ausgeführt, wenn an allen seinen Eingängen Daten verfügbar sind. Wenn die Ausführung eines Knotens abgeschlossen ist, werden die Daten an die Ausgangsanschlüsse übergeben und diese Ausgabedaten an den nächsten Knoten im Datenflusspfad weitergeleitet.

Visual Basic, C++, JAVA und die meisten anderen textbasierten Programmiersprachen folgen bei der Programmausführung einem Steuerflussmodell. Im Steuerfluss bestimmt die sequenzielle Reihenfolge der Programmelemente die Ausführungsreihenfolge eines Programms.

Da in LabVIEW der Datenfluss und nicht die sequenzielle Reihenfolge der Befehle die Ausführungsreihenfolge der Blockdiagrammelemente bestimmt, können Sie Blockdiagramme erstellen, die simultane Operationen beinhalten. So können zum Beispiel zwei While-Schleifen gleichzeitig ausgeführt und die Ergebnisse auf dem Frontpanel angezeigt werden.



LabVIEW ist ein multiasking- und multithreadingfähiges System, bei dem mehrere Ausführungs-Threads und mehrere VIs gleichzeitig ausgeführt werden können. Weitere Informationen zum simultanen Ausführen von Aufgaben finden Sie in der Application Note *Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability*.

Datenabhängigkeit und künstliche Datenabhängigkeit

Die Ausführung eines Programms kann befehls- oder datenflussorientiert sein. Bei der datenflussorientierten Programmierung wird ein Knoten, der Daten von einem anderen Knoten empfängt, wird immer erst dann ausgeführt, nachdem die Ausführung des vorhergehenden Knotens abgeschlossen ist.

Blockdiagrammknoten, die untereinander keine Verbindungen aufweisen, können in jeder beliebigen Reihenfolge ausgeführt werden. Obwohl in den *LabVIEW Development Guidelines* empfohlen wird, ein von links nach rechts und von oben nach unten orientiertes Layout zu verwenden, werden Knoten nicht notwendigerweise in der Reihenfolge von links nach rechts und von oben nach unten ausgeführt.

Bei Fehlen einer natürlichen Datenabhängigkeit kann die Ausführungsreihenfolge auch durch eine Sequenzstruktur gesteuert werden. Weitere Informationen zu Sequenzstrukturen finden Sie in Abschnitt *Sequenzstruk-*

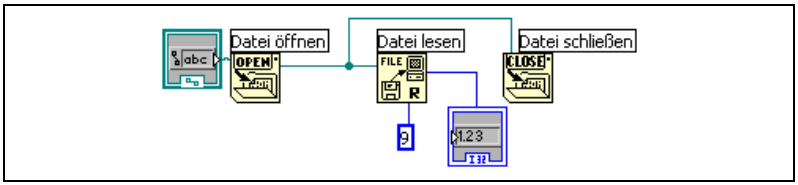
turen des Kapitels 8, *Schleifen und Strukturen*. Sie können jedoch auch Durchflussparameter zum Steuern der Ausführungsreihenfolge verwenden. Nähere Einzelheiten zu Durchflussparametern entnehmen Sie bitte dem Abschnitt *Durchgeschliffene Parameter* des Kapitels 14, *Datei-I/O*.

Eine weitere Möglichkeit, die Ausführungsreihenfolge zu steuern, besteht darin, eine künstliche Datenabhängigkeit zu schaffen. In diesem Fall werden die übergebenen Daten vom entsprechenden Knoten nicht wirklich benötigt. Stattdessen verwendet der empfangende Knoten den Datenempfang als Auslöser für die Programmausführung. Ein Beispiel hierzu ist das VI “Timing Template (data dep)”, das sich unter `examples\general\structs.llb` befindet.

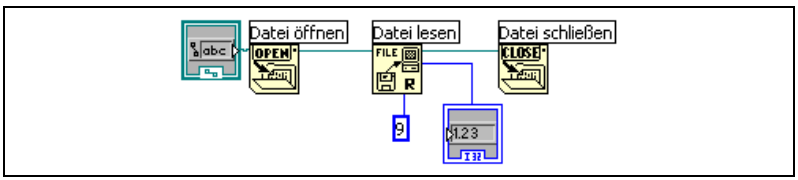
Fehlende Datenabhängigkeit

Gehen Sie nicht von einer Ausführungsreihenfolge von links nach rechts oder von oben nach unten aus, wenn keine Datenabhängigkeit existiert. Stellen Sie sicher, dass Sie die Sequenz der Ereignisse gegebenenfalls explizit definieren, indem Sie den Datenfluss durch die entsprechenden Verbindungen vorgeben.

Im folgenden Beispiel besteht keine Abhängigkeit zwischen den Funktionen “Datei lesen” und “Datei schreiben”, da diese beiden Funktionen nicht miteinander verbunden sind. Dieses Beispiel funktioniert möglicherweise nicht wie erwartet, weil es keine Möglichkeit gibt festzulegen, welche Funktion zuerst ausgeführt wird. Wenn die Funktion “Datei schließen” zuerst ausgeführt wird, arbeitet die Funktion “Datei schreiben” nicht.



Im folgenden Blockdiagramm wird eine Abhängigkeit eingerichtet, indem ein Ausgang der Funktion “Datei lesen” mit der Funktion “Datei schließen” verbunden wird. Die Funktion “Datei schließen” wird erst ausgeführt, wenn die Ausgabe der Funktion “Datei lesen” empfangen wird.



Datenfluss und Speicherverwaltung

Bei der datenflussorientierten Ausführung gestaltet sich die Speicherverwaltung einfacher als bei der befehlsorientierten Ausführung. In LabVIEW werden keine Variablen allokiert oder diesen Werte zugewiesen. Statt dessen erstellen Sie ein Blockdiagramm mit Verbindungen, welche die Datenübergabe repräsentieren.

VIs und Funktionen, die Daten generieren, weisen auch automatisch den Speicher für diese Daten zu. Wenn das VI oder die Funktion die Daten nicht mehr verarbeitet, hebt LabVIEW die Speicherzuweisung auf. Wenn Sie einem Array oder String neue Daten hinzufügen, weist LabVIEW zur Verwaltung dieser neuen Daten wieder ausreichend Speicher zu.

Da LabVIEW den Speicher automatisch verwaltet, haben Sie weniger Einfluss darauf, wann Speicher zugewiesen und wann die Zuweisung aufgehoben wird. Wenn das VI mit großen Datensätzen arbeitet, müssen Sie wissen, wann die Speicherzuweisung erfolgt. Ein Verständnis der zugrunde liegenden Prinzipien kann Ihnen dabei helfen, VIs zu schreiben, die bedeutend weniger Speicheranforderungen stellen. Eine geringere Speichernutzung kann dazu beitragen, die Geschwindigkeit zu steigern, mit der VIs ausgeführt werden.

Lesen Sie bitte zur Speicherzuweisung die Application Note *LabVIEW Performance and Memory Management*. Weitere Einzelheiten dazu, wie durch Auswahl des richtigen Datentyps bei der Entwicklung eines VIs die Speichernutzung optimiert werden kann, finden Sie auch in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter *Memory and Speed Optimization*.

Entwerfen des Blockdiagramms

Bei der Erstellung von Blockdiagrammen sollten folgende Hinweise beachtet werden:

- Gehen Sie beim Aufbau des Blockdiagramms immer von links nach rechts und von oben nach unten vor. Obwohl die Position der Blockdiagrammelemente keinen Einfluss auf die Ausführungsreihenfolge hat, wirkt das Blockdiagramm geordneter und ist einfacher zu verstehen, wenn Sie eine Datenflussrichtung beibehalten. Die Ausführungsreihenfolge wird ausschließlich durch die Verbindungen und Strukturen bestimmt.
- Vermeiden Sie es, Blockdiagramme zu erstellen, die mehr als einen oder zwei Bildschirmbreiten in Anspruch nehmen. Wenn ein Block-

diagramm umfangreich und komplex wird, ist es schwieriger zu verstehen und die Fehlersuche wird unter Umständen erschwert.

- Prüfen Sie, ob Sie einige Komponenten des Blockdiagramms in anderen VIs wiederverwenden können oder ob ein Abschnitt des Blockdiagramms als logische Komponente zusammengefasst werden kann. Falls ja, teilen Sie das Blockdiagramm in SubVIs auf, die bestimmte Aufgaben erledigen. Die Verwendung von SubVIs hilft bei der Änderungsverwaltung und sorgt dafür, dass Fehler in Blockdiagrammen schnell behoben werden können. Weitere Informationen zu SubVIs befinden sich im Abschnitt *SubVIs* des Kapitels 7, *Erstellen von VIs und SubVIs*.
- Verwenden Sie die Fehlerbehandlungs-VIs, –Funktionen und –Parameter, um Fehler im Blockdiagramm zu erkennen. Weiter Hinweise zur Fehlerbehandlung befinden sich in Abschnitt *Fehlerprüfung und Fehlerbehandlung* des Kapitels 6, *Ausführen von VIs und Fehlersuche*.
- Gestalten Sie das Blockdiagramm übersichtlich, indem Sie für kurze und klare Verbindungen sorgen. Blockdiagramme sind zwar auch funktionstüchtig, wenn sie ineffizient verdrahtet sind, können jedoch schwierig zu lesen sein und im Fehlerfall die Suche erschweren. Auch kann in einem unübersichtlichen Blockdiagramm der Anschein entstehen, dass das VI Funktionen ausführt, die nicht vorgesehen sind.
- Vermeiden Sie es, Verbindungen unter einen Strukturrahmen oder zwischen sich überlappenden Objekten zu ziehen, da LabVIEW in diesem Fall möglicherweise einige Verbindungssegmente überdeckt.
- Vermeiden Sie es, Objekte über Verbindungen zu platzieren. Mit Verbindungen werden nur die Objekte miteinander verknüpft, auf die Sie klicken. Wenn Sie einen Anschluss oder ein Symbol auf eine Verbindung setzen, könnte der Eindruck entstehen, als bestünde eine Verbindung.

- Verwenden Sie für Kommentare zum Blockdiagramm freie Beschriftungen.
- Um den Leerraum zwischen gehäuften oder zu nahe gruppierten Objekten zu vergrößern, drücken Sie die <Strg>-Taste und klicken Sie mit dem Positionierwerkzeug in den Blockdiagramm-Arbeitsbereich. Ziehen Sie den Bereich in der Größe auf, wie Sie Freiraum einfügen möchten.

  (Mac OS) Drücken Sie <Befehlstaste>. **(Sun)** Drücken Sie die <Meta>-Taste. **(Linux)** Drücken Sie die <Alt>-Taste.

  Ein von einem gestrichelten Rahmen umschlossenes Rechteck zeigt, wo der Leerraum eingefügt wird. Lassen Sie die Taste los, um den Leerraum einzufügen.

Weitere Hinweise zu diesem Thema finden Sie auch in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter *Block Diagram Style*.

6

Ausführen von VIs und Fehlersuche

Zum Ausführen eines VIs müssen Sie zunächst alle SubVIs, Funktionen und Strukturen mit den für die Anschlüsse zulässigen Datentypen verbinden. Es kann vorkommen, dass ein VI unerwartete Daten produziert oder nicht wie geplant abläuft. Mit Hilfe von LabVIEW können Sie die Ausführung von VIs konfigurieren und Fehler in der Blockdiagrammanordnung bzw. im Programmablauf erkennen.

Weitere Informationen ...

Weitere Informationen zur VI-Ausführung und zur Fehlersuche in VIs finden Sie in der *LabVIEW-Hilfe*.

Ausführen von VIs

Beim Ausführen eines VI wird die Funktion realisiert, für die das VI erstellt worden ist. Ein VI ist immer dann ausführbar, wenn die Schaltfläche **Ausführen** auf der Symbolleiste als weißer Pfeil wie links dargestellt erscheint. Außerdem zeigt der weiße Pfeil an, dass das VI als SubVI verwendet werden kann, wenn für das VI ein Anschlussfeld erstellt wurde. Bei Betätigung der Schaltfläche **Ausführen** wird der Pfeil wie in der Abbildung links schwarz dargestellt. Solange das VI läuft, sind keine Änderungen am Blockdiagramm möglich.

Das VI wird ausgeführt, wenn Sie entweder auf die Schaltfläche **Ausführen**, **Wiederholt ausführen** oder die Einzelschrittschaltflächen auf der Blockdiagramm-Symbolleiste klicken. Mit dem Klicken auf die Schaltfläche **Ausführen** wird das VI einmal ausgeführt. Das VI stoppt, wenn dessen Datenfluss abgeschlossen ist. Bei Betätigung der Schaltfläche **Wiederholt ausführen** (links dargestellt) läuft ein VI so lange kontinuierlich ab, bis es per Hand wieder angehalten wird. Wenn Sie auf die Einzelschrittschaltflächen klicken, wird das VI schrittweise ausgeführt. Weitere Informationen zum Einsatz der Einzelschrittschaltflächen bei der Fehlersuche in VIs finden Sie in Abschnitt *Einzelschrittausführung* dieses Kapitels.

Hinweis Vermeiden Sie es nach Möglichkeit, VIs über die Schaltfläche **Ausführung abbrechen** zu  stoppen. Lassen Sie statt dessen das VI vollständig ausführen, oder entwickeln Sie eine Methode, um das VI programmatisch zu anzuhalten. Dadurch befindet sich das VI in einem bekannten Status. Dazu kann zum Beispiel auf das Frontpanel eine Schaltfläche gelegt werden, bei deren Betätigung das Programm anhält.

Ausführungskonfiguration eines VIs

Um Einstellungen zur Ausführung eines VIs zu vorzunehmen, wählen Sie **Datei»VI-Einstellungen**, und klicken Sie im Pulldown-Menü **Kategorie** auf **Ausführung**. Sie können ein VI beispielsweise so konfigurieren, dass es beim Öffnen sofort gestartet wird, oder anhält, wenn es als SubVI aufgerufen wird. Darüber hinaus können VIs unterschiedliche Prioritäten zugewiesen werden. Wenn ein VI beispielsweise unbedingt ausgeführt werden muss, ohne auf den Abschluss einer anderen Operation zu warten, konfigurieren Sie das VI zur Ausführung mit zeitkritischer (höchster) Priorität. Ausführlichere Hinweise zum Erstellen von Multithread-VIs finden Sie in der Application Note *Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability* und zur Konfiguration der Ausführung von VIs im Kapitel 16, *Anpassen von VIs*.

Hinweis Wenn die Ausführung eines VIs mit unterschiedlichen Prioritäten falsch konfiguriert wird, kann das unter Umständen zu einem unerwarteten Verhalten des VIs führen. Bei den meisten VIs ist die Einstellung unterschiedlicher Ausführungsprioritäten jedoch nicht notwendig.

Fehlerbeseitigung in nicht ausführbaren VIs

Wenn ein VI nicht gestartet werden kann, enthält es einen Fehler. Die Schaltfläche **Ausführen** ist dann, wie in der Abbildung links, gebrochen dargestellt. Häufig erscheint der Pfeil auch während der Erstellung und Bearbeitung eines VIs gebrochen, solange dieses noch fehlerhaft ist. Wenn er sich nach Fertigstellung aller Verbindungen im Blockdiagramm nicht in einen intakten Pfeil verwandelt, ist das VI nicht ausführbar.

Fehlersuche in nicht ausführbaren VIs

Klicken Sie auf die gebrochene Schaltfläche **Ausführen** oder wählen Sie **Fenster»Fehlerliste anzeigen**, um herauszufinden, warum ein VI nicht ausgeführt werden kann. Im Fenster **Fehlerliste** werden alle Fehler aufgeführt. Im Abschnitt **VI-Liste** werden die Namen aller VIs im Speicher aufgeführt, die fehlerhaft sind. Im Abschnitt **Fehler und Warnungen** werden die Fehler und Warnungen für das VI angezeigt, das Sie im

Abschnitt **VI-Liste** markiert haben. Im Abschnitt **Details** werden die Fehler und in manchen Fällen auch Empfehlungen zur Fehlerbehebung und Hinweise auf weitere Informationen hierzu angezeigt. Wählen Sie **Hilfe**, um eine Online-Hilfedatei zu öffnen, in der die LabVIEW-Fehler und entsprechende Beschreibungen aufgeführt sind.

Klicken Sie auf die Schaltfläche **Fehler anzeigen** oder klicken Sie die Fehlerbeschreibung doppelt an, um das Blockdiagramm oder Frontpanel anzuzeigen und um das Objekt hervorzuheben, das den Fehler enthält.

Wenn Sie das Kontrollkästchen **Warnungen anzeigen** im Fenster **Fehlerliste** aktiviert haben und für ein VI eine Warnung ausgegeben wird, sehen Sie in der Symbolleiste die Schaltfläche **Warnung** (siehe links).

Konfigurieren Sie LabVIEW immer so, dass Warnungen im Fenster **Fehlerliste** angezeigt werden, indem Sie **Werkzeuge»Optionen** wählen, dann im Pulldown-Menü auf **Fehlersuche** klicken und das Kontrollkästchen **Standardmäßig Warnungen im Fehlerfenster anzeigen** aktivieren. Sie können diese Änderung bei geöffnetem Fenster **Fehlerliste** vornehmen, so dass die Änderung sofort sichtbar ist.

Warnungen verhindern nicht das Ausführen eines VIs. Sie dienen dazu, potenzielle Probleme mit VIs zu vermeiden.

Häufige Ursachen für nicht ausführbare VIs

Die folgende Liste enthält häufige Ursachen, warum ein VI als nicht ausführbar gekennzeichnet wird, während Sie es bearbeiten:

- Das Blockdiagramm enthält eine unterbrochene Verbindung aufgrund sich nicht entsprechender Datentypen oder loser, nicht verbundener Enden. Weitere Informationen zum Verbinden von Blockdiagrammobjekten finden Sie in Abschnitt *Verbindung von Blockdiagrammobjekten* des Kapitels 5, *Erstellen des Blockdiagramms*.
- Ein obligatorischer Blockdiagrammanschluss wurde nicht verbunden. Weitere Hinweise zu Anschlüssen, die verbunden werden müssen, damit ein VI funktionstüchtig ist, entnehmen Sie bitte dem Abschnitt *Festlegen von erforderlichen, empfohlenen und optionalen Ein- und Ausgängen* des Kapitels 7, *Erstellen von VIs und SubVIs*.
- Ein SubVI ist nicht ausführbar, oder Sie haben dessen Anschlussfeld bearbeitet, seit Sie das Symbol des SubVIs auf das Blockdiagramm des VIs gesetzt haben. Weitere Informationen zu SubVIs finden Sie im Abschnitt *SubVIs* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Fehlersuchtechniken

Wenn ein VI zwar ausführbar ist, jedoch unerwartete Daten ausgibt, können mit Hilfe folgender Techniken Probleme mit dem VI oder dem Blockdiagramm-Datenfluss erkannt und behoben werden.

- Verbinden Sie die Parameter Fehlerein- und -ausgang, die sich am unteren Rand der meisten in LabVIEW verfügbaren VIs und Funktionen befinden. Mit diesen Parametern können Fehler erkannt werden, die in den Knoten des Blockdiagramm aufgetreten sind, und es wird angegeben, ob und wo ein Fehler eingetreten ist. Diese Parameter können in jedem beliebigen VI verwendet werden. Weitere Informationen zum Einsatz dieser Parameter finden Sie im Abschnitt *Fehlerbehandlung* dieses Kapitels.
- Zur Ermittlung aller Ursachen, die zu Warnungen im VI führen, wählen Sie **Fenster»Fehlerliste anzeigen** und aktivieren Sie das Kontrollkästchen **Warnungen anzeigen**. Ermitteln und beheben Sie dann die im VI aufgetretenen Probleme.
- Klicken Sie mit dem Positionierwerkzeug dreifach auf eine Verbindung, um sie zusammen mit all ihren Abzweigungen zu markieren und sicherzustellen, dass die richtigen Anschlüsse verdrahtet worden sind.
- Über die **Kontext-Hilfe** können Sie sich zu allen Funktionen und VIs auf dem Blockdiagramm die voreingestellten Werte anzeigen lassen. Wenn Eingänge, deren Verbindung optional ist oder empfohlen wird, offen bleiben, werden von vielen VIs und Funktionen dafür Standardwerte verwendet. Ein boolesches Eingabeelement, das nicht verbunden ist, kann beispielsweise immer den Wert TRUE haben.
- Um SubVIs, Text und andere Objekte im VI zu finden, verwenden Sie die **Suchfunktion**.
- Wählen Sie zur Suche unverbundener SubVIs die Option **Durchsuchen»VI-Hierarchie anzeigen** aus. Im Gegensatz zu unverbundenen Funktionen erzeugen unverbundene VIs nicht immer Fehler – es sei denn, es handelt sich um einen Eingang, der verdrahtet werden muss, damit das VI funktionstüchtig ist. Wenn Sie versehentlich ein unverbundenes SubVI in das Blockdiagramm platziert haben, wird es parallel zum übergeordneten VI ausgeführt, sobald dieses gestartet wird. Es kann allerdings vorkommen, dass dadurch bestimmte Funktionen ausgeführt werden, die ursprünglich nicht im übergeordneten VI enthalten waren.
- Verwenden Sie die Highlight-Funktion, wenn Sie die Daten auf ihrem Weg durch das Blockdiagramm beobachten möchten.

- Sie können sich in Einzelschritten durch das VI bewegen, um jede Aktion des VIs im Blockdiagramm anzuzeigen.
- Mit Hilfe des Sonden-Werkzeugs können Zwischenwerte ausgegeben werden, und es lässt sich die Fehlerausgabe von VIs und Funktionen überprüfen. Dies ist besonders für VIs nützlich, die Ein- und Ausgabeoperationen (I/O) durchführen.
- Mit Hilfe von Haltepunkten können Sie die Ausführung anhalten, um in Einzelschritten fortzufahren oder um Sonden einzusetzen.
- Die Ausführung eines VIs kann auch ausgesetzt werden, um Werte von Bedien- und Anzeigeelementen zu bearbeiten, die Anzahl der Durchläufe zu steuern oder zum Ausführungsbeginn zurückzukehren.
- Deaktivieren Sie einen Abschnitt des Blockdiagramms vorübergehend, um festzustellen, ob das VI ohne diesen Abschnitt bessere Leistungen zeigt.
- Stellen Sie dann sicher, dass keines der VIs und keine Funktion undefinierte Werte ausgibt. Dies ist oft bei numerischen Werten der Fall. So kann zum Beispiel an einer bestimmten Stelle im VI eine Division durch Null auftreten und deshalb an einem Datenausgang **`Inf`** (Unendlich) ausgegeben werden, was mit den nachfolgenden Funktionen und SubVIs zu Problemen führt, da diese definierte numerische Werte erwarten.
- Wenn das VI langsamer als erwartet ausgeführt wird, sollten Sie sich vergewissern, dass die Highlight-Funktion deaktiviert ist. Schließen Sie außerdem die Frontpanels und Blockdiagramme von SubVIs, wenn Sie diese nicht benötigen, da die Ausführungsgeschwindigkeit auch durch offene Fenster herabgesetzt werden kann.
- Um festzustellen, ob an einem der Bedien- und Anzeigeelemente ein Überlauf auftritt (zum Beispiel, wenn ein Fließkommawert in einen Integer-Wert oder ein Integer in einen kleineren Integer konvertiert wurde), überprüfen Sie die numerische Darstellung. So ist es zum Beispiel nicht möglich, einen 16-Bit-Integer mit einer Funktion zu verbinden, die nur für 8-Bit-Integer geeignet ist. Wenn die Konvertierung trotzdem durchgeführt wird, kann es zu Datenverlust kommen.
- Stellen Sie fest, ob in FOR-Schleifen die vorgesehenen Iterationen durchgeführt werden, da sonst leere Arrays erzeugt werden.
- Die Initialisierung der Schieberegister muss korrekt ausgeführt werden, damit Daten von den Schieberegistern Daten nicht von einer Schleifenausführung zur nächsten gespeichert werden.

- Überprüfen Sie die Reihenfolge der Cluster-Elemente an den Ausgangs- und Endpunkten. Zwar können in LabVIEW bei der Bearbeitung eines VIs uneinheitliche Datentypen und Cluster-Größen erkannt werden, jedoch keine Unstimmigkeiten zwischen Elementen des gleichen Typs.
- Überprüfen Sie die Ausführungsreihenfolge der Knoten.
- Stellen Sie sicher, dass das VI keine verborgenen SubVIs enthält. Das kann vorkommen, wenn Sie zwei SubVIs versehentlich übereinander positioniert oder eine Struktur verkleinert haben, ohne auf das entsprechende SubVI zu achten.
- Sie können aber auch die Anzahl der verwendeten SubVIs mit den Ergebnissen von **Durchsuchen»Die SubVIs dieses VIs** und **Durchsuchen»Ungeöffnete SubVIs** vergleichen, um festzustellen, ob noch weitere SubVIs vorhanden sind. Mit dem **Hierarchie**-Fenster können Sie ebenfalls alle SubVIs zu einem VI anzeigen lassen. Um falsche Ergebnisse aufgrund verborgener VIs zu vermeiden, geben Sie an, dass es sich obligatorisch zu verbindende VI-Eingänge handelt.

Highlight-Funktion

Über die Schaltfläche für die **Highlight-Funktion** (siehe Abbildung links), kann im Blockdiagramm eine Ausführungsanimation angezeigt werden. Durch die Highlight-Funktion wird mit Hilfe von Kreisen der Datenfluss von einem Knoten zum anderen im Blockdiagramm veranschaulicht. Verwenden Sie die Highlight-Funktion in Verbindung mit der Einzelschrittausführung, um zu sehen, wie sich die Daten von Knoten zu Knoten durch ein VI bewegen.

Hinweis Bei Verwendung der Highlight-Funktion wird die Ausführungsgeschwindigkeit des VIs erheblich herabgesetzt.

Wenn ein **Fehlerausgang** einen Fehler meldet, erscheint der Fehlercode rot eingerahmt neben dem **Fehlerausgang**. Wenn kein Fehler auftritt, erscheint am **Fehlerausgang** ein grün eingerahmtes `OK`. Weitere Hinweise zu Fehler-Clustern finden Sie im Abschnitt *Fehler-Cluster* dieses Kapitels.

Einzelschrittausführung

Wenn ein VI im Einzelschrittmodus ausgeführt wird, lässt sich jede Aktion des VIs im Blockdiagramm verfolgen. Mit den Einzelschrittschaltflächen lässt sich die Ausführung eines VIs oder SubVIs nur im Einzelschrittmodus beeinflussen. Im Ausführungsmodus sind diese Schaltflächen deaktiviert. Um in den Einzelschrittmodus zu wechseln, klicken Sie auf der Blockdia-

gramm-Symbolleiste entweder auf die Schaltfläche **Überspringen** oder **Hineinspringen**. Wenn Sie den Cursor über die Schaltfläche **Überspringen**, **Hineinspringen** oder **Herausspringen** bewegen, wird ein Hinweisstreifen zur Beschreibung des nächsten Schrittes angezeigt, der nach Anklicken dieser Schaltfläche erfolgen würde. Sie können SubVIs im Einzelschrittmodus oder im Normalmodus ablaufen lassen.

Wenn ein VI im Einzelschrittmodus ausgeführt wird und dabei die Highlight-Funktion aktiviert ist, wird auf den Symbolen der SubVIs, die aktuell ausgeführt werden, ein Ausführungssymbol sichtbar (siehe Abbildung links).

Sonden-Werkzeug

Mit dem Sonden-Werkzeug (siehe Abbildung links) können während der Ausführung eines VIs an Verbindungsstücken Zwischenwerte angezeigt werden. Das Sonden-Werkzeug eignet sich besonders für komplizierte Blockdiagramme mit einer Reihe von Operationen, von denen jede inkorrekte Daten zurückgeben könnte. Verwenden Sie das Sonden-Werkzeug zusammen mit der Highlight-Funktion, dem Einzelschrittmodus und Haltepunkten, um festzustellen, ob und wo unkorrekte Werte auftreten. Wenn Daten verfügbar sind, wird die Sonde unverzüglich während des Einzelschrittmodus aktualisiert oder wenn die Ausführung an einem Haltepunkt unterbrochen wurde. Wenn die Ausführung im Einzelschrittmodus oder an einem Haltepunkt an einem Knoten ausgesetzt wird, kann das Sonden-Werkzeug auch im gerade ausgeführten Verbindungsstrang eingesetzt werden, um anzuzeigen, welcher Wert an der entsprechenden Stelle gerade übergeben wurde.

Sondentypen

Mit Hilfe einer allgemeinen Sonde, einem Anzeigelement der Palette **Elemente**, einer voreingestellten, einer benutzerdefinierten oder einer selbsterstellten Sonde können den Verbindungen während der VI-Ausführung Zwischenwerte entnommen werden.

Allgemeine Sonden

Mit dem herkömmlichen Sonden-Werkzeug können Werte angezeigt werden, die über eine bestimmte Verbindung von einem Knoten an einen anderen übergeben werden. Um das allgemeine Sonden-Werkzeug zu verwenden, klicken Sie mit der rechten Maustaste auf ein Verbindungsstück und wählen Sie aus dem Kontextmenü die Option **Sonde anpassen»Allgemein** aus.

Mit diesem Werkzeug können die Zwischenwerte nur angezeigt werden. Es kann jedoch nicht auf die Werte reagiert werden.

Sofern für den entsprechenden Datentyp keine voreingestellte oder benutzerdefinierte Sonde angegeben wurde, wird bei einem Rechtsklick auf eine Verbindung und Auswahl der Option **Sonde** immer das allgemeine Sonden-Werkzeug verwendet.

Eine benutzerdefinierte Sonde kann wie ein VI auf Fehler untersucht werden. Allerdings kann eine Sonde weder für das eigene Blockdiagramm noch für eines seiner SubVIs verwendet werden. Um eine Sonde auf Fehler zu untersuchen verwenden Sie die allgemeine Sonde.

Anzeige von Werten über Anzeigeelemente

Die Werte, die über ein bestimmtes Verbindungsstück an einen Knoten übergeben werden, können auch mittels eines Anzeigeelements sichtbar gemacht werden. Für numerische Werte kann in der Sonde beispielsweise ein Diagramm eingesetzt werden. Um das entsprechende Anzeigeelement auszuwählen, klicken Sie einen Verbindungsstrang mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü **Benutzerdef. Sonde» Elemente** aus. Statt dessen kann aber auch über das Symbol **Ein Element auswählen** auf der Palette **Elemente** benutzerdefinierte Elemente oder Typendefinitionen ausgewählt werden, die auf dem Computer oder einem gemeinsam genutzten Verzeichnis eines Servers gespeichert sind. Typendefinitionen werden zur Anzeige von Probedaten in Sonden genauso eingesetzt wie benutzerdefinierte Elemente.

Damit ein Anzeigeelement in die Verbindung platziert wird, muss der entsprechende Datentyp mit dem des Verbindungsstücks übereinstimmen.

Voreingestellte Sonden

Voreingestellte Sonden sind VIs, die umfassende Informationen über die Werte ausgeben, die an einem bestimmten Verbindungsstück an einen Knoten übergeben werden. So gibt zum Beispiel das VI “Refnum-Sonde” den VI-Namen, -Pfad und den Hexadezimalwert der entsprechenden Referenz aus. Mit solchen Sonden kann auch auf das Zwischenergebnis an einem Verbindungsstück reagiert werden. So können zum Beispiel mit einer Fehlersonde oder einem Fehler-Cluster der Status, der Code, die Quelle und die Beschreibung eines Fehlers angezeigt werden und es kann festgelegt werden, ob beim Auftreten eines Fehlers oder einer Warnung ein bedingter Haltepunkt gesetzt werden soll.

Die voreingestellten Sonden werden immer am Anfang des Kontextmenüs **Benutzerdef. Sonde** angezeigt. Klicken Sie zur Auswahl einer voreingestellten Sonde mit der rechten Maustaste auf ein Verbindungsstück, und wählen Sie im Kontextmenü die Option **Benutzerdef. Sonde** aus.
Es werden immer nur die Sonden angezeigt, die zum jeweiligen Datentyp der Verbindung passen.

Ein zur Verwendung der in LabVIEW integrierten Sonden ist das VI "Using Supplied Probes" im Verzeichnis `examples\general\probes.llb`.

Benutzerdefinierte Sonden

Mit dem Assistenten zur Erstellung benutzerdefinierter Sonden können Sie Sonden erstellen oder vorhandene Sonden an Ihre Bedürfnisse anpassen. Zur Anzeige des Assistenten klicken Sie mit der rechten Maustaste auf ein Verbindungsstück und wählen im Kontextmenü **Benutzerdef. Sonde» Allgemein** aus. Wenn Sie mehr Einfluss auf die Entnahme von Zwischenwerten nehmen möchten, erstellen Sie eine neue Sonde. Der Datentyp der neuen Sonde stimmt immer mit dem der Verbindung überein, die mit der rechten Maustaste angeklickt wurde. Um eine erstellte Sonde zu bearbeiten, muss sie von dem Verzeichnis aus geöffnet werden, in dem sie gespeichert ist.

Nachdem entweder über das Kontextmenü **Benutzerdef. Sonde**, das Symbol **Ein Element auswählen** in der Palette oder über den Assistenten zur Erstellung benutzerdefinierter Sonden ein Probedatenwerkzeug ausgewählt bzw. erstellt wurde, wird dieses für den jeweiligen Datentyp als Voreinstellung verwendet und bei der Auswahl der Option **Sonde** im Kontextmenü geladen. Es wird immer die exakt zum jeweiligen Datentyp passende Sonde geladen. Das heißt, mit einer Sonde für Fließkommawerte mit doppelter Genauigkeit können beispielsweise keine vorzeichenlosen 32-Bit-Integerwerte entnommen werden, auch wenn eine Konvertierung dieser Werte in LabVIEW möglich wäre.

Hinweis Wenn eine benutzerdefinierte Sonde jedesmal als Standardsonde für einen bestimmten Datentyp verwendet werden soll, ist sie in das Verzeichnis `user.lib\_probes\default` zu speichern. Auf keinen Fall sollte die Sonde unter `vi.lib\_probes` gespeichert werden, denn diese Dateien werden bei einer Neuinstallation oder Software-Aktualisierung überschrieben.

Warnungen zur Erstellung und Verwendung von benutzerdefinierten Sonden befinden sich in der *LabVIEW-Hilfe*.

Haltepunkte

Das Haltepunkt-Werkzeug (siehe Abbildung links) dient dazu, in ein VI, einen Knoten oder ein Verbindungsstück im Blockdiagramm einen Haltepunkt zu platzieren, so dass die Ausführung an dieser Stelle angehalten wird. Wenn Sie einen Haltepunkt an einer Verbindung setzen, wird die Ausführung unterbrochen, wenn die Daten die Verbindung passiert haben. Um die Ausführung zu unterbrechen, nachdem alle Knoten im Blockdiagramm ausgeführt wurden, setzen Sie einen Haltepunkt in das Blockdiagramm.

Wenn die Ausführung eines VIs an einem Haltepunkt angehalten wird, bringt LabVIEW das Blockdiagramm in den Vordergrund und umrahmt die Stelle, an der der Haltepunkt gesetzt wurde. Wenn Sie den Cursor über einen bestehenden Haltepunkt bewegen, ändert sich die schwarze Fläche im Cursor des Haltepunkt-Werkzeugs auf Weiß.

Wenn das VI während der Ausführung an einen Haltepunkt gelangt, hält es an und die Schaltfläche **Pause** erscheint rot. Sie können nun folgendermaßen vorgehen:

- Bewegen Sie sich mit Hilfe der Einzelschrittschaltflächen in Einzelschritten durch die Ausführung.
- Sondieren Sie Verbindungen, um die Zwischenwerte zu prüfen.
- Ändern Sie die Werte von Bedienelementen im Frontpanel.
- Klicken Sie auf die Schaltfläche **Pause**, um die Ausführung bis zum nächsten Haltepunkt oder bis zum Ausführungsende des VIs fortzusetzen.

Die Haltepunkte werden mit dem entsprechenden VI gespeichert. Sie sind jedoch nur aktiv, wenn das VI ausgeführt wird. Zur Anzeige aller Haltepunkte wählen Sie **Durchsuchen»Haltepunkte**.

Aussetzen der Ausführung

Die Ausführung eines SubVIs kann unterbrochen werden, um Werte von Bedien- und Anzeigeelementen zu ändern, zum Ausführungsbeginn zurückzukehren oder um festzulegen, wie oft das SubVI vor der Rückkehr zum aufrufenden Objekt ausgeführt werden soll. Sie können veranlassen, dass alle Aufrufe eines SubVIs bei ausgesetzter Ausführung gestartet werden, oder Sie können einen bestimmten Aufruf eines SubVIs aussetzen.

Um alle Aufrufe eines SubVIs auszusetzen, öffnen Sie das SubVI und wählen **Ausführen»Bei Aufruf unterbrechen**. Die Ausführung des SubVIs wird automatisch ausgesetzt, wenn es von einem anderen VI auf-

gerufen wird. Wenn Sie dieses Menüelement im Einzelschrittmodus wählen, wird die Ausführung des SubVIs nicht sofort ausgesetzt, sondern erst, wenn es aufgerufen wird.

Um den Aufruf eines bestimmten SubVIs auszusetzen, klicken Sie im Blockdiagramm mit der rechten Maustaste auf den SubVI-Knoten und wählen aus dem Kontextmenü die Option **Einstellungen SubVI-Knoten**. Aktivieren Sie hier das Kontrollkästchen **Bei Aufruf unterbrechen**, um die Ausführung nur bei dieser Instanz des SubVIs auszusetzen.

Ob die Ausführung eines VIs angehalten oder ausgesetzt wurde, sehen Sie im Fenster **Hierarchie** unter **Durchsuchen»VI-Hierarchie anzeigen**. Ein Pfeil-Symbol zeigt ein VI an, das im regulären oder im Einzelschrittmodus ausgeführt wird. Ein angehaltenes oder ausgesetztes VI ist durch ein Symbol gekennzeichnet. Ein grünes Pause-Symbol oder ein ungefülltes Symbol in schwarzweiß zeigt ein VI an, dessen Ausführung beim Aufruf angehalten wird. Wenn die Ausführung eines VIs aktuell angehalten wurde, wird dies durch ein rotes Pause-Symbol oder ein ausgefülltes Symbol in schwarzweiß symbolisiert. Ein unterbrochenes VI wird durch ein Ausrufezeichen gekennzeichnet. Ein VI kann gleichzeitig angehalten und ausgesetzt werden.

Ermitteln der aktuellen Instanz eines SubVIs

Wenn die Ausführung eines SubVIs angehalten wird, werden im Pulldown-Menü **Aufrufkette** angefangen vom VI der obersten Ebene bis zum SubVI alle VIs aufgeführt, die eine Aufrufkette bilden. Hierbei handelt es sich jedoch nicht um dieselbe Liste wie unter **Durchsuchen»Aufrufende dieses VIs**, da darin alle aufrufenden VIs angezeigt werden, ungeachtet dessen, ob sie aktuell ausgeführt werden oder nicht. Verwenden Sie das Menü **Aufrufkette**, um die aktuelle Instanz des SubVIs zu ermitteln, wenn das Blockdiagramm mehr als eine Instanz enthält. Wenn Sie ein VI aus dem Menü **Aufrufkette** wählen, wird dessen Blockdiagramm geöffnet, und LabVIEW hebt das VI hervor, welches das aktuelle SubVI aufgerufen hat.

Vorübergehendes Deaktivieren von Abschnitten des Blockdiagramms

Ähnlich wie beim Deaktivieren oder Auskommentieren eines Programmabschnitts bei textbasierten Programmiersprachen können auch VIs mit deaktivierten Blockdiagrammabschnitten ausgeführt werden. Auf diese Weise lässt sich feststellen, ob das VI ohne den entsprechenden Abschnitt bessere Leistungen zeigt.

Setzen Sie den Abschnitt, den Sie deaktivieren möchten, in eine Case-Struktur, und verwenden Sie eine boolesche Konstante, um beide Cases auszuführen. Weitere Informationen zum Einsatz von Case-Strukturen finden Sie im Abschnitt *Case-Strukturen* des Kapitels 8, *Schleifen und Strukturen*. Sie können auch eine Kopie des VIs erstellen und in der Kopie Abschnitte aus dem Blockdiagramm löschen. Anschließend verwerfen Sie die Version des VIs, die Sie nicht verwenden möchten.

Deaktivieren der Werkzeuge zur Fehlersuche

Wenn Sie möchten, dass Ihr VI weniger Speicherplatz belegt oder seine Leistung geringfügig gesteigert werden soll, können Sie die Fehlersuche deaktivieren. Klicken Sie mit der rechten Maustaste auf das Anschlussfeld, und wählen Sie **VI-Einstellungen**. Wählen Sie dazu aus dem Pulldown-Menü **Kategorie** die Option **Ausführung** aus und deaktivieren Sie das Kontrollkästchen **Automatische Fehlerbehandlung aktivieren**.

Undefinierte oder unvorhergesehene Daten

Mit undefinierten Daten, die als `NaN` (Not a Number, keine Zahl) oder `Inf` (Infinity, Unendlich) ausgegeben werden, werden alle nachfolgenden Operationen ungültig. Bei fehlerhaften Berechnungen oder undefinierten Ergebnissen geben Fließkommaoperationen folgende symbolische Werte aus:

- `NaN` (keine Zahl) steht für einen Fließkommawert, der bei ungültigen Operationen ausgegeben wird, wie beispielsweise bei der Berechnung der Quadratwurzel aus einer negativen Zahl.
- `Inf` (Unendlich) steht für einen Fließkommawert, der bei Operationen wie beispielsweise der Division durch Null ausgegeben wird.

Ganzzahlige Werte werden nicht auf Bedingungen wie positiver oder negativer Überlauf geprüft. Bei Fließkommawerten gilt diesbezüglich IEEE 754, also der *Standard for Binary Floating-Point Arithmetic*.

Bei Fließkommaoperationen werden `NaN` und `Inf` getreulich weitergegeben. Bei der Umwandlung in Ganzzahlen oder boolesche Werte verlieren die `NaN` und `Inf` ihre Bedeutung. So lautet das Ergebnis einer Division von Eins durch Null beispielsweise `Inf`. Wird `Inf` jedoch in eine 16-Bit-Ganzzahl umgewandelt, wird der Wert `32767` ausgegeben, der wie ein normaler Wert aussieht. Weitere Informationen zur Umwandlung numerischer Werte finden Sie im Abschnitt *Numerische Konvertierungen* des Anhangs B, *Polymorphe Funktionen*.

Bevor Werte in Integer-Datentypen umgewandelt werden, verwenden Sie das Sonden-Werkzeug, um die als Zwischenergebnisse ausgegebenen Fließkommawerte auf Gültigkeit zu prüfen. Um das Ergebnis auf `NaN` zu prüfen, verbinden Sie die Vergleichsfunktion “Keine Zahl/Kein Pfad/Keine RefNum?” mit dem Wert, den Sie für möglicherweise ungültig halten.

Voreingestellte Werte in Schleifen

Wenn die Anzahl der Durchläufe auf Null gestellt wird, geben FOR-Schleifen voreingestellte Werte aus.

Weitere Informationen zu voreingestellten Werten für bestimmte Datentypen finden Sie im Abschnitt *Datentypen für Bedien- und Anzeigeelemente* des Kapitels 5, *Erstellen des Blockdiagramms*.

FOR-Schleifen

In FOR-Schleifen werden immer dann unerwartete Werte erzeugt, wenn für die Anzahl der Durchläufe `0` angegeben wird oder wenn ein Eingang mit aktivierter Auto-Indizierung mit einem leeren Array verbunden wird. In diesem Fall wird die Schleife nicht ausgeführt, und für alle nicht auto-indizierten Ausgabetunnel werden die Standardwerte für den Tunnel-Datentyp ausgegeben. Mit Hilfe von Schieberegistern können Werte innerhalb einer Schleife übergeben werden, ungeachtet dessen, ob die Schleife ausgeführt wird oder nicht.

Nähere Hinweise zu FOR-Schleifen, Schieberegistern und zur Auto-Indizierung entnehmen Sie bitte dem Abschnitt *FOR- und While-Schleifenstrukturen* des Kapitels 8, *Schleifen und Strukturen*.

Standardwerte für Arrays

Wenn für ein Array ein Index verwendet wird, der außerhalb der Array-Grenzen liegt, wird für die Array-Elemente der voreingestellte Wert ausgegeben. Sie können die Funktion “Array-Größe” verwenden, um die Größe des Arrays zu ermitteln. Weitere Informationen zu Arrays finden Sie in Abschnitt *Arrays* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*. Weitere Informationen zur Indizierung finden Sie im Abschnitt *Auto-Indizierung von Schleifen* des Kapitels 8, *Schleifen und Strukturen*. Dass Sie ein Array versehentlich über seine Grenzen hinaus indizieren, kann entweder vorkommen, wenn Sie dazu eine While-Schleife verwenden oder wenn Sie an den **Index**-Eingang der Funktion “Array indizieren” einen zu großen Wert oder ein leeres Array übergeben.

Verhindern undefinierter Werte

Um festzustellen, ob ein VI undefinierte Daten ausgibt, sollten Sie sich nicht allein auf spezielle Werte wie `NaN`, `Inf` oder leere Arrays verlassen. Vergewissern Sie sich stattdessen, dass das VI definierte Daten produziert, indem Sie veranlassen, dass das VI einen Fehler zurückgibt, wenn eine Situation eintritt, in der wahrscheinlich undefinierte Daten ausgegeben werden.

Wenn Sie beispielsweise ein VI erstellen, das ein eingehendes Array für die Auto-Indizierung einer FOR-Schleife verwendet, legen Sie fest, wie das VI weiter verfahren soll, wenn das Eingabe-Array leer ist. In diesem Fall kann entweder ein Fehlercode ausgegeben werden oder die Ausgabewerte der Schleife können durch definierte Werte erstetzt werden.

Fehlerprüfung und Fehlerbehandlung

Jedem Fehler ist ein numerischer Code und eine entsprechende Fehlermeldung zugeordnet. Per Grundeinstellung reagiert LabVIEW automatisch auf Fehler während der Ausführung eines VIs. Das heißt, die Ausführung wird angehalten, das SubVI oder die Funktion, in der der Fehler auftrat, wird hervorgehoben, und es wird das **Fehler**-Dialogfeld angezeigt.

Um diese Einstellung zu deaktivieren, wählen Sie **Datei»VI-Einstellungen**, und klicken Sie im Pulldown-Menü **Kategorie** auf **Ausführung**. Wenn die automatische Bearbeitung von Fehlern auch für neue VIs deaktiviert werden soll, lässt sich die entsprechende Einstellung unter **Werkzeuge»Optionen** und anschließender Auswahl der Option **Blockdiagramm** aus dem Pulldown-Menü vornehmen.

Damit alle von SubVIs oder Funktionen gefundenen Fehler ignoriert werden, verbinden Sie den **Fehler (Ausgang)** des VIs bzw. der Funktion mit dem **Fehler (Eingang)** des VIs “Fehler löschen”. Um bei einem einzelnen VI oder einer einzelnen Funktion die automatische Fehlerbearbeitung zu deaktivieren, verbinden Sie den entsprechenden Parameter **Fehler (Ausgang)** mit dem **Fehler (Eingang)** eines anderen SubVIs bzw. einer anderen Funktion oder mit einem Element des Typs **Fehlerausgang**.

Verwenden Sie die VIs, Funktionen und Parameter zur Fehlerbehandlung in VIs, die Ihnen in LabVIEW zur Verfügung stehen. Wenn LabVIEW einen Fehler feststellt, können Sie beispielsweise ein Dialogfeld mit einer Fehlermeldung anzeigen. Verwenden Sie die Fehlerbehandlung in Verbindung mit den Fehlersuche-Werkzeugen, um Fehler zu suchen und zu bearbeiten. Der Einsatz der Fehlerbearbeitung wird dringend empfohlen.

Prüfen auf Fehler

Ganz gleich, wie viel Vertrauen Sie in die von Ihnen erstellten VIs setzen, Sie können nicht jedes Problem vorhersehen, auf das ein Benutzer möglicherweise stößt. Ohne einen Mechanismus zur Fehlerüberprüfung wissen Sie nur, dass das VI nicht ordnungsgemäß funktioniert. Die Fehlerprüfung zeigt Ihnen, warum und wo Fehler auftreten.

Wenn Sie irgendeine Art der Ein- oder Ausgabe (I/O) durchführen, ziehen Sie die Möglichkeit in Betracht, dass Fehler auftreten können. Beinahe alle I/O-Funktionen geben Fehlerinformationen zurück. Versehen Sie VIs mit Fehlerprüfmechanismen, insbesondere bei I/O-Operationen (Datei, seriell, Instrumentierung, Datenerfassung und Kommunikation), und stellen Sie ein Verfahren für eine geeignete Fehlerbehandlung bereit.

Die Fehlerprüfung in VIs kann Ihnen helfen, die folgenden Probleme zu erkennen:

- Sie haben Kommunikationskanäle falsch initialisiert oder es wurden inkorrekte Daten an ein externes Gerät geschrieben.
- Ein externes Gerät ist stromlos, defekt oder arbeitet nicht korrekt.
- Sie haben die Betriebssystemsoftware aktualisiert, wodurch der Pfad zu einer Datei oder die Funktionalität eines VIs oder einer Bibliothek geändert wurde beziehungsweise beeinträchtigt wird. Unter Umständen liegt ein Problem in einem VI oder einem Systemprogramm vor.

Fehlerbehandlung

Standardmäßig reagiert LabVIEW automatisch auf Fehler, indem es die Ausführung eines VIs anhält. Jedoch sind auch andere Methoden der Fehlerbearbeitung möglich. Wenn bei einem I/O-VI in einem Blockdiagramm beispielsweise eine Zeitüberschreitung auftritt, möchten Sie vielleicht nicht, dass die gesamte Applikation gestoppt wird, sondern das VI soll möglicherweise für einen bestimmten Zeitraum erneute Versuche unternehmen. In LabVIEW können Sie diese Fehlerbehandlungsentscheidungen im Blockdiagramm des VIs treffen.

Fehler werden von VIs und Funktionen entweder als numerische Fehlercodes oder über einen Fehler-Cluster ausgegeben. Normalerweise arbeiten Funktionen mit numerischen Fehlercodes, und VIs verwenden einen Fehler-Cluster, im Normalfall mit Fehlerein- und ausgängen. Weitere Hinweise zu Fehler-Clustern finden Sie im Abschnitt *Fehler-Cluster* dieses Kapitels.

Die Fehlerbehandlung in LabVIEW folgt dem Datenflussmodell. Ebenso wie die Daten durch ein VI fließen, können auch Fehlerinformationen wei-

tergegeben werden. Verbinden Sie die Fehlerinformationen vom Anfang des VIs bis zu dessen Ende. Fügen Sie am Ende des VIs ein Fehlerbehandler-VI ein, um zu ermitteln, ob das VI fehlerlos ausgeführt werden konnte. Verwenden Sie in jedem von Ihnen verwendeten oder erstellten VI die Cluster **Fehler (Eingang)** und **Fehler (Ausgang)**, um Fehlerinformationen durch das VI zu leiten.

Wenn das VI ausgeführt wird, prüft LabVIEW an jedem Ausführungsknoten auf Fehler. Wenn LabVIEW keine Fehler findet, wird der Knoten normal ausgeführt. Ansonsten wird der Fehler von einem Knoten an den nächsten übergeben, und der betroffene Teil der Funktion oder des VIs wird nicht ausgeführt. Der nächste Knoten geht ebenso vor und so weiter. Am Ende des Ausführungsflusses gibt LabVIEW den Fehler zurück.

Fehler-Cluster

In den Clustern **Fehler (Eingang)** und **Fehler (Ausgang)** sind folgende Daten enthalten:

- **Status** ist ein boolescher Wert, der TRUE zurückgibt, wenn ein Fehler aufgetreten ist. Die meisten VIs, Funktionen und Strukturen, die boolesche Daten akzeptieren, können auch diesen Parameter erkennen. Sie können einen Fehler-Cluster zum Beispiel mit dem booleschen Eingang der Stopp-, LabVIEW beenden- oder der Auswahlfunktion verbinden. Wenn ein Fehler auftritt, gibt das Fehler-Cluster ein TRUE zurück.
- **Code** ist ein 32-Bit-Integer mit Vorzeichen, mit dem der Fehler numerisch gekennzeichnet ist. Ein Fehlercode, der nicht Null ist, gekoppelt mit dem **Status** FALSE signalisiert eine Warnung und keinen schwerwiegenden Fehler.
- **Quelle** ist ein String, der angibt, an welcher Stelle der Fehler aufgetreten ist.

Verwenden von While-Schleifen für die Fehlerbearbeitung

Um eine While-Schleife im Fehlerfall zu stoppen, verbinden Sie mit Bedingungsanschluss der Schleife einen Fehler-Cluster. Wenn Sie den Fehler-Cluster mit dem Bedingungsanschluss verbinden, wird nur der Wert TRUE oder FALSE des Parameters **Status** des Fehler-Clusters an den Anschluss übergeben. Wenn ein Fehler auftritt, wird die While-Schleife gestoppt.

Wenn ein Fehler-Cluster mit dem Bedingungsanschluss verbunden ist, ändern sich die Kontextmenüelemente **Stopp wenn TRUE** und **Weiter wenn TRUE** in **Bei Fehler stoppen** und **Bei Fehler fortfahren**.

Fehlerbearbeitung mit Case-Strukturen

Wenn Sie einen Fehler-Cluster mit dem Selektoranschluss einer Case-Struktur verbinden, zeigt die Selektor-Kennung zwei Cases an, und zwar `Fehler` und `Kein Fehler`. Außerdem ändert sich die Rahmenfarbe: der `Fehler`-Case wird rot und der Case `Kein Fehler` wird grün dargestellt. Im Falle eines Fehlers wird der `Fehler`-Case ausgeführt. Weitere Informationen zum Einsatz von Case-Strukturen finden Sie in Abschnitt *Case-Strukturen* des Kapitels 8, *Schleifen und Strukturen*.

7

Erstellen von VIs und SubVIs

Nachdem Sie nun erfahren haben, wie ein Frontpanel und ein Blockdiagramm erzeugt wird, können Sie eigene VIs und SubVIs erstellen und können VIs, eigenständige Applikationen und gemeinsam genutzte Bibliotheken für andere Benutzer bereitstellen.

Weitere Informationen zum Planen eines Projekts einschließlich Informationen zu häufig auftretenden Problemstellungen bei der Entwicklung sowie zu den Werkzeugen, die Sie zum Entwickeln eines Projekts verwenden können, finden Sie im Handbuch *LabVIEW Development Guidelines*.

Weitere Informationen ...

Weitere Informationen zur Erstellung und Verwendung von SubVIs, zum Speichern von VIs sowie zum Erstellen von eigenständigen Applikationen und gemeinsamen Bibliotheken finden Sie in der *LabVIEW-Hilfe*.

Planen und Entwerfen eines Projekts

Bevor Sie eigene VIs entwickeln, erstellen Sie eine Liste der Aufgaben, welche die Benutzer ausführen müssen. Legen Sie die Benutzerschnittstellenkomponenten und die Anzahl und Art der Bedien- und Anzeigeelemente fest, die Sie für die Datenanalyse, das Anzeigen der Analyseergebnisse und so weiter benötigen. Besprechen Sie mit den zukünftigen Benutzern oder anderen Mitgliedern des Projektteams, wie und wann der Benutzer auf die Funktionen und Leistungsmerkmale zugreifen können muss. Erstellen Sie Beispiel-Frontpanels, und zeigen Sie diese den zukünftigen Benutzern oder den Mitgliedern des Projektteams, um festzustellen, ob das Frontpanel den Benutzern dabei hilft, ihre Aufgaben zu erledigen. Nutzen Sie diesen interaktiven Prozess, um die Benutzeroberfläche gegebenenfalls zu verfeinern.

Teilen Sie die Applikation in logische Teile handhabbarer Größe auf. Beginnen Sie mit einem einfachen Blockdiagramm, das die Hauptbestandteile Ihrer Applikation enthält. Beispielsweise könnte ein Blockdiagramm jeweils einen Block für die Konfiguration, die Datenerfassung, die Analyse der erfassten Daten, die Anzeige der Analyseergebnisse, das Speichern der Daten auf Datenträger und für die Fehlerbehandlung enthalten.

Nachdem Sie das Blockdiagramm grob gegliedert haben, definieren Sie die Ein- und Ausgänge. Gestalten Sie dann die SubVIs, aus denen die Hauptkomponenten Ihres Blockdiagramms bestehen. Durch SubVIs wird dieses einfacher zu verstehen und zu handhaben. Auch die Fehlersuche wird auf diese Weise erleichtert. Sie können auch SubVIs für allgemeine oder häufige Operationen erstellen, die wiederverwendet werden können. Testen Sie Ihre SubVIs, nachdem Sie sie erstellt haben. Sie können zwar auch Testroutinen für die höhere Ebene erstellen, jedoch ist das Auffinden von Fehlern in kleinen Modulen einfacher als das Testen einer Hierarchie aus mehreren VIs. Möglicherweise stellen Sie auch fest, dass der anfängliche Entwurf des grob gegliederten Blockdiagramms unvollständig ist. Die Verwendung von SubVIs für das Ausführen von Einzelaufgaben vereinfacht das Ändern oder Neuordnen der Applikation. Weitere Informationen zu SubVIs finden Sie in Abschnitt *SubVIs* dieses Kapitels.

Unter **Hilfe»Beispiele finden** können Sie nach Beispielen für Blockdiagramme und SubVIs suchen.

Entwerfen von Projekten mit mehreren Entwicklern

Wenn mehrere Entwickler am gleichen Projekt arbeiten, müssen im Vorfeld die Verantwortlichkeiten für die Programmierung, die Schnittstellen sowie Kodierungsformate festgelegt werden, um einen reibungslosen Ablauf des Entwicklungsprozesses und ein problemloses Funktionieren der Applikation sicherzustellen. Nähere Hinweise zu Kodierungsformaten finden Sie in den *LabVIEW Development Guidelines* im Abschnitt 6, *LabVIEW Style Guide*.

Speichern Sie Masterkopien der Projekt-VIs auf einem einzigen Computer, und führen Sie Richtlinien für die Quellcodekontrolle ein. Ziehen Sie die Verwendung des LabVIEW Professional Development Systems in Betracht, welches Quellcode-Kontrollwerkzeuge enthält, welche die gemeinsame Nutzung von Dateien vereinfachen. Zu diesen Werkzeugen gehört auch ein Dienstprogramm zum Vergleich von VIs und zur Anzeige der Unterschiede zwischen unterschiedlichen VI-Versionen. Weitere Informationen zum Einsatz der Quellcodeverwaltung finden Sie in den *LabVIEW Development Guidelines* im Abschnitt *Source Code Control* des Kapitels 2, *Incorporating Quality into the Development Process*.

VI-Vorlagen

Zu den in LabVIEW verfügbaren VI-Vorlagen gehören SubVIs, Funktionen, Strukturen und Frontpanel-Objekte. Sie dienen zur Erstellung von VIs für allgemeine Anwendungen der Messtechnik und sollen Ihnen den Einstieg in die LabVIEW-Programmierung erleichtern. Vorlagen werden immer unbenannt geöffnet und müssen daher zunächst benannt und gespeichert werden. Das Dialogfeld **Neu**, in dem sich auch die Vorlagen befinden, ist über die Option **Datei»Neu** zu erreichen. Es kann aber auch über die Option **Neu** das Pulldown-Menüs **Neu** im Startdialogfenster von **LabVIEW** geöffnet werden.

Wenn Sie eine VI-Vorlage als SubVI verwenden, wird der Anwender vor dem Schließen zum Speichern des VIs aufgefordert.

Erstellen von VI-Vorlagen

Die Erstellung einer benutzerspezifischen VI-Vorlage empfiehlt sich, wenn mehrere ähnliche Operationen ausgeführt werden sollen, da so nicht immer wieder dieselben Komponenten auf das Frontpanel platziert werden müssen. Dazu ist einfach ein VI zu erstellen und als Vorlage abzuspeichern.

Andere Arten von Dokumenten

Die Erstellung von globalen Variablen, benutzerdefinierten Bedienelementen, Laufzeitmenüs, polymorphen VIs oder von Vorlagen zu globalen Variablen und Bedienelementen ist im Dialogfeld **Neu** in der Liste **Neu erstellen** unter **Andere Arten von Dokumenten** möglich.

Verwendung der in LabVIEW integrierten VIs und Funktionen

LabVIEW enthält VIs und Funktionen, mit denen spezifische Applikationen wie beispielsweise Datenerfassungs-VIs und -Funktionen oder VIs, die auf andere VIs zugreifen bzw. mit anderen Applikationen kommunizieren, erstellt werden können. Diese VIs können Sie als SubVIs in Ihrer Applikation verwenden, um Entwicklungszeit einzusparen. Weitere Informationen zu SubVIs finden Sie in Abschnitt *SubVIs* dieses Kapitels.

Erstellen von Instrumentensteuerungs- und Datenerfassungs-VIs und -Funktionen

In LabVIEW sind viele hundert Beispiel-VIs verfügbar, die Sie in Ihre eigenen VIs integrieren können. Sie können diese Beispiele jedoch auch für Ihre Zwecke verändern oder Teile davon kopieren und in Ihre VIs einfügen.

Sie können die integrierten VIs und Funktionen verwenden, um externe Instrumente wie beispielsweise Oszilloskope zu steuern, und um Daten zu erfassen, wie beispielsweise die Ablesewerte eines Thermoelements.

Die VIs zur Instrumenten-I/O dienen zur Steuerung externer Messgeräte. Zum Steuern von Instrumenten mit LabVIEW muss die geeignete Hardware installiert und eingeschaltet sein und an Ihrem Computer betrieben werden können. Die VIs und Funktionen, die Sie zum Steuern von Instrumenten verwenden, sind abhängig von den Kommunikationsprotokollen für die Instrumentierung, die seitens Ihrer Hardware unterstützt werden. Weitere Informationen zum Erstellen von VIs zum Steuern von Instrumenten finden Sie im *LabVIEW Measurements Manual*.

Mit Hilfe der VIs und Funktionen zur Datenerfassung lassen sich Messwerte von DAQ-Geräten erfassen. Um diese VIs verwenden zu können, muss die NI-DAQ-Treibersoftware sowie DAQ-Hardware installiert sein. Weitere Informationen zum Installieren der NI-DAQ-Treibersoftware und von DAQ-Hardware sowie zum Erstellen von VIs für die Datenerfassung finden Sie im *LabVIEW Measurements Manual*. Nachdem Sie die gewünschten Daten erfasst haben, können Sie die integrierten Analyse-, Berichterstellungs- und Mathematik-VIs und -Funktionen verwenden, um die Daten zu analysieren, um Berichte zu erstellen und um mathematische Operationen mit diesen Daten vorzunehmen. Informationen über Analysen und mathematische Begriffe in LabVIEW finden Sie im Handbuch *LabVIEW Analysis Concepts*.

Erstellen von VIs, die auf andere VIs zugreifen

Mit den VIs und Funktionen zur Applikationssteuerung kann festgelegt werden, wie sich VIs verhalten sollen, wenn Sie als SubVIs aufgerufen werden oder von einem Anwender gestartet werden. Sie können diese VIs und Funktionen verwenden, um mehrere VIs gleichzeitig zu konfigurieren. Wenn Sie gemeinsam mit anderen LabVIEW-Benutzern in einem Netzwerk arbeiten, können Sie diese VIs und Funktionen auch verwenden, um von einem fernen Standort aus auf VIs zuzugreifen und diese zu steuern. Weitere Informationen zur Steuerung von VIs über das Netzwerk finden Sie im Kapitel 17, *Programmatische Steuerung von VIs*.

Erstellen von VIs, die mit anderen Applikationen kommunizieren

Mit den VIs und Funktionen zur Datei-I/O können Daten zwischen VIs und anderen Anwendungen, wie zum Beispiel Microsoft Excel, ausgetauscht werden. Sie können diese VIs und Funktionen verwenden, um Berichte zu erstellen oder um Daten aus einer anderen Applikation in das VI aufzunehmen. Weitere Informationen zum Datenaustausch mit anderen Applikationen finden Sie in Kapitel 14, *Datei-I/O*.

Mit Hilfe der Kommunikations-VIs und -Funktionen können LabVIEW-Daten mit Hilfe eines Kommunikationsprotokolls wie beispielsweise FTP über das Web übertragen und Client-/Server-Applikationen erstellt werden, die Kommunikationsprotokolle verwenden. Weitere Informationen zur Kommunikation mit anderen Applikationen im Netzwerk oder im Web finden Sie in Kapitel 18, *Arbeiten mit LabVIEW im Netzwerk*.

(Windows) Setzen Sie die ActiveX-Funktionen zum Einfügen von ActiveX-Objekten in VIs und zur Steuerung ActiveX-fähiger Applikationen ein. Weitere Informationen zum Einsatz der ActiveX-Technologie finden Sie in Kapitel 19, *Windows-Konnektivität*.

SubVIs

Nachdem Sie ein VI erstellt und dessen Symbol und Anschlussfeld erzeugt haben, können Sie es in einem anderen VI verwenden. Ein VI, das im Blockdiagramm eines anderen VIs aufgerufen wird, nennt man SubVI. Ein SubVI entspricht in textbasierten Programmiersprachen einem Unterprogramm. Ein SubVI-Knoten entspricht einem Subroutinenaufruf in textbasierten Programmiersprachen. Der SubVI-Knoten ist nicht das eigentliche SubVI, ebenso wenig wie die Aufrufanweisung für eine Subroutine in einem Programm mit der Subroutine selbst zu verwechseln ist. Ein Blockdiagramm mit mehreren identischen SubVI-Knoten ruft dasselbe SubVI mehrere Male auf.

Die Bedien- und Anzeigeelemente eines SubVIs empfangen Daten vom und geben Daten an das Blockdiagramm des aufrufenden VIs zurück. Klikken Sie in der Palette **Funktionen** auf das Symbol **VI auswählen**, wählen Sie ein VI aus, klicken es doppelt an und platzieren es in das Blockdiagramm, damit es als SubVI aufgerufen wird.

Hinweis Bevor Sie ein VI als SubVI verwenden können, müssen Sie dafür ein Anschlussfeld einrichten. Weitere Informationen zum Einrichten eines Anschlussfeldes finden Sie in Abschnitt *Anschlussfeld einrichten* dieses Kapitels.

Zur Bearbeitung eines SubVIs, das sich auf dem Blockdiagramm befindet, verwenden Sie das Bedien- oder Positionierwerkzeug und klicken es im Blockdiagramm doppelt an. Wenn Sie das SubVI speichern, wirken sich die Änderungen auf alle Aufrufe des SubVIs und nicht mehr nur auf die aktuelle Instanz aus.

Beim Ausführen eines SubVIs, wird dessen Frontpanel normalerweise nicht angezeigt. Wenn Sie möchten, dass bei einem Aufruf das Frontpanel einer einzelnen Instanz des SubVIs geöffnet wird, klicken Sie mit der rechten Maustaste auf das SubVI und wählen Sie im Kontextmenü **SubVI-Einstellungen** aus. Wenn Sie möchten, dass bei einem Aufruf die Frontpanel sämtlicher Instanzen des SubVIs geöffnet werden, wählen Sie **Datei»VI-Einstellungen**, klicken anschließend im Pulldown-Menü **Kategorie** auf **Fenstererscheinungsbild** und wählen dann die Option **Anpassen** aus.

Sich häufig wiederholende Vorgänge

Manche Operationen in einem VI werden häufig ausgeführt. In diesem Fall empfiehlt es sich, SubVIs und Schleifen einzusetzen. Im Beispiel in Abbildung 7-1 sehen Sie zwei gleiche Operationen.

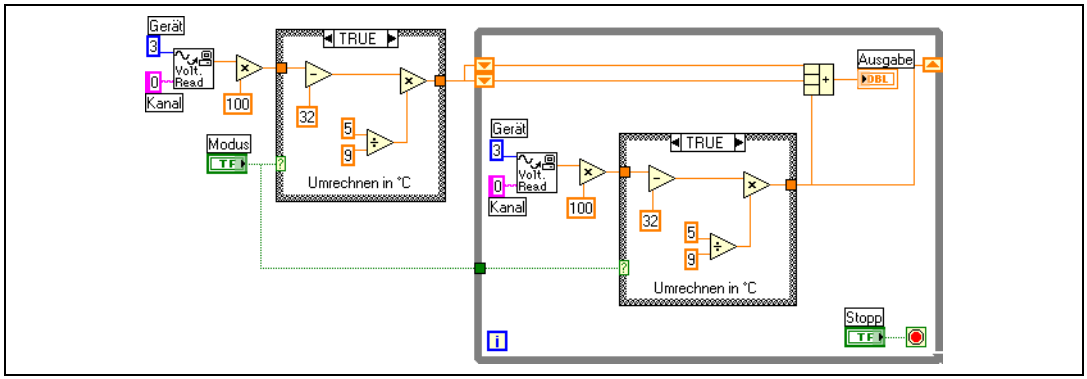


Abbildung 7-1. Blockdiagramm mit zwei gleichen Operationen

Um dieses VI zu vereinfachen, erstellen Sie, wie im Blockdiagramm in Abbildung 7-2, für diese Operation ein SubVI und rufen Sie dieses zwei Mal auf.

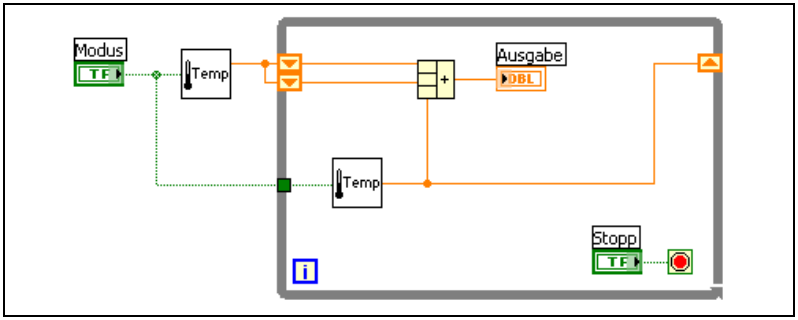


Abbildung 7-2. Zweimaliges Aufrufen eines SubVI

SubVIs können auch in anderen VIs wiederverwendet werden. Weitere Informationen zum Einsatz von Schleifen zum Kombinieren häufig auszuführender Operationen finden Sie im Kapitel 8, *Schleifen und Strukturen*.

Anschlussfeld einrichten

Um ein VI als SubVI verwenden zu können, müssen Sie ein Anschlussfeld wie links gezeigt erstellen. Das Anschlussfeld ist eine Sammlung von Anschlüssen, die den Bedien- und Anzeigeelementen dieses VIs entsprechen, ähnlich der Parameterliste eines Funktionsaufrufs in textbasierten Programmiersprachen. Mit dem Anschlussfeld werden die Eingänge und Ausgänge definiert, die mit dem VI verbunden werden sollen, damit Sie es als SubVI einsetzen können. Weitere Informationen zu Anschlussfeldern finden Sie in Abschnitt *Symbol- und Anschlussfeld* des Kapitels 2, *Einführung in die Welt der virtuellen Instrumente*.

Sie definieren Verbindungen, indem Sie jedem Anschluss im Anschlussfeld ein Frontpanel-Bedien- oder Anzeigeelement zuweisen. Um zu einem VI ein Anschlussfeld festzulegen, klicken Sie mit der rechten Maustaste auf des Symbol in der oberen rechten Ecke des Frontpanel-Fensters, und wählen Sie aus dem Kontextmenü die Option **Anschluss anzeigen** aus. Statt des VI-Symbols wird jetzt das Anschlussfeld des VIs angezeigt. Jedes Rechteck im Anschlussfeld entspricht einem Anschluss. Verwenden Sie die Rechtecke, um Eingänge und Ausgänge zuzuweisen. Die Anzahl der von LabVIEW im Anschlussfeld angezeigten Anschlüsse ist abhängig von der Anzahl der Bedien- und Anzeigeelemente auf dem Frontpanel.

Ein Anschlussfeld kann über maximal 28 Anschlüsse verfügen. Wenn das Frontpanel mehr als 28 Bedien- und Anzeigeelemente enthält, die programmgesteuert verwendet werden sollen, fassen Sie einige hiervon als Cluster zusammen und weisen diesem Cluster einen Anschluss des

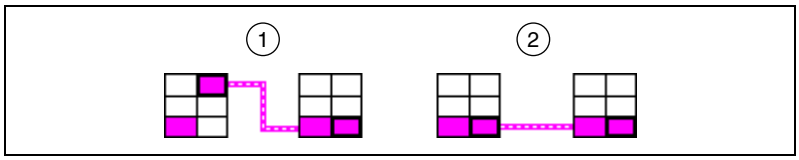
Anschlussfeldes zu. Weitere Informationen zum Gruppieren von Daten mit Hilfe von Clustern finden Sie im Abschnitt *Cluster* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Hinweis Wenn einem VI mehr als 16 Anschlüsse zugewiesen werden, ist es unter Umständen nicht mehr übersichtlich oder bedienfreundlich genug.

Zur Auswahl eines anderen Anschlussmusters für ein VI klicken Sie mit der rechten Maustaste auf das Anschlussfeld, und wählen Sie im Kontextmenü die Option **Anordnung der Anschlüsse** aus. Wählen Sie ein Anschlussfeldmuster mit zusätzlichen Anschlüssen. Sie können die zusätzlichen Anschlüsse unverbunden lassen, bis Sie diese benötigen. Aufgrund dieser Flexibilität sind Sie in der Lage, Änderungen mit minimalen Auswirkungen auf die Hierarchie der VIs vorzunehmen.

Wenn Sie eine Gruppe aus SubVIs erstellen, die Sie häufig zusammen einsetzen, weisen Sie den SubVIs ein konsistentes Anschlussfeld mit allgemeinen Eingängen an der gleichen Position zu, damit Sie sich besser daran erinnern, wo jeder Eingang zu finden ist. Wenn Sie ein SubVI erstellen, das eine Ausgabe produziert, die ein anderes SubVI als Eingabe verwendet, richten Sie die Eingabe- und Ausgabeverbindungen aufeinander aus, um die Verbindungsmuster zu vereinfachen. Setzen Sie den Cluster **Fehlereingang** in die untere linke und **Fehlerausgang** in die untere rechte Ecke des Frontpanels.

In der Abbildung 7-3 sehen Sie ein Beispiel für einen ordnungsgemäß und einen nicht ordnungsgemäß ausgerichteten Fehler-Cluster.



1 Nicht ordnungsgemäß 2 Ordnungsgemäß

Abbildung 7-3. Ausrichtung von Fehler-Clustern

Nähere Hinweise zur Einteilung des Anschlussfeldes finden Sie in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, im Abschnitt *Connector Panes*.

Festlegen von erforderlichen, empfohlenen und optionalen Ein- und Ausgängen

Bei der Erstellung eines VIs, das als SubVI weiterverwendet werden soll, kann festgelegt werden, welche Ein- und Ausgänge verbunden werden müssen, sollten oder können. Dadurch ist es weniger wahrscheinlich, dass Anschlüsse beim Verbinden übersehen werden.

Klicken Sie mit der rechten Maustaste auf einen Anschluss im Anschlussfeld und wählen Sie aus dem Kontextmenü **Diese Verbindung ist**. Die aktuelle Anschlusseinstellung ist mit einem Häkchen gekennzeichnet. Nun kann **erforderlich**, **empfohlen** oder **optional** ausgewählt werden.

Wenn Anschlüsse, die obligatorisch zu verbinden sind, offen bleiben, kann das VI nicht ausgeführt werden. Bei Ausgängen gibt es keine obligatorisch zu verbindenden Anschlüsse. Bei Eingangs- und Ausgangsanschlüssen bedeutet “empfohlen” oder “optional”, dass das Blockdiagramm, in dem sich das SubVI befindet, auch ausgeführt werden kann, wenn diese Anschlüsse nicht verbunden sind. In diesem Fall werden auch keine Warnungen ausgegeben.

Ein- und Ausgänge von VIs in `vi.lib` sind bereits als **Erforderlich**, **Empfohlen** oder **Optional** gekennzeichnet. LabVIEW setzt die Ein- und Ausgänge von VIs, die Sie erstellen, standardmäßig auf **Empfohlen**. Setzen Sie die Einstellung für einen Anschluss nur dann auf “Erforderlich”, wenn das VI über einen Ein- oder Ausgang verfügen muss, um ordnungsgemäß zu funktionieren.

In der **Kontext-Hilfe** erscheint der Parametername bei obligatorisch zu verbindenden Anschlüssen in fetter Schrift, bei empfohlenen in normaler und bei optionalen in grauer Schrift. Wenn Sie in der **Kontext-Hilfe** auf die Schaltfläche **Optionale Anschlüsse und kompletten Pfad verbergen** klicken, werden die Parameternamen der optional zu verbindenden Anschlüsse nicht angezeigt.

VI-Symbole erstellen

Zu jedem VI gehört ein Symbol, das in der rechten oberen Ecke des Frontpanels und Blockdiagramms angezeigt wird. Es dient zur grafischen Darstellung eines VIs und kann eine Kombination aus Text- und Grafikelementen enthalten. Wenn Sie ein VI als SubVI verwenden, dient dieses Symbol zur dessen Kennzeichnung im Blockdiagramm des VIs.

Das Standardsymbol enthält eine Zahl, die angibt, wie viele neue VIs Sie seit dem Start von LabVIEW geöffnet haben. Sie erstellen benutzerdefi-

nierte Symbole als Ersatz für das Standardsymbol, indem Sie mit der rechten Maustaste auf das Symbol in der oberen rechten Ecke des Frontpanels oder des Blockdiagramms klicken und aus dem Kontextmenü **Symbol bearbeiten** wählen oder das Symbol in der oberen rechten Ecke des Frontpanels doppelt anklicken.

Es kann auch eine Grafik in einem Verzeichnis ausgewählt werden und mit der Drag-and-Drop-Funktion in die rechte obere Ecke des Frontpanels oder Blockdiagramms gezogen werden. LabVIEW konvertiert die Grafik in ein 32×32-Pixel-Symbol.

Abhängig vom verwendeten Monitortyp können Sie jeweils ein separates Symbol für die Modi Schwarz/Weiß, 16 Farben oder 256 Farben erstellen. LabVIEW verwendet beim Drucken einfarbige Symbole, sofern Sie nicht über einen Farbdrucker verfügen.

Für nähere Informationen zur Erstellung eines Symbols lesen Sie in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter *Icons* nach.

Darstellung von SubVIs und Express-VIs in Form von Symbolen und erweiterbaren Knoten

Herkömmliche VIs und Express-VIs können entweder als Symbole oder erweiterbare Knoten dargestellt werden. Bei erweiterbaren Knoten wird das jeweilige Symbol auf einem farbigen Hintergrund dargestellt. Dieser ist bei SubVIs gelb und bei Express-VIs blau. Gegenüber Symbolen nehmen die erweiterbaren Knoten mehr Platz im Blockdiagramm ein. Sie sind jedoch einfacher zu verbinden und ermöglichen eine einfachere Dokumentation des Blockdiagramms. Per Voreinstellung werden SubVIs im Blockdiagramm durch Symbole dargestellt und Express-VIs durch erweiterbare Knoten. Wenn Sie möchten, dass ein Sub- oder Express-VI als erweiterbarer Knoten dargestellt wird, klicken Sie es mit der rechten Maustaste an und deaktivieren Sie im Kontextmenü die Option **Als Symbol anzeigen**.

Hinweis Bei der Darstellung als erweiterbarer Knoten kann jedoch das entsprechende Anschlussfeld nicht angezeigt werden und es ist nicht möglich, den Datenbankzugriff zu aktivieren.

Wenn ein erweiterbarer Knoten aufgezogen wird, werden unter dem Symbol die jeweiligen Ein- und Ausgangsanschlüsse angezeigt. Die Parameternamen optional zu verbindender Anschlüsse sind grau hinterlegt. Wenn das erweiterbare Feld nicht komplett aufgezogen ist, werden die rest-

lichen Anschlüsse, deren Verbindung erforderlich ist oder empfohlen wird und deren Parameternamen nicht aufgeführt sind, im farbigen Hintergrund des Symbols als Pfeile dargestellt. Wenn der erweiterbare Knoten aufgezogen und ein optionaler Anschluss im unteren Bereich verbunden ist, wird der Anschluss einem Verkleinern des Knotens auf eine Größe, in der der Parametername nicht mehr angezeigt wird, oben im farbig hinterlegten Feld dargestellt. Wird die Verbindung jedoch entfernt, wird der entsprechende Ein- oder Ausgangspfeil nicht dargestellt.

Standardmäßig werden im unteren Bereich des SubVI- oder Express-VI-Knotens immer die Eingänge zuerst aufgeführt. Wenn ein Anschluss im erweiterbaren Bereich des Knotens mit der rechten Maustaste angeklickt wird und die Option **Eingang/Ausgang auswählen** verwendet wird, kann aber auch ein bestimmter Ein- oder Ausgang gewählt werden, der an entsprechender Stelle angezeigt werden soll. Ein- und Ausgänge sind im Kontextmenü durch eine Linie voneinander getrennt. Die Parameternamen werden in erweiterbaren Sub- und Express-VIs im farbigen Feld angezeigt, mit dem das Symbol hinterlegt ist. Um die Symbolbreite an die Textlänge anzupassen, klicken Sie mit der rechten Maustaste auf das VI und wählen Sie aus dem Kontextmenü die Option **Größenanpassung an Text** aus.

Im folgenden SubVI, das als erweiterbarer Knoten dargestellt ist, werden vier von zehn Ein- und Ausgangsanschlüssen angezeigt.

Weitere Hinweise zu Express-VIs entnehmen Sie bitte dem Handbuch *Erste Schritte mit LabVIEW*.

Erstellen von SubVIs aus VI-Abschnitten

Konvertieren Sie einen Teil eines VIs in ein SubVI, indem Sie mit Hilfe des Positionierungswerkzeugs den Abschnitt des Blockdiagramms markieren, den Sie wiederverwenden möchten. Wählen Sie anschließend die Option **Bearbeiten»SubVI erstellen** aus. Der markierte Abschnitt des Blockdiagramms wird durch ein Symbol für das neue SubVI ersetzt. LabVIEW erstellt Bedien- und Anzeigeelemente für das neue SubVI und verbindet das SubVI mit den vorhandenen Verbindungen.

Das Erstellen von SubVIs auf diese Weise ist zwar praktisch, erfordert jedoch auch eine durchdachte Planung der logischen VI-Hierarchie und der Objekte, die in das SubVI aufgenommen werden sollen. Auch sollte die Funktionalität des entstehenden VIs möglichst nicht verändert werden.

Entwerfen von SubVIs

Wenn die Benutzer das Frontpanel eines SubVIs nicht anzeigen müssen, können Sie bei dessen Gestaltung, beispielsweise mit Farben und Schriftarten, Zeit sparen. Dennoch ist der Aufbau des Frontpanels von Bedeutung, da Sie das Frontpanel möglicherweise bei der Fehlersuche im VI anzeigen müssen.

Platzieren Sie die Bedien- und Anzeigeelemente so, wie sie im Anschlussfeld erscheinen. Setzen Sie Bedienelemente auf die linke und Anzeigeelemente auf die rechte Seite des Frontpanels. Setzen Sie die Cluster **Fehlereingang** in die untere linke Ecke des Frontpanels und die Cluster **Fehlerausgang** in die untere rechte Ecke. Weitere Informationen zum Einrichten eines Anschlussfeldes finden Sie in Abschnitt *Anschlussfeld einrichten* dieses Kapitels.

Wenn ein Bedienelement über einen voreingestellten Wert verfügt, setzen Sie diesen als Teil des Namens des Bedienelements in Klammern. Fügen Sie dem Bedienelementnamen gegebenenfalls auch die Maßeinheiten hinzu. Wenn mit einem Bedienelement beispielsweise die obere Grenztemperatur festgelegt wird, und dieses Element auf den Standardwert 25 °C eingestellt ist, nennen Sie das Bedienelement **Obere Grenztemperatur (25 degC)**. Wenn Sie das SubVI auf mehreren Plattformen einsetzen möchten, vermeiden Sie Sonderzeichen in Bedienelementnamen. Verwenden Sie also beispielsweise **degC** anstelle von **°C**.

Benennen Sie boolesche Bedienelemente so, dass die Benutzer erkennen können, welche Funktion das Bedienelement im Status TRUE (WAHR) ausführt. Verwenden Sie Namen wie **Abbrechen**, **Zurücksetzen** und **Initialisieren**, mit denen die Aktion aussagekräftig beschrieben wird.

VI-Hierarchien anzeigen

Im **Hierarchiefenster** sehen Sie eine grafische Darstellung der Aufrufhierarchie aller VIs im Speicher, einschließlich der Typdefinitionen und globalen Variablen. Zur Anzeige des **Hierarchiefensters** wählen Sie **Durchsuchen»VI-Hierarchie anzeigen**. Verwenden Sie dieses Fenster, um SubVIs und andere Knoten anzuzeigen, aus denen sich das aktuelle VI zusammensetzt.

Wenn Sie das Bedienwerkzeug über Objekte im **Hierarchiefenster** bewegen, zeigt LabVIEW den Namen des jeweiligen VIs an. Mit Hilfe des Positionierwerkzeugs können VIs aus dem **Hierarchiefenster** in das Blockdiagramm gezogen werden, um diese als SubVIs in einem anderen VI zu verwenden. Sie können auch einen oder mehrere Knoten auswählen und in die Zwischenablage kopieren und anschließend in andere Blockdiagramme einfügen. Mit einem Doppelklick auf ein VI lässt sich das entsprechende Frontpanel öffnen.

Bei VIs, die SubVIs enthalten, wird am unteren Rand des Symbols eine Pfeilschaltfläche angezeigt. Klicken Sie auf diese Pfeilschaltfläche, um SubVIs ein- oder auszublenden. Eine rote Pfeilschaltfläche wird angezeigt, wenn alle SubVIs ausgeblendet sind. Ansonsten ist ein schwarzer Pfeil zu sehen.

Speichern von VIs

VIs können entweder als einzelne Dateien gespeichert oder in einer VI-Bibliothek gruppiert werden. VI-Bibliotheksdateien verfügen über die Dateinamenerweiterung `.llb`. Es wird jedoch empfohlen, VIs als Einzeldateien in Verzeichnisse zu speichern, insbesondere, wenn mehrere Entwickler am gleichen Projekt arbeiten. Für nähere Informationen zur Gliederung von VIs in Verzeichnissen lesen Sie in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter *Organizing VIs in Directories* nach.

Vorteile des Speicherns von VIs als Einzeldateien

In der folgenden Liste werden die Gründe für das Speichern von VIs als Einzeldateien aufgeführt:

- Sie können das Dateisystem zum Verwalten der Einzeldateien verwenden.
- Sie können mit Unterverzeichnissen arbeiten.

- VIs und Bedienelemente lassen sich in einzelnen Dateien stabiler speichern als ein gesamtes Projekt in einer Einzeldatei.
- Sie können die integrierten Quellcode-Kontrollwerkzeuge des Professional Development Systems oder Quellcode-Kontrollwerkzeuge von Fremdanbietern verwenden.

Vorteile des Speicherns von VIs als Bibliotheken

In der folgenden Liste werden die Gründe für das Speichern von VIs als Bibliotheken aufgeführt:

- Sie können bis zu 255 Zeichen zum Benennen von Dateien verwenden.

  (Mac OS) Bei Mac OS 9.*x* oder einer Vorgängerversion sind Dateinamen auf 31 Zeichen begrenzt. Bei Bibliotheksnamen besteht jedoch keine Einschränkung.
- Eine VI-Bibliothek lässt sich einfacher auf andere Plattformen übertragen als mehrere Einzel-VIs. Außerdem kann damit besser sichergestellt werden, dass der Anwender alle benötigten Dateien erhält.
- Sie können die Dateigröße Ihres Projekts geringfügig verringern, da VI-Bibliotheken komprimiert werden und dadurch weniger Speicherplatz erfordern.
- Sie können VIs in einer Bibliothek als Top-Level VIs markieren, LabVIEW öffnet dann automatisch alle Top-Level VIs, wenn Sie die Bibliothek öffnen.

Hinweis Viele der integrierten VIs und Beispiele sind ebenfalls in VI-Bibliotheken gespeichert, wodurch sichergestellt wird, dass sie auf allen Plattformen an derselben Stelle gespeichert sind.

Wenn Sie mit VI-Bibliotheken arbeiten, ziehen Sie in Betracht, Ihre Applikation in mehrere VI-Bibliotheken aufzuteilen. Setzen Sie die High-Level-VIs in eine VI-Bibliothek, und richten Sie weitere Bibliotheken ein, die VIs nach Funktionen getrennt enthalten. Das Speicher von Änderungen dauert bei VI-Bibliotheken länger als bei einzelnen VIs, da die Änderungen in größere Dateien gespeichert werden müssen. Das Speichern von Änderungen in große Bibliotheken kann auch den Speicherbedarf erhöhen und dadurch die Performance verringern. Versuchen Sie, die Größe der Bibliotheken auf 1 MB zu begrenzen.

Verwalten von VIs in Bibliotheken

Verwenden Sie den VI Dateimanager, auf den Sie durch Auswahl von **Werkzeuge»Verwaltung von VI-Bibliotheken** zugreifen, um das Kopieren, Umbenennen und Löschen von Dateien in VI-Bibliotheken zu vereinfachen. Mit diesem Werkzeug können Sie auch neue VI-Bibliotheken und -Verzeichnisse erstellen und VI-Bibliotheken in Verzeichnisse umwandeln und umgekehrt. Das Erstellen von neuen VI-Bibliotheken und -Verzeichnissen und das Umwandeln von VI-Bibliotheken in Verzeichnisse und umgekehrt ist wichtig, wenn Sie VIs mit Quellcode-Kontrollwerkzeugen verwalten müssen.

Bevor Sie den VI Dateimanager aufrufen, schließen Sie alle VIs, die möglicherweise betroffen sind, um zu vermeiden, dass eine Dateioperation mit einem VI vorgenommen wird, das sich bereits im Speicher befindet.

(Windows 2000/XP/Me/98) Mit einem Doppelklick auf die `.llb`-Datei im Windows Explorer kann der Inhalt der Bibliothek angezeigt werden. Die darin befindlichen Dateien können wie gewöhnliche Dateien geöffnet, verschoben, kopiert, umbenannt oder gelöscht werden.

Benennen von VIs

Verwenden Sie beim Speichern von VIs aussagekräftige Namen. Beschreibende Namen wie `Temperaturüberwachung.vi` und `Serielles Lesen & Schreiben.vi` vereinfachen die Identifikation eines VIs und geben Auskunft, wie es zu verwenden ist. Wenn Sie unklare Namen wie beispielsweise `VI#1.vi` verwenden, werden Sie möglicherweise Schwierigkeiten beim Identifizieren von VIs bekommen, insbesondere dann, wenn Sie mehrere VIs gespeichert haben.

Überlegen Sie, ob die Benutzer die VIs auf einer anderen Plattform ausführen werden. Vermeiden Sie Zeichen, die bei einigen Betriebssystemen für spezielle Zwecke vorbehalten sind, wie beispielsweise `\ : / ? * < >` und `#`.

Wenn VIs unter Mac OS 9.*x* oder einer älteren Version ausgeführt werden sollen, sind für VI-Namen möglichst weniger als 31 Zeichen zu verwenden.

Speichern für eine Vorläuferversion

VIs können für eine ältere Version von LabVIEW gespeichert werden. Dadurch wird die Aktualisierung von LabVIEW erleichtert, und die VIs können bei Bedarf in zwei verschiedenen LabVIEW-Versionen verwaltet werden. Wenn Sie auf eine neue Version von LabVIEW aktualisieren, können Sie auf die letzte Version der VIs zurückgreifen.

Wenn Sie ein VI für eine Vorläuferversion speichern, wandelt LabVIEW nicht nur dieses VI um, sondern auch alle VIs in dessen Hierarchie mit Ausnahme der `vi.lib`-Dateien.

Oftmals nutzt ein VI Funktionen, die in der Vorläuferversion von LabVIEW nicht verfügbar waren. In solchen Fällen speichert LabVIEW so viel von dem VI wie möglich und erstellt einen Bericht der Komponenten, die nicht umgewandelt werden konnten. Dieser Bericht wird unverzüglich im Dialogfeld **Warnungen** angezeigt. Klicken Sie auf **OK**, um diese Warnungen zu bestätigen und das Dialogfeld zu schließen. Klicken Sie auf **Speichern**, um die Warnungen für eine spätere Überprüfung in einer Textdatei zu speichern.

Bereitstellen von VIs

Wenn Sie VIs auf anderen Computern oder für andere Benutzer bereitstellen möchten, überlegen Sie, ob das Blockdiagramm vom Anwender bearbeitbar sein soll oder ob es ausgeblendet oder entfernt werden soll.

Wenn ein VI einschließlich des Quellcodes (Blockdiagramms) anderen Anwendern zugänglich gemacht werden soll, ist es unter Umständen sinnvoll, das Blockdiagramm durch ein Passwort zu schützen. In diesem Fall ist das Blockdiagramm weiterhin verfügbar, der Benutzer muss jedoch ein Passwort eingeben, damit er das Blockdiagramm anzeigen oder bearbeiten kann.

Wenn VIs an Programmierer weitergegeben werden sollen, die mit anderen Programmiersprachen arbeiten, kann das VI in eine ausführbare Anwendung oder eine Shared Library umgewandelt werden. Eine eigenständige Applikation oder eine gemeinsam genutzte Bibliothek ist die richtige Wahl, wenn Sie nicht davon ausgehen, dass die Benutzer die VIs bearbeiten. Die Benutzer können zwar die Applikation oder die gemeinsame Bibliothek nutzen, sie können jedoch die Blockdiagramme nicht bearbeiten oder anzeigen.

Hinweis Ausführbare Anwendungen und Shared Libraries können nur im LabVIEW Professional Development System oder mit Hilfe des Application Builders erstellt werden.

Wenn ein VI anderen Anwendern zugänglich gemacht werden soll, kann auch das Blockdiagramm entfernt werden, so dass das VI nicht mehr bearbeitet werden kann. Wählen Sie **Datei»Mit Optionen speichern**, um VIs ohne die Blockdiagramme zu speichern und so die Dateigröße zu reduzieren, und um zu verhindern, dass Benutzer den Quellcode ändern. VIs, die

ohne Blockdiagramm gespeichert worden sind, lassen sich weder auf eine andere Plattform übertragen noch auf eine zukünftige Version von LabVIEW aktualisieren.

Vorsicht! Wenn Sie VIs ohne Blockdiagramme speichern, stellen Sie sicher, dass die Originalversion der VIs *nicht* überschrieben wird, indem Sie zum Beispiel als Speicherplatz ein anderes Verzeichnis wählen oder die VIs umbenennen.

Weitere Informationen zum Übertragen von VIs auf andere Plattformen und das Lokalisieren von VIs finden Sie in der Application Note *Portieren und Lokalisieren von LabVIEW-VIs*.

Erstellen von eigenständigen Applikationen und Shared Libraries

Mit Hilfe des Application Builders unter **Werkzeuge»Applikation oder "Shared Library" (DLL) erzeugen** lassen sich neben ausführbaren Anwendungen und Installationsprogrammen auch gemeinsam genutzte Bibliotheken (Shared Libraries, DLLs) erstellen. Diese empfehlen sich bei VIs, die über textbasierte Programmiersprachen aufgerufen werden sollen, wie zum Beispiel LabWindows™/CVI™, Microsoft Visual C++ oder Microsoft Visual Basic.

Über Shared Libraries können die Funktionen eines erstellten VIs anderen Programmierern zur Verfügung gestellt werden. Daneben lässt sich mit Hilfe von Shared Libraries in LabVIEW entwickelter Quellcode in andere Programmiersprachen einbinden.

Ausführbare Anwendungen und Shared Libraries können jedoch nur ausgeführt werden. Eine Anzeige oder Bearbeitung des Blockdiagramms ist nicht möglich.

Hinweis Mit dem LabVIEW Professional Development System wird der Application Builder bereitgestellt. Wenn Sie mit dem LabVIEW Base Package oder dem Full Development System arbeiten, können Sie den Application Builder separat erwerben. Besuchen Sie dazu unsere Internetseite `ni.com/info` und geben Sie den Info-Code `rdlv21` ein. Über die Registerkarten des Dialogfensters **Applikation oder Shared Library erzeugen** können dann für die zu erstellende Anwendung oder Shared Library verschiedene Einstellungen getroffen werden. Nachdem Sie diese Einstellungen definiert haben, speichern Sie diese in ein Skript, damit Sie die Applikation gegebenenfalls auf einfache Weise erneut erstellen können.

Für weitere Informationen zur Installation des Application Builders lesen Sie bitte die *LabVIEW Application Builder Release Notes*.

Teil II

Erstellen und Bearbeiten von VIs

In diesem Teil werden die Komponenten, VIs und Funktionen von LabVIEW erörtert, die Ihnen zum Erstellen von Applikationen zur Verfügung stehen. Zu jedem Kapitel werden die Einsatzmöglichkeiten der jeweiligen LabVIEW-Komponente sowie die einzelnen Klassen von VIs und Funktionen beschrieben.

Teil II, *Erstellen und Bearbeiten von VIs*, umfasst folgende Kapitel:

- In Kapitel 8, *Schleifen und Strukturen*, wird die Verwendung von Strukturen im Blockdiagramm erörtert, mit deren Hilfe Blockdiagrammabschnitte wiederholt, unter bestimmten Bedingungen oder in einer bestimmten Reihenfolge ausgeführt werden können.
- Im Kapitel 9, *Ereignisgesteuerte Programmierung*, finden Sie in eine Einführung in das Erzeugen und Benennen von Ereignissen sowie die dynamische und statische Ereignisregistrierung.
- Kapitel 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*, gibt Ihnen einen Überblick über die Verwendung von Strings, Arrays und Clustern zur Gruppierung von Daten.
- In Kapitel 11, *Lokale und globale Variablen*, wird erläutert, wie mit Hilfe von lokalen und globalen Variablen ein Datenaustausch zwischen Blockdiagrammabschnitten möglich ist, die nicht auf herkömmliche Weise miteinander verbunden werden können.
- Kapitel 12, *Graphen und Diagramme*, gibt Ihnen einen Einblick in die Verwendung von Graphen und Diagrammen zur grafischen Darstellung von Daten.
- In Kapitel 13, *VIs für Grafiken und Sound*, ist die Verwendung von Grafiken und Sound in VIs beschrieben.
- Kapitel 14, *Datei-I/O*, gibt Ihnen eine Übersicht über die Verwendung von Datei-I/O-Operationen.

- In Kapitel 15, *Dokumentieren und Drucken von VIs*, wird das Dokumentieren und Drucken von VIs erörtert.
- In Kapitel 16, *Anpassen von VIs*, finden Sie Hinweise dazu, wie Sie VIs und SubVIs entsprechend Ihren Bedürfnissen anpassen können.
- Kapitel 17, *Programmatische Steuerung von VIs*, beschreibt die Kommunikation mit VIs und anderen Instanzen von LabVIEW, so dass LabVIEW und VIs programmatisch gesteuert werden können.
- In Kapitel 18, *Arbeiten mit LabVIEW im Netzwerk*, finden Sie Informationen zur Verwendung von VIs für die Kommunikation oder Vernetzung mit anderen Prozessen, die beispielsweise in anderen Applikationen oder auf Netzwerkrechnern ausgeführt werden.
- In Kapitel 19, *Windows-Konnektivität*, wird erörtert, wie Objekte, Befehle und Funktionen in LabVIEW so konfiguriert werden, dass andere Windows-Applikationen darauf zugreifen können.
- Im Kapitel 20, *Aufrufen von Code aus textbasierten Programmiersprachen*, finden Sie Hinweise dazu, wie Programmcode textbasierter Programmiersprachen von LabVIEW aus aufgerufen werden kann und wie DLLs eingesetzt werden.
- In Kapitel 21, *Formeln und Gleichungen*, wird die Verwendung von Gleichungen in VIs beschrieben.

8

Schleifen und Strukturen

Strukturen sind grafische Darstellungen der Schleifen und Case-Anweisungen in textbasierten Programmiersprachen. Verwenden Sie Strukturen im Blockdiagramm, um Codeblöcke zu wiederholen und Code bedingungsabhängig oder in einer bestimmten Reihenfolge auszuführen.

Wie andere Knoten verfügen auch Strukturen über Anschlüsse, über die sie mit anderen Blockdiagrammknoten verbunden werden können, werden automatisch ausgeführt, wenn Eingabedaten verfügbar sind, und liefern Daten an Ausgabeverbindungen, wenn die Ausführung abgeschlossen ist.

Jede Struktur hat einen markanten, in der Größe veränderbaren Rahmen, mit dem der Abschnitt des Blockdiagramms umschlossen wird, der entsprechend den Regeln der Struktur ausgeführt wird. Ein Blockdiagrammabschnitt innerhalb einer Struktur wird als Subdiagramm bezeichnet. Die Anschlüsse, die Daten an Strukturen übergeben beziehungsweise Daten aus Strukturen übernehmen, werden Tunnel genannt. Ein Tunnel ist ein Verbindungspunkt an einem Strukturrahmen.

Weitere Informationen ...

Weitere Informationen zum Verwenden von Strukturen finden Sie in der *LabVIEW-Hilfe*.

Mit folgenden Strukturen, die sich auf der Palette **Strukturen** befinden, kann gesteuert werden, wie in einem Blockdiagramm Prozesse ausgeführt werden:

- **FOR-Schleife**—Wiederholt die Ausführung eines Subdiagramms so oft wie vorgegeben.
- **While-Schleife**—Führt ein Subdiagramm so lange aus, bis eine bestimmte Bedingung erfüllt ist.
- **Case-Struktur**—Enthält mehrere Rahmen, von denen je nach dem Wert, der an der Struktur anliegt, jeweils einer ausgeführt wird.
- **Sequenzstruktur**—Enthält ein oder mehrere Subdiagramme, die nacheinander ausgeführt werden.

- **Formelknoten**—Führt mathematische Operationen auf Grundlage einer numerischen Eingabe durch. Weitere Informationen zum Einsatz von Formelknoten finden Sie im Abschnitt *Formelknoten* des Kapitels 21, *Formeln und Gleichungen*.
- **Ereignisstruktur**—Enthält einen oder mehrere Rahmen, die dann ausgeführt werden, wenn während des Programmablaufs vom Benutzer bestimmte Ereignisse ausgelöst werden.

Um ein entsprechendes Kontextmenü aufzurufen, klicken Sie mit der rechten Maustaste auf den Rahmen der Struktur.

FOR- und While-Schleifenstrukturen

Mit FOR- und While-Schleifen ist es möglich, sich wiederholende Operationen zu steuern.

FOR-Schleifen

Bei einer FOR-Schleife (siehe Abbildung links) wird das Subdiagramm so oft wie vorgegeben ausgeführt.

Durch den Eingabewert des Schleifenzählers (siehe links) wird bestimmt, wie oft das Subdiagramm wiederholt werden soll. Sie können die Anzahl explizit einstellen, indem Sie einen Wert von außerhalb der Schleife mit der linken oder der oberen Seite des Zähl-Anschlusses verbinden, oder Sie können die Anzahl implizit mit der Auto-Indizierung festlegen. Weitere Informationen zum impliziten Festlegen der Anzahl finden Sie in Abschnitt *Verwenden der Auto-Indizierung zum Einstellen der Wiederholungen von FOR-Schleifen* dieses Kapitels.

Der Iterationsanschluss (ein Ausgabeanschluss, siehe links) gibt die Anzahl der abgeschlossenen Schleifendurchläufe aus. Die Zählung beginnt dabei immer bei Null. Das heißt, während des ersten Durchlaufs gibt der Iterationsanschluss den Wert `0` aus.

Sowohl der Schleifenzähler als auch der Iterationsanschluss sind vom Typ “vorzeichenbehafteter Long Integer”. Wenn Sie eine Fließkommazahl mit dem Zähl-Anschluss verbinden, rundet LabVIEW diese Zahl und zwingt sie innerhalb des Bereichs. Wenn am Schleifenzähler der Wert `0` oder ein negativer Wert anliegt, wird die Schleife nicht ausgeführt, und an den Ausgängen werden die für den jeweiligen Datentyp voreingestellten Werte ausgegeben.

Fügen Sie der FOR-Schleife Schieberegister hinzu, um Daten aus der aktuellen Wiederholung an die nächste Wiederholung zu übergeben. Nähere Einzelheiten zum Hinzufügen von Schieberegistern zu einer Schleife finden Sie in Abschnitt *Schieberegister und Rückkopplungsknoten in Schleifen* dieses Kapitels.

While-Schleifen

Ähnlich einer Do-Schleife oder einer Repeat-Until-Schleife in textbasierten Programmiersprachen wird bei einer While-Schleife wie links gezeigt ein Subdiagramm so lange ausgeführt, bis eine bestimmte Bedingung zutrifft.

Die While-Schleife führt das Subdiagramm aus, bis der Bedingungsanschluss, ein Eingabeanschluss, einen bestimmten booleschen Wert empfängt. Der Bedingungsanschluss ist standardmäßig auf **Weiter wenn TRUE** (siehe links) eingestellt. Wenn der Bedingungsanschluss auf **Stopp wenn TRUE** eingestellt ist, wird die Schleife so lange ausgeführt, bis der Wert TRUE anliegt. Um den Anschluss wie links abgebildet auf **Weiter wenn TRUE** zu stellen, klicken Sie mit der rechten Maustaste auf den Rahmen der Schleife und wählen Sie die entsprechende Option aus dem Kontextmenü. In diesem Fall wird die Schleife so lange ausgeführt, bis der Wert FALSE anliegt. Sie können den Bedingungsanschluss auch mit dem Bedienwerkzeug anklicken, um die Bedingung zu ändern.

Über den Bedingungsanschluss einer While-Schleife können Sie auch eine grundlegende Fehlerbehandlung durchführen. Wenn Sie einen Fehler-Cluster mit dem Bedingungsanschluss verbinden, wird nur der Wert TRUE (WAHR) oder FALSE (FALSCH) des Parameters **Status** des Fehler-Clusters an den Anschluss übergeben. Darüber hinaus ändern sich auch die Kontextmenüelemente **Stopp wenn TRUE** und **Weiter wenn TRUE** in **Bei Fehler stoppen** und **Bei Fehler fortfahren**. Weitere Informationen zu Fehler-Clustern und zur Fehleranalyse finden Sie in Abschnitt *Fehlerprüfung und Fehlerbehandlung* des Kapitels 6, *Ausführen von VIs und Fehlersuche*.

Der Iterationsanschluss (ein Ausgabeanschluss, siehe links) gibt die Anzahl der abgeschlossenen Schleifendurchläufe aus. Die Zählung beginnt dabei immer bei Null. Das heißt, während des ersten Durchlaufs gibt der Iterationsanschluss den Wert `0` aus.

Fügen Sie der While-Schleife Schieberegister hinzu, um Daten aus der aktuellen Wiederholung an die nächste Wiederholung zu übergeben. Nähere Einzelheiten zum Hinzufügen von Schieberegistern zu einer

Schleife finden Sie in Abschnitt *Schieberegister und Rückkopplungsknoten in Schleifen* dieses Kapitels.

Vermeiden von endlosen While-Schleifen

Wenn Sie den Anschluss des booleschen Bedienelements außerhalb der While-Schleife platzieren, wie in Abbildung 8-1 gezeigt, und das Bedienelement auf FALSE (FALSCH) und der Bedingungsanschluss beim Start der Schleife auf **Stopp wenn TRUE** eingestellt ist, ergibt sich eine Endlosschleife. Eine Endlosschleife kann auch entstehen, wenn das Bedienelement außerhalb der Schleife auf TRUE und der Bedingungsanschluss auf **Weiter wenn TRUE** gesetzt wurde.

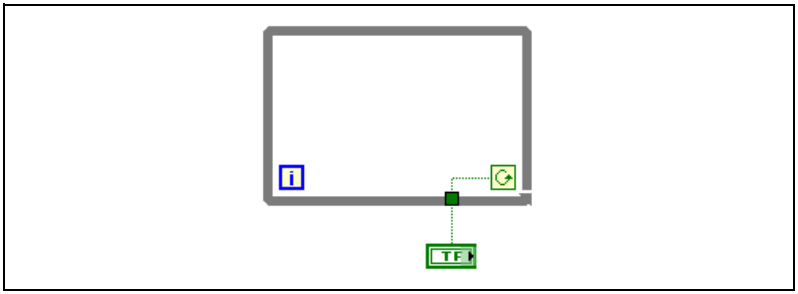


Abbildung 8-1. Endlose While-Schleife

Mit dem Ändern des Wertes des Bedienelements lässt sich die Endlosschleife nicht stoppen, da der Wert nur einmal gelesen wird, bevor die Schleife beginnt. Zum Stoppen einer Endlosschleife müssen Sie die Ausführung des VIs abbrechen, indem Sie auf der Symbolleiste auf die Schaltfläche **Abbrechen** klicken.

Auto-Indizierung von Schleifen

Wenn Sie ein Array mit dem Eingangstunnel einer FOR- oder While-Schleife verbinden, können Sie jedes Element in diesem Array durch Aktivierung der Auto-Indizierung lesen und verarbeiten. Weitere Informationen zu Arrays finden Sie im Kapitel 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Wenn Sie ein Array mit einem Eingabetunnel auf dem Rand der Schleife verbinden und die Auto-Indizierung am Eingabetunnel aktivieren, werden die Elemente des Arrays beginnend mit dem ersten Element nacheinander an die Schleife übergeben. Ist die Auto-Indizierung deaktiviert, wird das gesamte Array an die Schleife übergeben. Wenn Sie einen Array-Ausgabetunnel indizieren, empfängt das Ausgabe-Array ein neues Element aus

jeder Wiederholung der Schleife. Daher entsprechen auto-indizierte Ausgabe-Arrays in der Größe immer der Anzahl der Wiederholungen.

Klicken Sie mit der rechten Maustaste auf den Rahmen der Schleife, und wählen Sie aus dem Kontextmenü **Indizierung aktivieren** oder **Indizierung deaktivieren**, um die Auto-Indizierung zu aktivieren oder zu deaktivieren. Die Auto-Indizierung für While-Schleifen ist standardmäßig deaktiviert.

Die Schleife indiziert Skalarelemente aus 1D-Arrays, 1D-Arrays aus 2D-Arrays und so weiter. Das Gegenteil geschieht an Ausgabetunneln. Skalare Elemente werden sequenziell zu 1D-Arrays zusammengefasst, 1D-Arrays zu 2D-Arrays und so weiter.

Verwenden der Auto-Indizierung zum Einstellen der Wiederholungen von FOR-Schleifen

Wenn Sie die Auto-Indizierung für ein Array aktivieren, das mit dem Eingangsanschluss einer FOR-Schleife verbunden ist, stellt LabVIEW den Zähleranschluss entsprechend der Array-Größe ein, so dass Sie den Zähleranschluss nicht verbinden müssen. Da FOR-Schleifen verwendet werden können, um Arrays elementweise zu verarbeiten, aktiviert LabVIEW die Auto-Indizierung standardmäßig für jedes Array, das Sie mit einer FOR-Schleife verbinden. Wenn Arrays nicht elementweise verarbeitet werden müssen, kann die Auto-Indizierung deaktiviert werden.

Wenn Sie die Auto-Indizierung für mehrere Tunnel aktivieren oder den Zähleranschluss festlegen, bestimmt das kleinste Element die Anzahl der Widerholungen (Anzahl der Array-Zellen bzw. Zählanschluss) Wenn beispielsweise zwei auto-indizierte Arrays mit 10 beziehungsweise 20 Elementen an die Schleife übergeben werden und Sie den Zähl-Anschluss mit dem Wert 15 verbinden, wird die Schleife zehnmal ausgeführt, und somit indiziert die Schleife nur die ersten 10 Elemente des zweiten Arrays. Wenn Sie Daten aus zwei Quellen in einem Graphen zeichnen und die ersten 100 Elemente gezeichnet werden sollen, verbinden Sie den Zähl-Anschluss mit dem Wert 100. Wenn eine der Datenquellen nur 50 Elemente umfasst, wird die Schleife 50 Mal ausgeführt, und es werden nur die ersten 50 Elemente indiziert. Verwenden Sie die Funktion “Array-Größe”, um die Größe des Arrays festzustellen.

Wenn Sie einen Array-Ausgabetunnel indizieren, empfängt das Ausgabe-Array ein neues Element aus jeder Wiederholung der Schleife. Daher entsprechen auto-indizierte Ausgabe-Arrays in der Größe immer der Anzahl der Wiederholungen. Wenn die Schleife beispielsweise zehn Mal

ausgeführt wird, weist das Ausgabe-Array 10 Elemente auf. Wenn Sie die Auto-Indizierung für einen Ausgabetunnel deaktivieren, wird nur das Element der letzten Wiederholung der Schleife an den nächsten Knoten im Blockdiagramm übergeben. Die Auto-Indizierung wird durch ein Klammer-Symbol in den Übergabetunnel am Schleifenrand dargestellt. Die Dicke der Verbindung zwischen dem Ausgabetunnel und dem nächsten Knoten zeigt auch an, ob die Schleife mit Auto-Indizierung arbeitet. Mit Auto-Indizierung ist die Verbindung dicker, denn die Verbindung enthält ein Array und keinen Skalar.

Auto-Indizierung mit While-Schleife

Wenn Sie die Auto-Indizierung für ein Array aktivieren, das an eine While-Schleife übergeben wird, indiziert die While-Schleife das Array auf die gleiche Weise wie eine FOR-Schleife. Allerdings ist die Anzahl der von einer While-Schleife ausgeführten Wiederholungen nicht durch die Größe des Arrays beschränkt, da die While-Schleife wiederholt wird, bis eine bestimmte Bedingung zutrifft. Wenn eine While-Schleife über das Ende des Eingabe-Arrays hinaus indiziert, wird der Standardwert des Elementtyps des Arrays an die Schleife übergeben. Sie können verhindern, dass der Standardwert an die While-Schleife übergeben wird, indem Sie die Funktion “Array-Größe” verwenden. Die Funktion “Array-Größe” gibt an, wie viele Elemente das Array beinhaltet. Richten Sie die While-Schleife so ein, dass die Ausführung beendet wird, wenn die Anzahl der Wiederholungen der Größe des Arrays entspricht.

Vorsicht! Da Sie die Größe des Ausgabe-Arrays nicht im Vorfeld bestimmen können, ist das Aktivieren der Auto-Indizierung für die Ausgabe einer FOR-Schleife effizienter als für die einer While-Schleife. Zu viele Wiederholungen können dazu führen, dass das System Probleme aufgrund von ungenügendem Speicher zurückmeldet.

Erstellen von Arrays mit Hilfe von Schleifen

Mit Schleifen können die Elemente eines Arrays nicht nur ausgelesen und verarbeitet werden, sondern es ist sogar möglich, mit FOR- und While-Schleifen Arrays zu erstellen. Verbinden Sie dazu den Ausgang eines in der Schleife befindlichen VIs oder einer Funktion mit dem Rahmen der Schleife. Bei While-Schleifen klicken Sie den entstandenen Ausgabetunnel mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Indizierung aktivieren**. Bei FOR-Schleifen wird die Indizierung per Voreinstellung automatisch aktiviert. Der Ausgabetunnel ist ein Array mit den Werten, die von der Funktion oder dem VI in allen Schleifendurchläufen ausgegeben wurden.

Weitere Beispiele zum Erstellen von Arrays finden Sie unter `examples\general\arrays.llb`.

Schieberegister und Rückkopplungsknoten in Schleifen

Mit Schieberegistern und Rückkopplungsknoten werden in FOR- und While-Schleifen Werte von einem Schleifendurchlauf zum nächsten übergeben.

Schieberegister

Wenn Werte eines Schleifendurchlaufs jeweils in den nächsten übernommen werden sollen, empfiehlt sich die Verwendung von Schieberegistern. Ein Schieberegister wird als ein Paar von Anschlüssen dargestellt (siehe links), die sich an den Längsseiten der Schleife gegenüber liegen. Der rechte Anschluss (durch einen Pfeil nach oben gekennzeichnet) speichert die Werte jedes Durchlaufs. Diese werden jeweils in den nächsten Durchlauf übernommen. Zur Erstellung eines Schieberegisters klicken Sie mit der rechten Maustaste auf den linken oder rechten Rand einer Schleife und wählen im Kontextmenü die Option **Schieberegister hinzufügen**.

Mit Schieberegistern kann jeder beliebige Datentyp übertragen werden, da sie sich automatisch auf den Datentyp des ersten Objekts einstellen, das mit dem Schieberegister verbunden ist. Die Daten, die an die Anschlüsse des jeweiligen Schieberegisters übergeben werden, müssen vom gleichen Typ sein.

Zur Initialisierung eines Schieberegisters verbinden Sie den Anschluss auf der linken Seite der Schleife mit einem Bedienelement oder einer Konstante. Damit wird der Wert zurückgesetzt, den das Schieberegister beim Start der Schleifenausführung weitergibt. Wenn Sie das Register nicht initialisieren, verwendet die Schleife den Wert der letzten Schleifenausführung oder den Standardwert für den entsprechenden Datentyp, wenn die Schleife noch nicht ausgeführt wurde.

Wenn also bei jedem VI-Start der zuletzt ausgegebene Wert als Anfangswert verwendet werden soll, muss das Register nicht initialisiert werden. Auf diese Weise können zum Beispiel Statusinformationen für die nachfolgende Ausführung des VIs gespeichert werden. Nachdem die Schleife ausgeführt wurde, verbleibt der letzte im Schieberegister gespeicherte Wert im rechten Anschluss.

Eine Schleife kann auch mehrere Schieberegister enthalten. Dadurch können die Werte mehrerer verschiedener Operationen in die jeweils nächste Schleifenausführung übernommen werden.

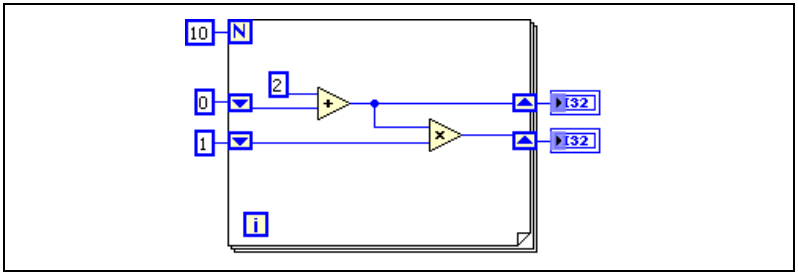


Abbildung 8-2. Schleife mit mehreren Schieberegistern

Gestapelte Schieberegister

Über gestapelte Schieberegister haben Sie die Möglichkeit, auf die Werte vergangener Schleifendurchläufe zuzugreifen. Um ein gestapeltes Schieberegister zu erstellen, klicken Sie den linken Anschluss mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Element hinzufügen**. Mit Hilfe von gestapelten Schieberegistern können die Werte mehrerer vorangegangener Schleifendurchläufe gespeichert und in die folgenden übertragen werden.

Sie kommen daher nur an der linken Seite der Schleife vor, da mit über den rechten Anschluss immer jeweils die Werte der aktuellen Schleifenausführung in die nächste übertragen werden.

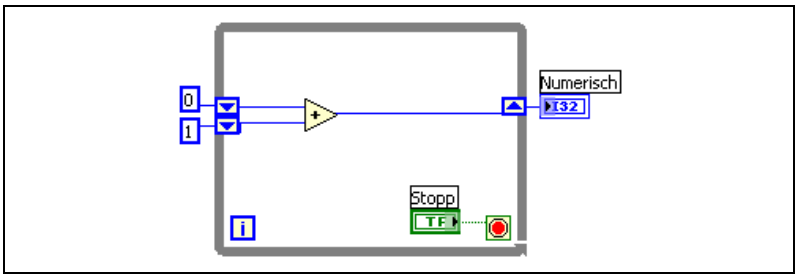


Abbildung 8-3. Gestapeltes Schieberegister

Wenn dem linken Anschluss ein weiterer hinzugefügt wird, werden die Werte der letzten zwei Durchläufe in den nächsten übertragen, wobei der aktuellere Wert immer im oberen Register gespeichert wird. Im unteren Anschluss befindet sich jeweils der Wert der davor liegenden Ausführung.

Ersetzen von Schieberegistern durch Tunnel

Wenn keine Werte mehr von einer Schleifenausführung in die nächste übertragen werden müssen, kann ein Schieberegister auch durch einen Tunnel ersetzt werden. Klicken Sie dazu das Schieberegister mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Durch Tunnel ersetzen**.

Wenn Sie in einer FOR-Schleife den Ausgangsanschluss eines Schieberegisters durch einen Tunnel ersetzen, wird die Verbindung zu einem damit verbundenen Objekt außerhalb der Schleife ungültig, da bei FOR-Schleifen standardmäßig die Indizierung aktiviert ist Um die Indizierung zu deaktivieren, klicken Sie mit der rechten Maustaste auf den Tunnel, und wählen Sie die Option **Indizierung deaktivieren**. Die Verbindung wird dann automatisch wiederhergestellt. Wenn die Indizierung beibehalten werden soll, entfernen Sie die ungültige Verbindung und das Anzeigeelement, klicken Sie mit der rechten Maustaste auf den Tunnel und wählen Sie **Anzeigeelement erstellen**.

Weitere Informationen zur Indizierung in Schleifen finden Sie im Abschnitt *Auto-Indizierung von Schleifen* dieses Kapitels.

Ersetzen von Tunneln durch Schieberegister

Wenn Werte von einer Schleifenausführung in die nächste übertragen werden sollen, kann ein Tunnel auch durch ein Schieberegister ersetzt werden. Klicken Sie dazu den Tunnel mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Durch Schieberegister ersetzen**. Es wird immer automatisch ein Paar von Schieberegisteranschlüssen erstellt, auch wenn an der gegenüberliegenden Seite des angeklickten Tunnels kein weiterer Tunnel vorhanden ist. Wenn es bereits ein Tunnelpaar gibt, wird aus dem angeklickten Tunnelanschluss ein Schieberegisteranschluss, und der Cursor verwandelt sich in ein Schieberegistersymbol. Klicken Sie nun auf einen Tunnel auf der gegenüberliegenden Seite des angeklickten Tunnels, um ihn durch ein Schieberegister zu ersetzen, oder klicken Sie an die entsprechende Stelle, um den anderen Anschluss einzufügen. Wenn ein Schieberegisteranschluss hinter einem Tunnel angezeigt wird, ist dieses nicht verbunden.

Wenn in einer While-Schleife ein Tunnel mit Indizierung in ein Schieberegister umgewandelt werden soll, werden alle Verbindungen zu Knoten außerhalb der Schleife ungültig, da Schieberegister nicht zur Auto-Indizierung geeignet sind. Entfernen Sie in diesem Fall die ungültige Verbindung, verbinden den rechten Ausgang des Schieberegisters mit einem anderen Tunnel, klicken diesen mit der rechten Maustaste an, wählen aus dem Kon-

textmenü die Option **Indizierung aktivieren** und verbinden Sie den Tunnel mit dem Knoten.

Weitere Informationen zur Indizierung in Schleifen finden Sie im Abschnitt *Auto-Indizierung von Schleifen* dieses Kapitels.

Rückkopplungsknoten

Ein Rückkopplungsknoten, wie links dargestellt, erscheint immer dann in einer FOR- oder While-Schleife, wenn der Ausgang eines SubVIs, einer Funktion oder einer Gruppe von SubVIs bzw. Funktionen an den Eingang desselben VIs bzw. derselben Funktion oder Gruppe angeschlossen wird. Ähnlich wie bei einem Schieberegister werden auch bei einem Rückkopplungsknoten nach jeder Schleifenausführung Werte beliebigen Datentyps gespeichert und in den nächsten Durchlauf übernommen. Sie dienen vor allem dazu, unnötig lange Verbindungen in Schleifen zu ersetzen. Mit dem Pfeil wird jeweils die Richtung des Datenflusses angezeigt.

Rückkopplungsknoten können auch manuell ausgewählt und in eine FOR- oder While-Schleife eingefügt werden. Wenn ein Rückkopplungsknoten auf eine Verbindung gesetzt wird, bevor diese verzweigt wurde, werden alle Werte an den Tunnel übergeben. Wenn der Knoten dagegen nach dem Verzweigen der Verbindung zum Tunnel eingesetzt wird, gibt er jeden Wert an den Eingang des VIs oder der Funktion zurück und übergibt den letzten Wert an den Tunnel.

Die FOR-Schleife in Abbildung 8-4 wird beispielsweise zehn Mal wiederholt. Bevor der Wert an den Eingang der Additionsfunktion übergeben wird, überträgt der Rückkopplungsknoten jeweils den Wert der vorangegangenen Schleifenausführung zum Tunnel. Im Tunnel befindet sich also immer der Wert des letzten Durchlaufs. Wenn die Schleife das letzte Mal wiederholt wird, speichert der Rückkopplungsknoten den letzten Wert (in diesem Fall 45) und gibt ihn weder an den Tunnel noch an die numerische Anzeige weiter. Bei Beendigung des VIs wird im numerischen Anzeigeelement daher 36 angezeigt, also der Wert der vorletzten Durchlaufs.

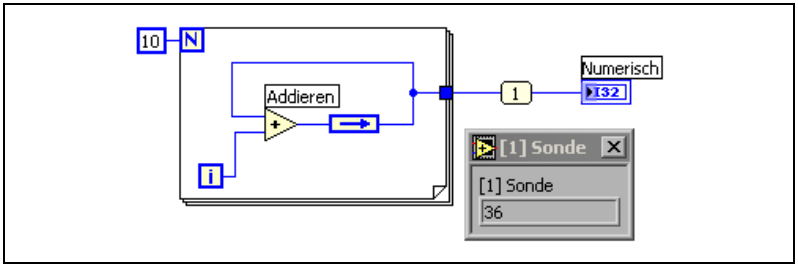


Abbildung 8-4. Ausgabe des vorletzten Wertes aus der FOR-Schleife

Die FOR-Schleife in Abbildung 8-5 wird ebenfalls zehn Mal wiederholt. In diesem Fall übergibt der Rückkopplungsknoten den Wert der letzten Schleifenausführung jedoch bei jedem Durchlauf an den Eingang der Additionsfunktion. Beim letzten Durchlauf wird Wert des vorherigen (36) an den Eingang der Additionsfunktion übertragen. Diese addiert nun den vom Iterationsanschluss erzeugten Wert (9) mit dem Wert, der vom Rückkopplungsknoten übergeben wird (36) und gibt das Ergebnis an den Tunnel. Bei Ausführungsende des VIs wird daher in der numerischen Anzeige **45** angezeigt.

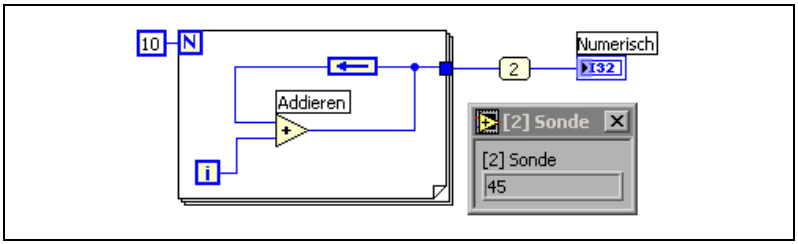


Abbildung 8-5. Ausgabe des letzten Wertes aus der FOR-Schleife

Initialisierung von Rückkopplungsknoten

Um die Schleife zu initialisieren, klicken Sie den Rückkopplungsknoten mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Initialisierungsanschluss**. Wenn Sie einen Rückkopplungsknoten auswählen oder ein initialisiertes Schieberegister in einen solchen umwandeln, wird der Initialisierungsanschluss automatisch angezeigt. Mit diesem wird der Anfangswert zurückgesetzt, den der Rückkopplungsknoten bei der ersten Schleifenausführung weitergibt. Wenn keine Initialisierung erfolgt, gibt der Rückkopplungsknoten entweder den letzten Wert, der in den Knoten geschrieben wurde, oder bei erstmaliger Ausführung der Schleife den Standardwert für den jeweiligen Datentyp weiter. Wenn der Initialisierunganschluss unverbunden bleibt, wird bei jeder Ausführung des

VIs als Anfangswert des Rückkopplungsknotens der letzte Wert der vorangegangenen Ausführung verwendet.

Ersetzen von Schieberegistern durch Rückkopplungsknoten

Um ein Schieberegister durch einen Rückkopplungsknoten zu ersetzen, klicken Sie dieses mit der rechten Maustaste an und wählen aus dem Kontextmenü die Option **Durch Rückkopplungsknoten ersetzen**. Zum Umwandeln eines Rückkopplungsknotens in ein Schieberegister klicken Sie diesen mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Durch Schieberegister ersetzen**.

Steuern des Timings

Möglicherweise möchten Sie die Geschwindigkeit steuern, mit der ein Prozess ausgeführt wird, wie beispielsweise die Geschwindigkeit, mit der Daten in einem Diagramm gezeichnet werden. Sie können dazu die Wartefunktion in der Schleife verwenden, um anzugeben, wie lange gewartet werden soll, bevor die Schleife erneut ausgeführt wird. Die Angabe erfolgt in Millisekunden.

Weitere Informationen zur Optimierung der Speicherausnutzung mit Hilfe einer Wartefunktion finden Sie in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter der Überschrift *Memory and Speed Optimization*.

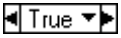
Case- und Sequenzstrukturen

In Case-, Sequenz- und Ereignisstrukturen sind immer mehrere Subdiagramme enthalten. Eine Case-Struktur führt ein Subdiagramm abhängig von dem an die Struktur übergebenen Eingabewert aus. Bei der flachen und der gestapelten Sequenzstruktur werden die darin enthaltenen Subdiagramme nacheinander ausgeführt. In einer Ereignis-Struktur hängt die Ausführung des Subdiagramms davon ab, wie der Benutzer auf das VI Einfluss nimmt.

Case-Strukturen

Eine Case-Struktur (siehe links) hat immer mindestens zwei Subdiagramme oder Cases. Davon ist immer nur jeweils ein Subdiagramm sichtbar, und die Struktur führt immer nur jeweils einen Case aus. Welches Subdiagramm ausgeführt wird, hängt vom jeweiligen Eingabewert ab.

Die Case-Struktur entspricht der CASE- bzw. `if...then...else`-Anweisung in textbasierten Programmiersprachen.



Im Case-Selektor am oberen Rand der Struktur (siehe links) wird der Name des jeweiligen Rahmens (mittlerer Teil) sowie die Dekrement- und Inkrement-Schaltfläche auf jeder Seite. Damit kann zwischen den einzelnen Rahmen umgeschaltet werden. Wenn Sie auf den Pfeil nach unten klicken, der sich rechts neben der Case-Beschriftung befindet, wird ein Pull-down-Menü mit den vorhandenen Cases angezeigt.

Um festzulegen, welcher Case ausgeführt werden soll, verbinden Sie einen Eingabewert mit dem Selektoranschluss (siehe links). Dabei kann es sich um einen Integer-, einen booleschen Wert, einen String oder einen Wert vom Enum-Typ handeln. Der Selektoranschluss kann an jede beliebige Stelle am linken Rand der Case-Struktur platziert werden. Wenn ein boolescher Wert an die Struktur übergeben wird, enthält diese einen `TRUE`- und einen `FALSE`-Case. Bei Integer-Werten und Enum-Werten oder Strings kann die Struktur beliebig viele Cases haben.

Für den Fall, dass Werte außerhalb des zulässigen Bereichs auftreten, sollte bei Case-Strukturen immer ein Standard-Case festgelegt werden. Ansonsten muss für jeden möglichen Eingabewert ein Case erstellt werden. Wenn der Selektoranschluss zum Beispiel vom Typ Integer ist, und Cases für die Eingangswerte `1`, `2` und `3` eingerichtet sind, sollte auch ein Standard-Case für den Fall vorhanden sein, dass der Eingabewert unerwartet `4` oder ein anderer ungültiger Integer-Wert ist.

Case-Selektorwerte und Datentypen

In die Selektorkennung können einzelne Werte, aber auch Listen und Bereiche von Werten eingegeben werden. Wenn mehrere Werte aufgelistet werden sollen, müssen diese durch Kommas voneinander getrennt werden. Zahlenbereiche sollten in der Form `10..20` eingegeben werden. Im vorliegenden Beispiel hieße das, dass alle Zahlen von 10 bis einschließlich 20 im Bereich enthalten sind. Sie können auch nach oben oder unten offene Bereiche verwenden. So wären bei `..100` alle Zahlen kleiner gleich 100 und bei `100..` alle Zahlen größer gleich 100 enthalten. Listen und Bereiche können jedoch auch kombiniert werden, wie zum Beispiel `..5, 6, 7..10, 12, 13, 14`. Wenn Werte mit sich überlappenden Bereichen in die Selektorkennung eingegeben werden, zeigt die Case-Struktur diese in der kürzestmöglichen Form an. Die Angabe im vorangegangenen Beispiel würde dann beispielsweise durch **`..10, 12..14`** ersetzt. Bei String-Bereichen sind im Bereich `a..c` beispielsweise `a` und `b` enthalten, jedoch nicht `c`. Damit `c` im Bereich enthalten ist, muss die Form `a..c,c` verwendet werden.

Wenn Sie String- und Enum-Werte in einem Case-Selektor verwenden, werden die Werte in Anführungszeichen angezeigt, zum Beispiel `"rot"`, `"grün"` und `"blau"`. Diese müssen jedoch nicht eingegeben werden; es sei denn, der String oder Enum-Wert enthält ein Komma oder stellt einen Bereich dar (`","` oder `"..."`). Zur Eingabe nicht-darstellbarer Zeichen in Strings sind durch einen Backslash eingeleitete Zeichenfolgen (Escape-Sequenzen) wie beispielsweise `\r` für Wagenrücklauf (Carriage Return, CR), `\n` für einen Zeilenvorschub (Linefeed, LF) und `\t` für einen Tabulator zu verwenden. Eine Liste dieser Escape-Sequenzen finden Sie in der *LabVIEW-Hilfe*.

Wenn Sie den Datentyp der Verbindung ändern, die mit dem Selektoranschluss einer Case-Struktur verbunden ist, wandelt diese automatisch die CASE-Selektorwerte in den neuen Datentyp um, sofern eine solche Konvertierung möglich ist. Wenn Sie einen numerischen Wert wie 19 in einen String umwandeln, lautet der String `"19"`. Strings können nur in einen numerischen Wert umgewandelt werden, wenn sie eine Zahl darstellen. Die anderen Werte bleiben Strings. Wenn Sie eine Zahl in einen booleschen Wert umwandeln, konvertiert LabVIEW `0` in FALSE und `1` in TRUE. Alle anderen numerischen Werte werden zu Strings.

Wenn der Selektorwert einen anderen Datentyp als das mit dem Selektoranschluss verbundene Objekt hat, wird dieser rot dargestellt, um anzuzeigen, dass die Struktur (und damit das VI) nicht ausführbar ist. Aufgrund möglicher Rundungsfehler in der Berechnung können jedoch keine Fließkommazahlen als Case-Selektorwerte verwendet werden. Wenn ein Fließkommawert mit dem Case verbunden wird, rundet LabVIEW den Wert auf die nächste gerade Ganzzahl. Wenn ein Fließkommawert in den Case-Selektor eingegeben wird, erscheint dieser rot, um anzuzeigen, dass das VI erst ausgeführt werden kann, wenn der Wert gelöscht oder bearbeitet wurde.

Eingabe- und Ausgabetunnel

Eine Case-Struktur kann mehrere Ein- und Ausgabetunnel enthalten. Die Eingänge stehen allen Rahmen zur Verfügung, müssen jedoch nicht für jeden Rahmen verwendet werden. Die Ausgabetunnel müssen dagegen für jeden Case festgelegt werden. Wenn Sie für einen Case einen Ausgabetunnel erstellen, erscheinen bei allen anderen Cases an derselben Position am Rahmen ebenfalls Tunnel. Wenn ein Case keinen Wert an den Tunnel übergibt, erscheinen alle Tunnel als weiße Quadrate. Die Datenquelle eines Ausgabetunnels kann für jeden Case unterschiedlich sein. Die verwendeten Datentypen müssen jedoch zueinander kompatibel sein. Wenn Sie den Ausgabetunnel mit der rechten Maustaste anklicken und aus dem Kontext-

menü die Option **Standardwert für offene Verbindung** auswählen, werden für alle nicht verbundenen Tunnel Standardwerte verwendet.

Fehlerbearbeitung mit Case-Strukturen

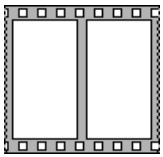
Wenn der Selektoranschluss einer Case-Struktur mit einem Fehler-Cluster verbunden wird, zeigt die Selektorkennung zwei Cases an, und zwar `Fehler` und `Kein Fehler`. Der Rahmen des `Fehler`-Cases wird rot und der des `Kein Fehler`-Cases grün dargestellt. Je nach Fehlerstatus wird immer der entsprechende Case ausgeführt. Weitere Informationen zur Fehlerbearbeitung finden Sie im Abschnitt *Fehlerbehandlung* des Kapitels 6, *Ausführen von VIs und Fehlersuche*.

Sequenzstrukturen

Sequenzstrukturen sind dadurch gekennzeichnet, dass die darin enthaltenen Rahmen in sequentieller Reihenfolge ausgeführt werden. Sie können jedoch auch nur aus einem Subdiagramm bestehen. Sequenzstrukturen sollten nicht zu häufig verwendet werden. Wann sich der Einsatz empfiehlt, erfahren Sie in den Abschnitten *Einsatz von Sequenzstrukturen* und *Zu häufigen Einsatz von Sequenzstrukturen vermeiden* dieses Kapitels.

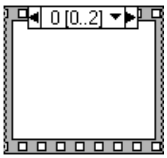
Es gibt zwei Arten von Sequenzstrukturen: flache und gestapelte.

Flache Sequenzstruktur



Bei einer flachen Sequenzstruktur wie links abgebildet werden alle Rahmen nebeneinander angezeigt und von links nach rechts ausgeführt. Mit flachen Sequenzstrukturen lässt sich eine zu häufige Verwendung von lokalen Sequenzvariablen vermeiden und die Dokumentation des Blockdiagramms verbessern. Beim Hinzufügen oder Löschen von Rahmen passt sich die Größe der Struktur automatisch an. Rahmen können auch über Ausschneiden und Einfügen neu angeordnet werden.

Gestapelte Sequenzstruktur



Bei einer gestapelten Sequenzstruktur wie links abgebildet werden die Rahmen übereinander dargestellt, so dass immer jeweils ein Rahmen sichtbar ist. Die Abarbeitung erfolgt der Reihe nach bei Rahmen 0 beginnend. Die Daten werden immer erst nach Beendigung des letzten Rahmens ausgegeben. Die gestapelte Sequenzstruktur empfiehlt sich insbesondere, um Platz auf dem Blockdiagramm zu sparen.

In der Beschriftung am oberen Rand der gestapelten Sequenzstruktur (siehe links) wird die Nummer des aktuellen Rahmens und die Gesamtzahl der

Rahmen angezeigt. Hier können die verfügbaren Rahmen durchblättert und neu angeordnet werden. Die Rahmenbeschriftung am oberen Rand der Sequenzstruktur ähnelt der Selektorkennung in einer Case-Struktur. Sie enthält in der Mitte die Rahmennummer und an den Außenkanten Pfeile zur Rahmenauswahl. Klicken Sie auf die Pfeile, sich um die verfügbaren Rahmen anzeigen zu lassen. Bei einem Klick auf den nach unten zeigenden Pfeil rechts neben der Rahmennummer wird ein Kontextmenü mit den verfügbaren Rahmen geöffnet. Zum Umsortieren der Rahmen klicken Sie mit der rechten Maustaste auf den Rand des Rahmens und wählen Sie aus dem Kontextmenü die Option **Diesen Rahmen zu ... setzen** mit der entsprechenden Rahmennummer aus.

Im Gegensatz zu Case-Strukturen können bei Sequenzstrukturen keine Werte in die Beschriftung eingegeben werden. Wenn Rahmen hinzugefügt, entfernt oder in ihrer Anordnung verändert werden, wird die Nummerierung automatisch angepasst.

Einsatz von Sequenzstrukturen

Sequenzstrukturen sollten verwendet werden, um für Blockdiagrammabschnitte eine Ausführungsreihenfolge festzulegen, bei denen keine natürliche Datenabhängigkeit vorliegt. Datenabhängigkeit bedeutet, dass ein Knoten erst ausgeführt werden kann, wenn der Knoten, von dem dieser Daten erhält, die Ausführung beendet hat.

Die Ausführungsreihenfolge wird innerhalb der einzelnen Rahmen einer Sequenzstruktur wie im Rest des Blockdiagramms durch die Datenabhängigkeit bestimmt. Weitere Informationen zur Datenabhängigkeit finden Sie in Abschnitt *Datenabhängigkeit und künstliche Datenabhängigkeit* des Kapitels 5, *Erstellen des Blockdiagramms*.

Anders als bei Case-Strukturen kann es bei gestapelten Sequenzstrukturen zu jedem Tunnel immer nur eine Datenquelle geben. Zwar kann die Ausgabe grundsätzlich von jedem Rahmen erfolgen, jedoch muss immer die Ausführung aller Rahmen abgeschlossen sein. Wie bei Case-Strukturen stehen die Daten an den Eingabetunneln für alle Rahmen zur Verfügung.

Um Daten aus einem Rahmen an einen beliebigen nachfolgenden Rahmen einer gestapelten Sequenzstruktur zu übergeben, verwenden Sie lokale Sequenz-Variablen, wie links dargestellt. In der lokalen Sequenz-Variable des Rahmens, der die Datenquelle enthält, wird ein nach außen weisender Pfeil angezeigt. Der Anschluss im nachfolgenden Rahmen enthält einen nach innen weisenden Pfeil, womit angezeigt wird, dass der Anschluss als Datenquelle für den Rahmen fungiert. Sie können lokale Sequenz-Varia-

blen erst in den Rahmen auslesen, die dem Rahmen, der auf die lokale Sequenz-Variable schreit folgen.

Beispiele zur Verwendung von Sequenzstrukturen finden Sie unter `examples\general\structs.llb`.

Zu häufigen Einsatz von Sequenzstrukturen vermeiden

Die sonst übliche parallele Verarbeitung von Programmteilen in LabVIEW sollte nur in Ausnahmefällen durch Sequenzstrukturen unterbunden werden. Sequenzstrukturen stellen zwar die Ausführungsreihenfolge sicher, machen jedoch parallele Operationen unmöglich. So könnten dann beispielsweise asynchrone Operationen, bei denen I/O-Geräte wie PXI-, GPIB-, DAQ-Geräte und serielle Ports verwendet werden, nicht gleichlaufend mit anderen Operationen ausgeführt werden. Bei der Verwendung von Sequenzstrukturen werden Abschnitte des Blockdiagramms ausgeblendet und der natürliche Datenfluss von links nach rechts wird unterbrochen.

Um die Ausführungsreihenfolge zu steuern, empfiehlt es sich, zwischen Knoten eine Datenabhängigkeit herzustellen. Dazu können zum Beispiel die Fehlerein- und -ausgänge verwendet werden, zu denen Sie im Kapitel 6, *Ausführen von VIs und Fehlersuche* im Abschnitt *Fehlerbehandlung* weitere Hinweise finden.

Wenn ein Anzeigeelement in verschiedenen Rahmen der Sequenzstruktur aktualisiert werden soll, sollte ebenfalls auf Sequenzstrukturen verzichtet werden. Beispiel: Ein VI, das in einer Testapplikation verwendet wird, verfügt über ein **Status**-Anzeigeelement, das den Namen des aktuellen Tests anzeigt. Wenn für jeden Test ein SubVI vorhanden ist, das von einem anderen Rahmen aus aufgerufen wird, kann das Anzeigeelement nicht in jedem Rahmen aktualisiert werden, und es entsteht eine ungültige Verbindung wie in Abbildung 8-6.

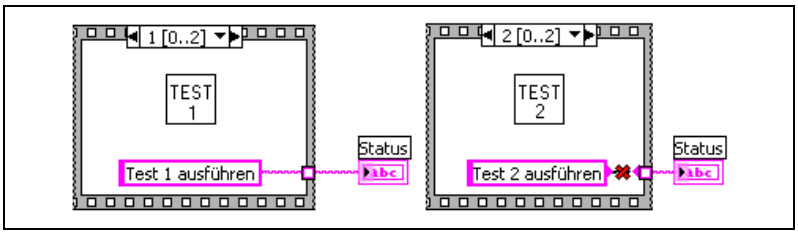


Abbildung 8-6. Aktualisieren eines Anzeigeelements von verschiedenen Rahmen einer gestapelten Sequenzstrukturen aus

Da alle Rahmen einer Sequenz-Struktur ausgeführt werden, bevor die Daten die Struktur verlassen, kann dem **Status**-Anzeigeelement nur von einem Rahmen ein Wert zugewiesen werden.

Daher sollten für eine solche Aufgabe stattdessen eine Case-Struktur und eine While-Schleife eingesetzt werden, wie in Abbildung 8-7 dargestellt.

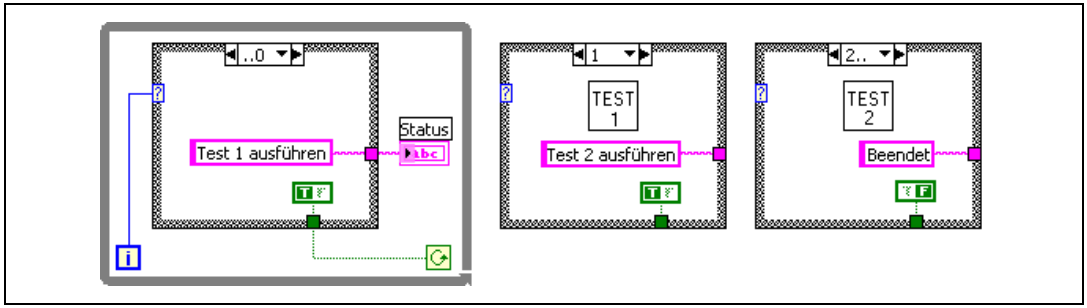


Abbildung 8-7. Aktualisieren eines Anzeigeelements von verschiedenen Cases einer Case-Struktur aus

Jeder Case in einer Case-Struktur entspricht einem Rahmen der Sequenzstruktur. Mit jedem Durchlauf der While-Schleife wird der nächste Case ausgeführt. Das **Status**-Anzeigeelement zeigt bei jedem Case den Status des VIs an. Das **Status**-Anzeigeelement wird im Case vor demjenigen Case aktualisiert, der das entsprechende SubVI aufruft, da die Daten die Struktur verlassen, nachdem jeder Case ausgeführt wurde.

Anders als bei Sequenzstrukturen kann bei Case-Strukturen in jedem Case eine Datenausgabe erfolgen, mit der die While-Schleife beendet werden kann. Wenn beispielsweise beim Ausführen des ersten Tests ein Fehler auftritt, kann die Case-Struktur an den Bedingungsanschluss den Wert FALSE übergeben, um die Schleife zu beenden. Bei einer Sequenzstruktur müssten auch im Fehlerfall alle Rahmen ausgeführt werden.

Sequenzstrukturen ersetzen

Um eine flache in eine gestapelte Sequenzstruktur umzuwandeln, klicken Sie diese mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Durch gestapelte Sequenz ersetzen**. Um eine gestapelte Sequenzstruktur in eine flache umzuwandeln, verfahren Sie analog und wählen die Option **Ersetzen»Durch flache Sequenz ersetzen**.

9

Ereignisgesteuerte Programmierung

LabVIEW ist eine datenflussabhängige Programmierumgebung, das heißt, die Ausführung des Blockdiagramms wird durch den Datenfluss bestimmt. Bei der ereignisgesteuerten Programmierung kann die Ausführung des Blockdiagramms durch weitere Faktoren gesteuert werden, wie zum Beispiel einen Eingriff des Anwenders über das Frontpanel oder andere asynchrone Vorgänge.

Hinweis Die ereignisgesteuerte Programmierung ist nur mit dem LabVIEW Full bzw. Professional Development System möglich. Ein VI, das mit den entsprechenden Funktionen erstellt wurde, kann zwar auch auf einem System, auf dem das LabVIEW Base Package läuft, ausgeführt werden, jedoch können an den Komponenten zur Ereignisverarbeitung keine Modifikationen vorgenommen werden.

Weitere Informationen ...

Weitere Hinweise zur Verwendung von Ereignissen in Applikationen finden Sie in der *LabVIEW-Hilfe*.

Was sind Ereignisse?

Ein Ereignis ist eine asynchrone Meldung darüber, dass etwas stattgefunden hat. Ein Ereignis kann über die Benutzeroberfläche, eine externe Ein- und Ausgabe oder über andere Programmbestandteile hervorgerufen werden. Ereignisse der Benutzeroberfläche sind zum Beispiel Mausklicks oder Tastenbetätigungen. Zu den Ereignissen der externen I/O zählen zum Beispiel Hardwaretakte oder -trigger, die erzeugt werden, wenn die Datenerfassung beendet oder ein Fehler aufgetreten ist. Ereignisse können aber auch programmatisch erzeugt und zur Kommunikation mit verschiedenen Programmteilen verwendet werden. In LabVIEW werden nur programmatisch erzeugte bzw. an der Benutzeroberfläche auftretende Ereignisse unterstützt, jedoch keine Ereignisse, die über externe I/O ausgelöst werden.

Bei einem ereignisgesteuerten Programm haben Ereignisse im System direkten Einfluss auf den Ausführungsablauf, während bei verfahrensmäßig ablaufenden Programmen eine vorbestimmte Abarbeitungsreihenfolge eingehalten werden muss. Ereignisgesteuerte Programme enthalten gewöhnlich eine Schleife, die auf das Ereignis wartet, bei dessen Eintreten einen bestimmten Blockdiagrammabschnitt ausführt und sich anschließend wiederholt, um auf das nächste Ereignis zu warten. Die jeweilige Reaktion auf ein Ereignis ist von dem dafür entwickelten Programmcode abhängig. Die Reihenfolge, in der ein ereignisgesteuertes Programm ausgeführt wird, hängt davon ab, welche Ereignisse in welcher Reihenfolge auftreten. So kann es zum Beispiel vorkommen, dass bestimmte Programmteile sehr oft ausgeführt werden, da mit ihnen häufig auftretende Ereignisse bearbeitet werden, während andere Programmteile unter Umständen nie ausgeführt werden.

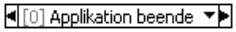
Welchen Vorteil bieten Ereignisse?

Mit Frontpanel-Ereignissen kann die Ausführung des Blockdiagramms getaktet werden. Das heißt, jedesmal, wenn der Anwender auf bestimmte Weise Veränderungen im Frontpanel vornimmt, wird die Ausführung eines bestimmten Cases ausgelöst, mit dem das Ereignis verarbeitet wird. Ansonsten müsste der jeweilige Zustand der Frontpanel-Objekte in einer Schleife kontinuierlich abgefragt und auf diese Weise festgestellt werden, ob eine Veränderung aufgetreten ist. Dadurch benötigt die CPU jedoch sehr viel Zeit, und es besteht die Gefahr, dass zu schnelle Änderungen nicht erkannt werden. Mit benutzergesteuerten Ereignissen kann direkt auf bestimmte Eingriffe des Anwenders reagiert werden, ohne dazu das Frontpanel kontinuierlich abfragen zu müssen, da das Blockdiagramm jedesmal informiert wird, wenn ein bestimmter Vorgang stattgefunden hat. Bei ereignisgesteuerten Programmen verringert sich die CPU-Auslastung, das Blockdiagramm wird vereinfacht, und es wird sichergestellt, dass das Programm tatsächlich auf alle Aktionen des Anwenders reagiert.

Mit programmatisch erzeugten Ereignissen kann zwischen verschiedenen Programmteilen kommuniziert werden, zwischen denen keine Datenabhängigkeit besteht. Programmatisch erzeugte Ereignisse haben zum größten Teil dieselben Vorzüge wie benutzergesteuerte Ereignisse. Darüber hinaus kann für beide Typen derselbe Programmcode verwendet werden. Dadurch ist es einfach, komplexere Architekturen wie zum Beispiel “Queued State Machines”, also über Queues (Warteschlangen) gesteuerte Case-Strukturen, zu implementieren.

Komponenten für Ereignisstrukturen

Zur Verarbeitung von Ereignissen in einer Applikation sind Ereignisstrukturen (siehe links) zu verwenden. In Ereignisstrukturen ist das VI “Auf Meldung warten” integriert. Ansonsten arbeiten diese wie Case-Strukturen. Eine Ereignisstruktur kann mehrere Cases zur Verarbeitung verschiedener Ereignisse haben. Es ist sogar möglich, einen Case so zu definieren, dass mehrere Ereignisse stattfinden können. Jedoch darf immer nur jeweils ein Ereignis auftreten. Nach dem Start einer Ereignisstruktur wird auf eines der konfigurierten Ereignisse gewartet und bei dessen Eintreten wird der entsprechende Case ausgeführt. Nach der Verarbeitung eines Ereignisses wird die Ausführung der Struktur beendet. Sie läuft also nicht automatisch weiter, um mehrere Ereignisse zu verarbeiten. Genau wie bei der Funktion “Auf Meldung warten” kann auch bei Ereignisstrukturen eine Zeitüberschreitung eintreten. In diesem Fall wird ein bestimmter Timeout-Case ausgeführt.



In der Beschriftung des Ereignis-Selektors, wie links dargestellt, wird angezeigt, für welches Ereignis der jeweils angezeigte Case spezifiziert worden ist. Zur Anzeige der anderen Ereignis-Cases ist der Pfeil nach unten neben der Beschriftung anzuklicken und ein Case aus dem Kontextmenü auszuwählen.

(Windows) Sie können auch den Cursor über die Selektorbeschriftung bewegen und dabei die Taste <Strg> drücken. **(UNIX)** Drücken Sie die <Meta>-Taste.

Über den Anschluss “Ereignis-Timeout” an der oberen linken Ecke der Ereignisstruktur (siehe Abbildung links) wird festgelegt, wie lange auf das Auftreten eines Ereignisses gewartet werden soll. Die Angabe erfolgt in Millisekunden. Voreinstellung ist –1, das heißt, es besteht keine Zeitbegrenzung. Wenn der Anschluss mit einem Wert verbunden wird, muss immer ein Timeout-Case erstellt werden.

Ein Ereignisdatenknoten wie in der Abbildung dargestellt arbeitet ähnlich wie die Funktion “Nach Namen aufschlüsseln”. Ein solcher Knoten befindet sich jeweils an der rechten und linken Innenseite eines Ereignis-Cases. Er dient dazu, die Werte zu erkennen, die beim Eintreten eines Ereignisses von LabVIEW an die Struktur übergeben werden. Datenknoten können um weitere Datenelemente erweitert werden, indem sie nach oben oder unten aufgezogen werden. Jedes aufgeführte Datenelement kann für jedes Ereignisdatenelement verwendet werden. Je nachdem, für welche Ereignisse der jeweilige Case definiert ist, werden im Knoten unterschiedliche Daten angezeigt. Wenn ein Case so angelegt wurde, dass er zur Handhabung meh-

rerer Ereignisse in der Lage ist, werden im Knoten nur die Daten angezeigt, die für alle Ereignisse des entsprechenden Cases gelten.

Ereignisfilterknoten (wie in der Abbildung links) funktionieren ähnlich wie Ereignisdatenknoten. Ein solcher Knoten befindet sich jeweils an der Innenseite des rechten Randes eines Filterereignis-Cases. Der Knoten erkennt den Anteil der im Ereignis-Datenknoten verfügbaren Daten, die durch den Ereignis-Case modifiziert werden können. Abhängig davon, welche(s) Ereignis(se) der jeweilige Case verarbeiten kann, werden in einem solchen Knoten verschiedene Daten angezeigt. Per Voreinstellung werden diese an derselben Stelle wie die entsprechenden Datenelemente bei Ereignisdatenknoten angezeigt. Unverbundene Datenelement werden nicht verändert. Weitere Informationen zu Filterereignissen finden Sie im Abschnitt *Melder- und Filterereignisse* dieses Kapitels.

Zur Anzeige der Anschlüsse für dynamische Ereignisse (siehe Abbildung links) klicken Sie mit der rechten Maustaste auf die Ereignisstruktur und wählen die Option **Anschlüsse für dynamische Ereignisse anzeigen** aus. Diese dienen nur zur Registrierung dynamischer Ereignisse. Weitere Informationen zur Verwendung dieser Anschlüsse befinden sich in den Abschnitten *Dynamische Ereignisregistrierung* und *Dynamische Umregistrierung* dieses Kapitels.

Hinweis Genau wie bei Case-Strukturen können auch bei Ereignisstrukturen Tunnel verwendet werden. Standardmäßig müssen die Ausgabekanäle hier jedoch nicht in jedem Case verbunden sein. Bei allen unverbundenen Tunneln werden die voreingestellten Werte für den jeweiligen Datentyp verwendet. Um die Struktur so einzustellen, dass Tunnel generell verbunden werden müssen, klicken Sie mit der rechten Maustaste auf einen Tunnel und deaktivieren Sie im Kontextmenü die Option **Standardwert für offene Verbindung**.

Nähere Informationen zu den voreingestellten Werten für bestimmte Datentypen befinden sich in der *LabVIEW-Hilfe*.

Melder- und Filterereignisse

Es gibt zwei Typen von Ereignissen, die an der Benutzeroberfläche auftreten: Melder und Filter.

Über Meldereregnisse wird LabVIEW informiert, wenn eine Aktion bereits stattgefunden hat, das heißt, wenn beispielsweise der Benutzer den Wert eines Bedienelements schon geändert hat. Mit Meldereregnissen kann also auf ein Ereignis nach dessen Auftreten reagiert werden. Ein Meldereregnis an einem bestimmten Objekt kann dabei von beliebig vielen Ereignisstrukturen verarbeitet werden. Beim Auftreten eines Ereignisses

wird an jede Ereignisstruktur eine Kopie des Ereignisses übergeben, so dass es von den verschiedenen Strukturen parallel verarbeitet werden kann.

Mit Filterereignissen werden Sie darüber in Kenntnis gesetzt, wenn eine Anwenderinteraktion stattgefunden hat, bevor diese von LabVIEW verarbeitet wird. So können Sie festlegen, wie das Programm auf verschiedene Ereignisse reagieren soll. Demnach bieten Filterereignisse die Möglichkeit zur Einflussnahme auf die Verarbeitung von Ereignissen, so dass diese unter Umständen auch anders verarbeitet werden können als standardmäßig eingestellt. Im Filterereignis-Case einer Ereignisstruktur können die Daten zum jeweiligen Ereignis vor dessen Verarbeitung bestätigt oder verändert werden. Ein Ereignis kann auch verworfen werden, so dass es keinerlei Einfluss auf das VI hat. Demnach kann eine Ereignisstruktur zum Beispiel so konfiguriert werden, dass das Ereignis “Panel schließen?” verworfen wird. Dadurch kann der Benutzer das Frontpanel des VIs nicht interaktiv schließen. Die Bezeichnungen von Filterereignissen enden immer mit einem Fragezeichen, wie zum Beispiel “Panel schließen?”, um sie von Melderereignissen zu unterscheiden. Zu den meisten Filterereignissen gibt es ein entsprechendes Melderereignis (ohne Fragezeichen). Dieses wird nach dem Filterereignis erzeugt, wenn das Ereignis in keinem Case verworfen wurde.

Genau wie bei Melderereignissen kann ein und dasselbe Filterereignis an einem bestimmten Objekt von beliebig vielen Ereignisstrukturen verarbeitet werden. Die Filterereignisse werden jedoch der Reihe nach an die darauf konfigurierten Strukturen weitergeleitet. Die Weiterleitung erfolgt dabei in der Reihenfolge, in der die Ereignisse registriert wurden. Weitere Informationen zur Registrierung von Ereignissen finden Sie im Abschnitt *Verwenden von Ereignissen in LabVIEW* dieses Kapitels. In jeder Ereignisstruktur muss zunächst der Ereignis-Case abgeschlossen werden, bevor eine Meldung an die nächste Ereignisstruktur erfolgen kann. Die Übergabe an die nächste Struktur erfolgt auch dann, wenn die Ereignisdaten in einer Struktur geändert wurden. Wenn ein Ereignis jedoch verworfen wurde, wird es nicht weitergeleitet. Die Verarbeitung der Anwenderaktion, die das Ereignis ausgelöst hat, wird erst beendet, wenn das Ereignis von keiner der festgelegten Ereignisstrukturen verworfen wurde.

Hinweis Es wird empfohlen, Filterereignisse immer nur dann zu verwenden, wenn die Verarbeitung der Anwenderaktion aktiv beeinflusst werden soll, indem das Ereignis entweder verworfen wird oder die entsprechenden Daten modifiziert werden. Um zu ermitteln, ob eine bestimmte Aktion vorgenommen wurde, sollten statt dessen Melderereignisse verwendet werden.

Die Cases einer Ereignisstruktur, mit denen Filterereignisse verarbeitet werden, enthalten immer einen Ereignisfilterknoten, wie im Abschnitt *Komponenten für Ereignisstrukturen* dieses Kapitels beschrieben. Wenn die Anschlüsse dieses Knotens verbunden werden, können damit die Ereignisdaten verändert werden. Ansonsten bleiben alle Datenelemente unverändert. Um ein Ereignis vollständig zu verwerfen, ist der Anschluss **Verwerfen?** mit dem Wert TRUE zu verbinden.

Hinweis Es ist nicht möglich, mit einem einzelnen Case sowohl Melder- als auch Filterereignisse zu verarbeiten. Es können jedoch mit einem Case mehrere Meldereignisse verarbeitet werden. Bei Filterereignissen ist das nur möglich, wenn die Daten zu allen Ereignissen übereinstimmen.

Verwenden von Ereignissen in LabVIEW

In LabVIEW können viele verschiedene Ereignisse erzeugt werden. Um jedoch unerwünschte Ereignisse zu vermeiden, sollte mit Hilfe der Registrierung festgelegt werden, über welche Ereignisse eine Meldung erfolgen soll. Die Ereignisregistrierung kann entweder statisch oder dynamisch erfolgen.

Bei der statischen Registrierung kann festgelegt werden, welche der am Frontpanel eines VIs auftretenden Ereignisse in welchem Case der Ereignisstruktur auf dem Blockdiagramm verarbeitet werden sollen. Die entsprechenden Ereignisse werden bei der Ausführung des VIs automatisch erfasst. Jedes Ereignis steht entweder mit einem Frontpanel-Bedienelement des VIs, dem Frontpanel-Fenster als solchem oder der LabVIEW-Applikation in Zusammenhang. Es ist jedoch nicht möglich, eine Ereignisstruktur so zu spezifizieren, dass Ereignisse am Frontpanel eines anderen VIs ausgeführt werden können. Diese Art von Konfiguration wird als “statisch” bezeichnet, da während der Ausführung des VIs nicht mehr beeinflusst werden kann, welche Ereignisse von der Struktur verarbeitet werden sollen. Weitere Informationen zur statischen Registrierung finden Sie im Abschnitt *Statische Registrierung von Ereignissen* dieses Kapitels.

Bei der dynamischen Registrierung von Ereignissen werden dagegen VI-Server eingesetzt, so dass Objekte für die Ereignisse erstellt werden sollen, mit Hilfe von Applikations-, VI- und Bedienelement-Referenzen auch während der Ausführung spezifiziert werden können. Dadurch ist eine flexiblere Kontrolle darüber möglich, welche Ereignisse in LabVIEW wann erzeugt werden. Die dynamische Ereignisregistrierung ist jedoch komplexer als die statische, da dazu der Umgang mit VI-Server-Referenzen und speziellen Blockdiagrammfunktionen zur Registrierung bzw. zum Löschen der Registrierung für Ereignisse erforderlich ist, während bei der statischen

Registrierung die Spezifikationen in der Ereignisstruktur automatisch verarbeitet werden. Nähere Hinweise zur dynamischen Registrierung befinden sich in Abschnitt *Dynamische Ereignisregistrierung* dieses Kapitels.

Hinweis Grundsätzlich gilt, dass Ereignisse an der Benutzeroberfläche immer das Ergebnis einer Anwenderinteraktion auf dem Frontpanel sind. Ereignisse wie “Wertänderung” werden daher nicht erzeugt, wenn der VI-Server, DataSocket oder globale bzw. lokale Variablen verwendet werden. Wie das Ereignis “Wertänderung” programmatisch erzeugt werden kann, wird in der *LabVIEW-Hilfe* unter der Beschreibung zur Eigenschaft “Wert (Ereignis auslösend)”. Programmatisch ausgelöste Ereignisse werden häufig anstelle von Queues oder Meldern verwendet.

Zu den Ereignisdaten in LabVIEW gehören immer ein Zeitstempel, eine Enum zur Anzeige des jeweiligen Ereignisses und eine VI-Server-Referenz zu dem Objekt, von dem es ausgelöst wurde. Mit Zeitstempeln werden Millisekunden gezählt. Sie dienen dazu, die Zeit zwischen zwei Ereignissen zu berechnen oder die Reihenfolge zu bestimmen, in der bestimmte Ereignisse auftreten. Die Referenz auf das Objekt, an dem das Ereignis ausgelöst wird, ist strikt auf die entsprechende VI-Server-Klasse bezogen. Je nach auslösendem Objekt sind die Ereignisse in Klassen wie “Applikation”, “VI” oder “Bedienelement” untergliedert. Wenn ein einzelner Case mehrere Ereignisse für Objekte unterschiedlicher VI-Server-Klassen abhandelt, ist der Referenztyp die gemeinsame Eltern-Klasse aller Objekte. Wenn zum Beispiel ein einzelner Case in der Ereignisstruktur so konfiguriert wird, dass er auf Ereignisse eines numerischen Bedienelements und eine Farbrampen-Bedienenlement reagiert, ist der Typ des Bedienelements der Ereignisquelle numerisch, da beide Elemente zur Klasse “numerisch” gehören. Weitere Informationen zu VI-Server-Klassen befinden sich in Kapitel 17, *Programmatische Steuerung von VIs*.

Hinweis Wenn ein Ereignis sowohl in der VI- als auch in der Bedienelement-Klasse registriert ist, wird das VI-Ereignis zuerst erzeugt. Bei Bedienelement-Referenzen werden zunächst Ereignisse für Container-Objekte (wie Cluster) und anschließend für die darin enthaltenen Elemente erzeugt. Wenn ein Ereignis zu einem Container-Objekt von der Struktur verworfen wird, werden zu den darin enthaltenen Elemente jedoch keine Ereignisse ausgelöst.

Zu jeder Ereignisstruktur und jedem Knoten zur Registrierung für Ereignisse gehört eine Queue (Warteschlange), in der die Ereignisse gespeichert werden. Beim Eintreten eines Ereignisses wird in jede darauf registrierte Queue eine Kopie gespeichert. In einer Ereignisstruktur werden alle Ereignisse verarbeitet, die sich in der dazugehörigen Queue befinden und in den Queues der mit den Anschlüssen für dynamische Ereignisse verbundenen

Knoten zur Ereignisregistrierung. Mit Hilfe der Queues wird sichergestellt, dass alle Ereignisse zuverlässig in der Reihenfolge ihres Auftretens an die entsprechende Ereignisstruktur übergeben werden.

Per Voreinstellung wird das Frontpanel so lange gesperrt, bis ein Ereignis an einem darauf befindlichen Objekt verarbeitet worden ist. Während dessen erfolgt keine Reaktion auf Frontpanel-Aktivitäten. Diese werden jedoch zur anschließenden Verarbeitung in einem Puffer gespeichert. Es ist jedoch weiterhin möglich, das Fenster zu verschieben, Bildlaufleisten oder die Schaltfläche **Abbrechen** zu betätigen. Um die Frontpanel-Sperre bei Melderereignissen aufzuheben, ist die entsprechende Option im Dialogfeld **Ereignisse bearbeiten** zu deaktivieren. Bei Filterereignissen muss die Sperre aktiviert sein, damit der interne Zustand von LabVIEW bis zur vollständigen Bearbeitung des aktuellen Ereignisses nicht verändert wird.

Ereignisse können in LabVIEW aber auch dann ausgelöst werden, wenn keine Ereignisstruktur zu deren Verarbeitung vorhanden ist. Da bei jeder Ausführung einer Ereignisstruktur immer nur ein Ereignis bearbeitet werden kann, ist es sinnvoll, diese zur Verarbeitung aller auftretenden Ereignisse in eine While-Schleife zu setzen.

Vorsicht! Wenn keine Ereignisstruktur ausgeführt wird und die Frontpanel-Sperre aktiviert ist, reagiert das VI nicht mehr. In diesem Fall ist das VI über die Schaltfläche **Abbrechen** zu beenden. Um die Frontpanel-Sperre aufzuheben, klicken Sie die Ereignisstruktur mit der rechten Maustaste an und deaktivieren Sie im Dialogfeld **Ereignisse bearbeiten** die Option **Frontpanel bis zum Ausführungsende des Ereignis-Case sperren**. Bei Filterereignissen kann die Sperre jedoch nicht aufgehoben werden.

In der *LabVIEW-Hilfe* finden Sie Warnungen zur Verwendung der Ereignissteuerung und alle konfigurierbaren Ereignisse.

Statische Registrierung von Ereignissen

Die statische Registrierung ist nur für Ereignisse an der Benutzeroberfläche verfügbar. Eine Ereignisstruktur zur Verarbeitung eines statisch registrierten Ereignisses lässt sich über das Dialogfeld **Ereignisse bearbeiten** konfigurieren. Wählen Sie dazu die Quelle aus, an der das Ereignis erwartet wird. Dabei kann es sich um die Applikation, das VI oder ein einzelnes Bedienelement handeln. Wählen Sie anschließend die Art des Ereignisses aus, zum Beispiel “Größenänderung des Frontpanels” oder “Wertänderung”. Bearbeiten Sie den Ereignis-Case dem jeweiligen Verwendungszweck entsprechend. Weitere Informationen zum Dialogfeld **Ereignisse bearbeiten** und zur statischen Registrierung von Ereignissen finden Sie in der *LabVIEW-Hilfe*.

Wenn ein VI mit einer Ereignisstruktur ausgeführt wird, erfolgt die statische Erfassung der Ereignisse automatisch und auf nachvollziehbare Art und Weise. Ereignisse können in VIs nur dann auftreten, wenn diese gerade ausgeführt werden oder das entsprechende VI von einem ausgeführten VI als SubVI aufgerufen wird.

Wenn ein VI ausgeführt wird, dann wird es zusammen mit den im Blockdiagramm enthaltenen SubVIs in den Ausführungszustand “reserviert” versetzt. Das VI lässt sich dann weder bearbeiten, noch kann die Schaltfläche **Ausführen** betätigt werden, da es während der Ausführung des übergeordneten VIs jederzeit als SubVI aufgerufen werden kann. Wenn ein VI in den reservierten Zustand versetzt wird, werden alle Ereignisse, die über Ereignisstrukturen auf dem Blockdiagramm statisch konfiguriert wurden, automatisch erfasst. Bei Ausführungsende des Haupt-VIs wird es zusammen mit allen dazugehörigen SubVIs in den Leerlaufzustand versetzt und die Registrierung der Ereignisse wird automatisch aufgehoben.

Weitere Beispiele zur statischen Registrierung von Ereignissen befinden sich unter `examples\general\uievents.llb`.

Dynamische Ereignisregistrierung

Bei der dynamischen Registrierung von Ereignissen haben Sie die vollständige Kontrolle darüber, welche Ereignisse wann erzeugt werden. Die dynamische Registrierung von Ereignissen empfiehlt sich dann, wenn nur ein Teil einer Applikation ereignisgesteuert sein soll oder während der Ausführung andere VIs oder Bedienelemente für Ereignisse konfiguriert werden sollen. Bei der dynamischen Registrierung können Ereignisse auch in SubVIs verarbeitet werden und nicht nur in dem VI, in dem sie auftreten.

Zur Verarbeitung dynamisch registrierter Ereignisse sind die folgenden vier Schritte erforderlich:

1. Ermitteln der VI-Server-Referenzen für die Objekte, die auf Ereignisse konfiguriert werden sollen.
2. Registrieren der Objekte, durch Verbinden der VI-Server-Referenzen mit dem Knoten zur Registrierung von Ereignissen.
3. Platzieren der Ereignisstruktur in eine While-Schleife, damit alle Ereignisse an den Objekten verarbeitet werden, bis die Bedingung zur Beendigung der Schleife erfüllt ist.
4. Mit der Funktion “Registrierung für Ereignisse aufheben” wird die Ereignissteuerung beendet.

Für die dynamische Registrierung eines Ereignisses ist zunächst eine VI-Server-Referenz notwendig. Dazu sollten die Funktionen “Applikationsreferenz öffnen” und “VI-Referenz öffnen” verwendet werden. Um eine Referenz zu einem Bedienelement zu erhalten, wählen Sie einen Eigenschaftsknoten zur Ermittlung aller Bedienelemente eines VIs oder klicken Sie das Element mit der rechten Maustaste an und wählen Sie zur Erstellung einer entsprechenden Konstante aus dem Kontextmenü die Option **Erstelle»Referenz**. Nähere Einzelheiten zu VI-Server-Referenzen befinden sich im Abschnitt *Applikations- und VI-Referenzen* des Kapitels 17, *Programmatische Steuerung von VIs*.

Um Elemente dynamisch für Ereignisse zu konfigurieren, wählen Sie die Funktion “Für Ereignisse registrieren”. Dieser Knoten zeigt standardmäßig ein Ereignis an, kann aber um weitere Ereignisse erweitert werden. Jeder Eingang “Ereignis-Quelle” kann nun mit einer Applikations-, VI- oder Bedienelement-Referenz verbunden werden. Klicken Sie dann jeden Eingang mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü über die Option **Ereignisse** aus, für welches Ereignis das jeweilige Objekt konfiguriert werden soll. Die zur Auswahl verfügbaren Ereignisse hängen vom Typ der VI-Server-Referenz ab, die mit dem Eingang “Referenz-Quelle” verbunden wurde. Die im Kontextmenü angezeigten Ereignisse sind dieselben wie die, die bei der statischen Registrierung im Dialogfeld **Ereignisse bearbeiten** angezeigt werden. Bei Ausführung des Knotens werden die Objekte, die an den Eingängen “Ereignis-Quelle” anliegen, für die angegebenen Ereignisse registriert. Sobald ein Objekt registriert ist, werden alle entsprechenden Ereignisse in einer Queue gespeichert, um dann von einer Ereignisstruktur abgearbeitet zu werden. Alle vor der Ausführung des Knotens auftretenden Ereignisse sind davon ausgenommen; es sei denn, ein anderes Objekt ist vorher darauf registriert gewesen.

Hinweis Anders als bei Eigenschaftsknoten muss bei Registrierungsknoten der obere linke Eingang nicht verbunden werden. Dieser Eingang ist nur notwendig, wenn eine vorhandene Registrierung geändert werden soll. Weitere Hinweise zur Umregistrierung von Objekten auf andere Ereignisse befinden sich im Abschnitt *Dynamische Umregistrierung* dieses Kapitels.

Wenn Sie eine Ereignisstruktur mit der rechten Maustaste anklicken und die Option **Anschlüsse für dynamische Ereignisse anzeigen** auswählen, werden die Anschlüsse für dynamische Ereignisse sichtbar. Diese verhalten sich ähnlich wie Schieberegister. Am linken Anschluss kann entweder eine einzelne RefNum zur Ereignisregistrierung oder ein entsprechender RefNum-Cluster anliegen. Wenn an der linken Seite des rechten Anschlusses keine Verbindung vorliegt, enthält er dieselben Werte wie der linke

Anschluss. Um die Konfiguration auf ein Ereignis dynamisch zu verändern, kann die RefNum zur Ereignisregistrierung oder der entsprechende Cluster über einen Knoten zur Ereignisregistrierung mit der Innenseite des rechten Anschlusses verbunden werden. Weitere Informationen zu den Anschlüssen für dynamische Ereignisse befinden sich in Abschnitt *Dynamische Umregistrierung* dieses Kapitels.

Bei dem Ausgang eines Knotens zur Ereignisregistrierung handelt es sich um eine RefNum zur Registrierung für Ereignisse. Dieser strikte Datentyp wird in LabVIEW für die Daten verwendet, die an eine Ereignisstruktur übergeben werden. Wenn Sie den Cursor über die RefNum bewegen, werden die registrierten Ereignisse in der **Kontext-Hilfe** angezeigt. Nachdem ein Knoten zur Registrierung für Ereignisse konfiguriert worden ist, verbinden Sie den Ausgang **Ereignis-Registrierung (RefNum)** mit dem linken Anschluss für dynamische Ereignisse der Struktur und legen fest, wie die Ereignisse verarbeitet werden sollen. Eine von einer Ereignis-RefNum ausgehende Verbindung sollte niemals abgezweigt werden, da sonst möglicherweise eine Queue von mehreren Strukturen ausgelesen wird und somit eine Race Condition eintritt (das heißt, die Strukturen konkurrierend auf dieselbe Ressource zugreifen), was zu einem unvorhersehbaren Verhalten der Applikation führen kann.

Zur Konfiguration einer Ereignisstruktur zur dynamischen Verarbeitung von Ereignissen verwenden Sie das Dialogfeld **Ereignisse bearbeiten**. Die dynamisch registrierten Ereignisquellen sind im Bereich **Ereignis-Quellen** unter **dynamisch** aufgeführt. Die Namen der Ereignisquellen stimmen immer mit den Referenzen überein, die mit dem Knoten zur Registrierung für Ereignisse verbunden worden sind und sind auch in derselben Reihenfolge aufgelistet. Hier kann nun die gewünschte Ereignisquelle ausgewählt werden. Ereignisse, die bereits über den Knoten zur Ereignisregistrierung konfiguriert wurden, werden im Bereich **Ereignisse** hervorgehoben. Bearbeiten Sie nach der Auswahl des Ereignisses den Case, mit dem das Ereignis verarbeitet werden soll, und zwar dem Anwendungszweck Ihres VIs entsprechend.

Um die Ereignissteuerung zu beenden, verbinden Sie den Ereignis-Anschluss an der rechten Seite der Ereignisstruktur mit dem Eingang **Ereignisregistrierung (RefNum)** mit einer außerhalb der While-Schleife befindlichen Funktion des Typs “Ereignisregistrierung aufheben”. Bei Ausführung dieser Funktion werden alle von der RefNum angegebenen Ereignisse aufgehoben, und die entsprechende Queue wird zusammen mit allen darin befindlichen Ereignissen entfernt. Wenn Sie die Ereignissteuerung nicht aufheben und der Anwender nach Beendigung der While-Schleife Aktionen ausführt, die Ereignisse auslösen, speichert

LabVIEW diese weiterhin kontinuierlich in die Queue. Wenn das Frontpanel bei Ereignissen gesperrt wird, reagiert das VI dann unter Umständen nicht mehr. In diesem Fall wird die Queue erst aufgehoben, wenn das VI im Ruhezustand ist.

Alle konfigurierten Ereignisse werden auch dann aufgehoben, wenn die Ausführung des Haupt-VIs beendet wird. Es wird jedoch insbesondere bei Applikationen mit langen Ausführungszeiten empfohlen, die Registrierung manuell aufzuheben, da so weniger Speicherplatz benötigt wird.

Beispiel für dynamische Ereignissteuerung

Das Blockdiagramm in Abbildung 9-1 zeigt ein Beispiel zur dynamischen Registrierung für Ereignisse. In diesem Fall soll das Ereignis ausgelöst werden, wenn der Cursor in das String-Bedienelement eintritt.

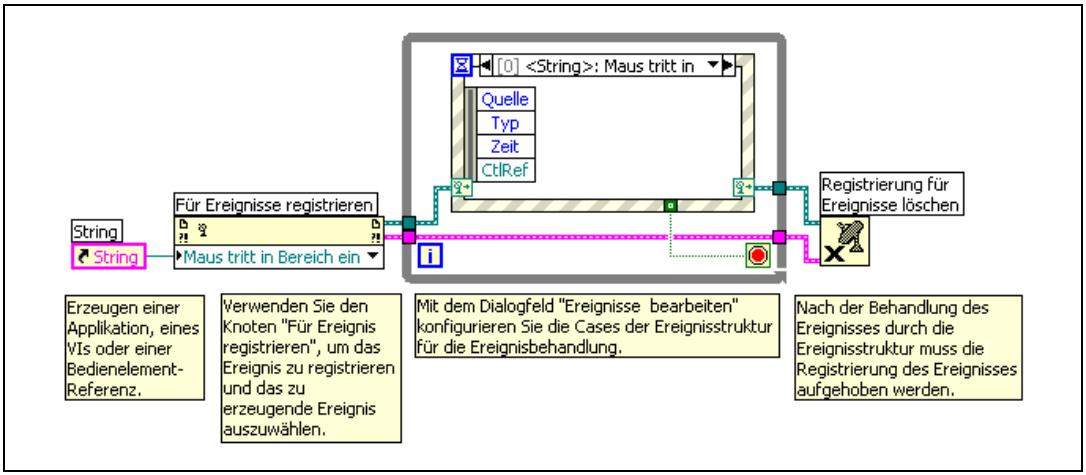


Abbildung 9-1. Dynamische Registrierung für Ereignisse

Weitere Informationen dazu, wie die dynamische Registrierung für Ereignisse vorgenommen wird, befinden sich in der *LabVIEW-Hilfe*.

Weitere Beispiele zu diesem Thema befinden sich unter `examples\general\dynamicevents.llb`.

Dynamische Umregistrierung

Bei der dynamischen Registrierung ist es möglich, während der Ausführung an der Konfiguration der Objekte für Ereignisse Änderungen vorzunehmen oder andere Objekte für bestimmte Ereignisse auszuwählen. Verbinden Sie dazu an einem Knoten zur Ereignisregistrierung den oben

links befindlichen Anschluss **Ereignis-Registrierung (RefNum)**. Der Knoten zeigt dann automatisch dieselben Ereignisse mit denselben Referenztypen wie im Knoten an, mit dem die RefNum ursprünglich erstellt wurde. Solange der Eingang **Ereignis-Registrierung (RefNum)** verbunden ist, kann die Funktion weder neu konfiguriert noch manuell vergrößert oder verkleinert werden.

Wenn an einen Eingang des Typs **Ereignis-Quelle** eine Referenz übergeben wird und der Eingang **Ereignis-Registrierung (RefNum)** ebenfalls verbunden ist, werden alle Referenzen, die zuvor registriert waren, durch die des Knotens ersetzt, mit dem die Registrierung ursprünglich erstellt wurde. Um die Registrierung für ein einzelnes Ereignis aufzuheben, kann der entsprechende **Ereignis-Quelle**-Eingang auch mit einer Konstante des Typs “Keine RefNum” verbunden werden. Bei Eingängen, die nicht verbunden werden, bleibt die Konfiguration unverändert. Die Funktion “Ereignisregistrierung aufheben” dient dazu, alle Ereignisse zu einer bestimmten Referenz aufzuheben.

Im Beispiel in Abbildung 9-2 sehen Sie, wie Ereignisse während der Laufzeit anderen Objekten zugewiesen werden können. Wenn das dargestellte Blockdiagramm ausgeführt wird, registriert LabVIEW die numerische Referenz und wartet darauf, dass am entsprechenden Bedienelement **Numerisch** ein Ereignis auftritt. Sobald an diesem Element eine Wertänderung erfolgt, führt der entsprechende Ereignis-Case einen Knoten zur Ereignisregistrierung aus, um das Bedienelement von **Numerisch** auf **Numerisch 2** zu ändern. Wenn der Anwender anschließend den Wert des Bedienelements **Numerisch** verändert, wird kein entsprechendes Ereignis mehr ausgelöst. Dafür werden nun bei Wertänderungen am Element **Numerisch 2** Ereignisse ausgelöst. Der Knoten zur Ereignisregistrierung wird auch weiterhin bei jeder Wertänderung am Element **Numerisch 2** ausgeführt, was jedoch wirkungslos bleibt, da diesem Element das Ereignis “Wertänderung” bereits zugewiesen worden ist.

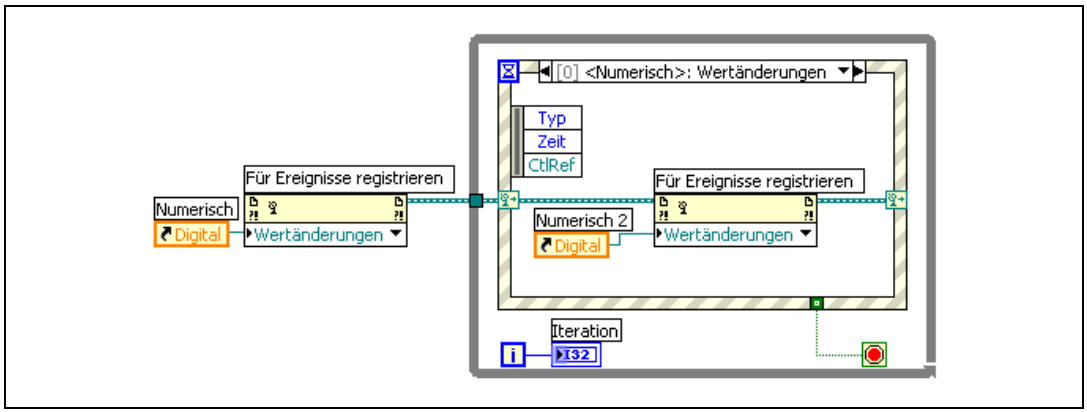


Abbildung 9-2. Dynamische Veränderung der Ereignisregistrierung

Hinweis Es ist nicht möglich, statisch registrierte Ereignisse dynamisch zu verändern.

Benutzerereignisse

Benutzergesteuerte Ereignisse, also Ereignisse mit benutzerdefinierten Werten, können programmatisch erstellt und benannt werden. Genau wie Queues und Melder ermöglichen benutzergesteuerte Ereignisse die asynchrone Kommunikation zwischen verschiedenen Teilen einer Applikation. Sie können in derselben Ereignisstruktur verarbeitet werden wie Ereignisse auf der Benutzeroberfläche.

Erstellen und Registrieren benutzergesteuerter Ereignisse

Um ein Benutzerereignis zu definieren, verbinden Sie ein Frontpanel-Objekt wie beispielsweise einen Frontpanel-Anschluss oder eine Blockdiagrammfunktion mit der Funktion “Benutzerereignis definieren”. Der Datentyp des Benutzerereignisses hängt dabei jeweils vom Datentyp des Objekts ab. Zur Benennung des Ereignisses wird automatisch die Objektbezeichnung verwendet. Wenn es sich bei dem Datentyp um einen Cluster handelt, werden die Ereignisdaten durch Bezeichnung und Typ jedes einzelnen Cluster-Objekts bestimmt. Ansonsten gehört zu dem Ereignis nur ein einziger Wert des jeweiligen Typs. Als Bezeichnung für das Ereignis und den Wert wird der Name auf der Beschriftung des Objekts verwendet.

Beim Ausgabewert des Parameters **Benutzerereignis (Ausgang)** der Funktion “Benutzerereignis erstellen” handelt es sich um eine strikt typisierte RefNum mit Bezeichnung und Datentyp des Benutzerereignisses.

Dieser Ausgang ist mit einem Eingang des Typs **Ereignis-Quelle** eines Knotens zur Ereignisregistrierung zu verbinden.

Benutzerereignisse werden genauso verarbeitet wie dynamisch registrierte Ereignisse an der Benutzeroberfläche. Sie verbinden dazu den Ausgang **Ereignisregistrierung (RefNum)** des Knotens zur Ereignisregistrierung mit dem Anschluss für dynamische Ereignisse der Ereignisstruktur. Um einen Case in der Struktur zur Verarbeitung des entsprechenden Ereignisses zu konfigurieren, verwenden Sie das Dialogfeld **Ereignisse bearbeiten**. Die Bezeichnung des Benutzerereignisses wird im Bereich **Ereignis-Quellen** unter **dynamisch** angezeigt.

Die dazugehörigen Daten erscheinen im Ereignisdatenknoten am linken Rand der Struktur. Benutzerereignisse sind Meldererereignisse und können im gleichen Case wie an der Benutzeroberfläche auftretende oder andere Ereignisse verarbeitet werden.

Auch mit dem Knoten zur Ereignisregistrierung können verschiedene Ereignistypen verbunden werden.

Erzeugen eines Benutzerereignisses

Mit Hilfe der Funktion “Benutzerereignis erzeugen” kann das programmierte Ereignis zusammen mit den dazugehörigen Ereignisdaten über eine entsprechend konfigurierte Ereignisstruktur an andere Teile der Applikation übergeben werden. Für die Funktion ist eine Benutzerereignis-RefNum und ein Ereignisdatenwert erforderlich. Der Wert muss vom selben Datentyp wie das Benutzerereignis sein.

Die Funktion kann nur ein Benutzerereignis erzeugen, wenn dieses nicht registriert ist. Wenn das Ereignis zwar registriert ist, jedoch keine Ereignisstruktur zu dessen Verarbeitung zur Verfügung steht, wird es in eine Queue eingereiht und verbleibt dort so lange, bis eine Ereignisstruktur ausgeführt wird, mit der es verarbeitet werden kann. Ein Objekt kann mehrfach für dasselbe Benutzerereignis registriert werden, indem mehrere Knoten zur Registrierung für Ereignisse verwendet werden. In diesem Fall wird bei jeder Ausführung der Funktion “Benutzerereignis erzeugen” an alle Queues zu Ereignisregistrierungs-RefNums eine Kopie des Ereignisses übergeben.

Hinweis Um eine Benutzerinteraktion auf dem Frontpanel zu simulieren, kann ein Benutzerereignis mit derselben Bezeichnung und demselben Datentyp wie ein vorhandenes Ereignis an der Benutzeroberfläche erstellt werden. So können Sie zum Beispiel ein Benutzerereignis mit dem Namen `Wertänd1` erstellen, indem Sie einen Cluster mit zwei booleschen Feldern erstellen, und zwar `OldVal` und `NewVal`, die dieselben Ereignisdaten-

elemente wie das Ereignis “Wertänderung” an einem booleschen Bedienelement haben. Das simulierte Ereignis `Wertänd1` kann sogar im gleichen Ereignis-Case verarbeitet werden wie ein echtes Ereignis des Typs “Wertänderung (boolesch)”. Die Ereignisstruktur wird dann sowohl ausgeführt, wenn der Knoten zur Erzeugung eines Benutzerereignisses ein Ereignis erzeugt hat, als auch wenn eine tatsächliche Wertänderung am Bedienelement stattfindet.

Registrierung für Benutzerereignisse aufheben

Wenn Benutzerereignisse nicht mehr benötigt werden, sollte die entsprechende Registrierung aufgehoben und das Ereignis entfernt werden. Dazu ist die entsprechende RefNum mit dem Eingang **Benutzerereignis** der Funktion “Benutzerereignis löschen” zu verbinden. Damit die Funktionen auch in der richtigen Reihenfolge ausgeführt werden, sollte der Parameter **Fehler (Ausgang)** der Funktion “Ereignisregistrierung aufheben” mit dem **Fehler (Eingang)** der Funktion “Benutzerereignis löschen” verbunden werden.

Beim Beenden des jeweiligen Haupt-VIs wird die Registrierung aller Ereignisse aufgehoben und Benutzerereignisse werden automatisch gelöscht. Es wird jedoch insbesondere bei Applikationen mit langen Ausführungszeiten empfohlen, manuell die Registrierung aufzuheben und die Ereignisse zu löschen, da so weniger Speicherplatz benötigt wird.

Beispiel für Benutzerereignis

Im Blockdiagramm in Abbildung 9-3 sehen Sie ein Beispiel zur Verwendung von Benutzerereignissen. Die Bezeichnung des Ereignisses, `My Data`, wird über einen konstanten Cluster-Wert angegeben, und der Datentyp über einen String mit der Bezeichnung `String`. Mit dem Knoten zur Ereignisregistrierung erfolgt die Registrierung für das Benutzerereignis zur jeweiligen RefNum. Mit einer Ereignisstruktur in einer While-Schleife wartet das VI darauf, dass das Ereignis eintritt. Parallel zur Ausführung der While-Schleife generiert die Funktion “Benutzerereignis erzeugen” das Ereignis, woraufhin der Benutzerereignis-Case in der Ereignisstruktur ausgeführt wird. Wenn die While-Schleife beendet wird, hebt das VI die Registrierung für das Ereignis auf und löscht es.

Um zu sehen, wie die Ereignisdaten im VI von Knoten zu Knoten übergeben werden, erstellen Sie das VI in Abbildung 9-3 und führen Sie es mit aktivierter Highlight-Funktion aus.

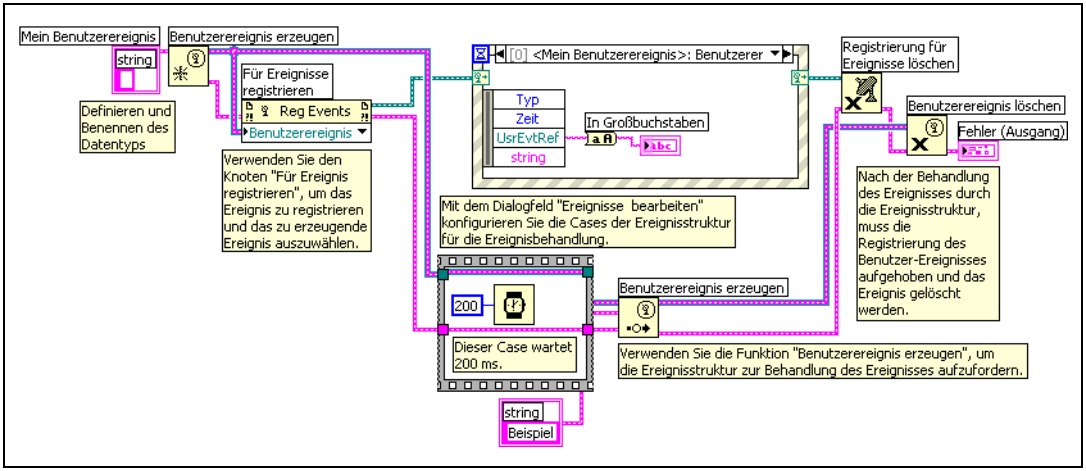


Abbildung 9-3. Erzeugen von Benutzerereignissen

Weitere Beispiele zur dynamischen Registrierung von Ereignissen befinden sich unter `examples\general\dynamicevents.llb`.

10

Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern

Daten können mit Hilfe von Strings, Arrays und Clustern gruppiert werden. Mit Strings werden ASCII-Zeichen gruppiert. Mit Arrays werden Datenelemente des gleichen Typs zusammengefasst. Cluster bieten Ihnen die Möglichkeit, Datenelemente unterschiedlichen Typs zu gruppieren.

Weitere Informationen ...

Weitere Informationen zum Gruppieren von Daten mit Strings, Arrays und Clustern finden Sie in der *LabVIEW-Hilfe*.

Strings

Ein String ist eine Zeichenkette, die darstellbare und nicht darstellbare ASCII-Zeichen umfassen kann. Strings sind plattformunabhängig und werden unter anderem in folgenden Anwendungsbereichen eingesetzt:

- Erstellen von einfachen Textmeldungen.
- Übergabe von numerischer Werte als Zeichenketten an Instrumente und anschließendes Konvertieren der Strings in Zahlen.
- Speichern von numerischen Daten auf Datenträger. Zum Speichern von Zahlen in einer ASCII-Datei müssen Sie die Zahlen zunächst in Strings umwandeln, bevor Sie diese in eine Datei auf dem Datenträger schreiben.
- Erstellen von Anweisungen und Aufforderungen für den Benutzer mit Hilfe von Dialogfeldern.

Auf dem Frontpanel erscheinen Strings als Tabellen, Texteingabefelder und Beschriftungen. Verwenden Sie zum Bearbeiten und Manipulieren von Strings die String-Funktionen im Blockdiagramm. Mit diesen Funktionen werden die Strings für die Verwendung in anderen Applikationen wie Textverarbeitungs- und Tabellenkalkulationsprogrammen oder für den Einsatz in anderen VIs oder Funktionen formatiert.

Strings auf dem Frontpanel

Mit den String-Bedien- und Anzeigeelementen können Texteingabefelder und Beschriftungen nachgeahmt werden. Weitere Informationen zu den String-Bedien- und Anzeigeelementen finden Sie in Abschnitt *String-Bedien- und Anzeigeelemente* des Kapitels 4, *Erstellen des Frontpanels*.

String-Anzeigearten

Klicken Sie mit der rechten Maustaste auf ein String-Bedien- oder Anzeigeelement auf dem Frontpanel, um aus den in Tabelle 10-1 dargestellten Anzeigearten zu wählen. Die Tabelle enthält auch eine Beispielmeldung für jede Anzeigeart.

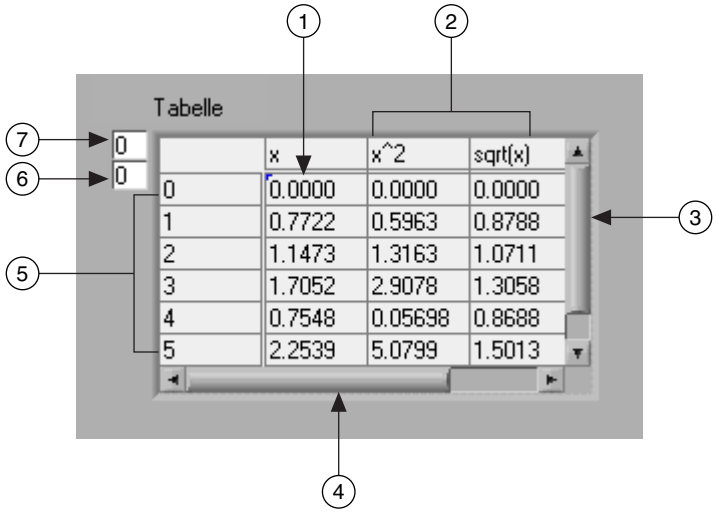
Tabelle 10-1. String-Anzeigearten

Anzeigeart	Beschreibung	Meldung
Normale Anzeige	Zeigt die druckbaren Zeichen unter Verwendung der Schriftart des Bedienelements an. Nicht darstellbare Zeichen werden in diesem Modus meist als Kästchen angezeigt.	`Es gibt vier Anzeigearten.` `\ ist ein Backslash.`
'\' Code-Anzeige	Zeigt Escape-Sequenzen für alle nicht darstellbaren Zeichen an.	`Es\sgibt\svier\sAnzeigearten.` `\n\\\sist\sein\sBackslash.`
Passwort-Anzeige	Zeigt für jedes Zeichen einschließlich Leerzeichen ein Sternchen (`*`) an.	`**********` `**********`

Hex-Anzeige	Zeigt den ASCII-Wert jedes Zeichens in Hexadezimalschreibweise anstelle des Zeichens selbst an.	`4573 2067 6962 7420 7669 6572`
		`2041 6E7A 6569 6765 6172 7465`
		`6E2E 0A5C 2069 7374 2065 696E`
		`2042 6163 6B73 6C61 7368 2E`

Tabellen

Mit Hilfe des Tabellen-Bedienelements kann auf dem Frontpanel eine Tabelle erstellt werden. Jede Zelle in einer Tabelle, die durch eine Zeile und eine Spalte definiert ist, enthält einen String. Daher ist die Tabelle die Anzeigeform für ein 2D-Array aus Strings. In Abbildung 10-1 sehen eine Tabelle mit allen dazugehörigen Bestandteilen. Weitere Informationen zu Arrays finden Sie in Abschnitt *Arrays* dieses Kapitels.



1	Zelle, die mit Indexwerten dargestellt wird	5	Zeilenbeschriftungen
2	Spaltenbeschriftungen	6	horizontaler Index
3	vertikale Bildlaufleiste	7	vertikaler Index
4	horizontale Bildlaufleiste		

Abbildung 10-1. Bestandteile einer Tabelle

Programmgesteuertes Bearbeiten von Strings

Folgende Bearbeitungsvorgänge an Strings sind mit den String-Funktionen möglich:

- Suchen, Abrufen und Ersetzen von Zeichen oder Substrings in einem String.
- Ändern des gesamten Textes in einem String in Groß- oder Kleinbuchstaben.
- Suchen und Abrufen von gleichlautenden Mustern innerhalb eines Strings.
- Abrufen einer Zeile aus einem String.
- Drehen und Umkehren von Text innerhalb eines Strings.
- Verknüpfen von zwei oder mehr Strings.
- Löschen von Zeichen aus einem String.

Beispiele für die Verwendung der String-Funktionen zum Bearbeiten von Strings finden Sie in `examples\general\strings.llb`. Weitere Einzelheiten dazu, wie durch die programmatische Bearbeitung von Strings

eine höhere Speichereffizienz erzielt werden kann, finden Sie auch in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter *Memory and Speed Optimization*.

Formatieren von Strings

Um Daten in einem anderen VI, einer anderen Funktion oder einer anderen Applikation verwenden zu können, müssen sie oftmals in einen String konvertiert werden. Anschließend ist der String so zu formatieren, dass er vom VI, der Funktion oder der Applikation gelesen werden kann. Bei Microsoft Excel muss der String beispielsweise durch Tabulatoren, Kommas oder Leerzeichen begrenzt sein. Damit werden die Zahlenwerte oder Wörter in einzelne Zellen eingeordnet.

Wenn Sie beispielsweise ein aus Zahlen bestehendes 1D-Array mit der Funktion “Datei schreiben” in eine Tabelle schreiben möchten, sollten Sie das Array als String formatieren und jede Zahl mit einem Trennzeichen wie beispielsweise einem Tabulator trennen. Um ein aus Zahlen bestehendes Array mit dem VI “In Spreadsheet-Datei schreiben” in eine Tabelle zu schreiben, müssen Sie das Array mit der Funktion “Array in Tabellen-String” formatieren und ein Format und ein Trennzeichen angeben.

Mit den String-Funktionen sind folgende Operationen möglich:

- Verknüpfen von zwei oder mehr Strings.
- Extrahieren eines Teilstrings aus einem String.
- Umwandeln von Daten in Strings.
- Formatieren eines Strings zur Verwendung in einem Textverarbeitungs- oder Tabellenkalkulationsprogramm.

Mit den VIs und Funktionen zur Datei-I/O können Strings in Text- und Tabellenverarbeitungsdateien gespeichert werden.

Formatbezeichner

Häufig müssen zur Formatierung eines Strings im Parameter **Format-String** Formatbezeichner angegeben werden. Ein Formatbezeichner ist ein Code, der angibt, wie Daten in einen oder aus einem String umgewandelt werden sollen. LabVIEW verwendet Konvertierungscodes, um das Textformat des Parameters festzulegen. Beispielsweise konvertiert der Formatbezeichner `%x` eine hexadezimale Ganzzahl in einen String bzw. umgekehrt.

Bei den Funktionen “In String formatieren” und “In String suchen” können für den Parameter **Format-String** mehrere Formatbezeichner verwendet

werden, und zwar einer für jede Eingabe an die oder Ausgabe aus der erweiterbaren Funktion.

Bei den Funktionen “Array in Tabellenstring” und “Tabellenstring in Array” wird im **Format String**-Parameter nur ein Formatbezeichner verwendet, da diese Funktionen nur jeweils einen zu konvertierenden Eingang haben. LabVIEW behandelt alle zusätzlichen Bezeichner, die Sie in diese Funktionen einfügen, als Buchstabenketten ohne besondere Bedeutung.

Zahlen und Strings

Numerische Daten und String-Daten unterscheiden sich voneinander, da es sich bei String-Daten um ASCII-Zeichen handelt, bei numerischen Daten jedoch nicht. Text- und Tabellenkalkulationsdateien akzeptieren nur Strings. Wenn Sie numerische Daten an eine Text- oder Tabellendatei schreiben möchten, müssen Sie die numerischen Daten zunächst in einen String umwandeln.

Wenn Sie einem vorhandenen String einen Satz Zahlen hinzufügen möchten, konvertieren Sie die numerischen Daten zunächst in einen String und verwenden dann die Funktion “Strings verknüpfen” oder eine andere String-Funktion, um den neuen String dem vorhandenen String hinzuzufügen. Die Konvertierungsfunktionen für Strings und Zahlen dienen zur Konvertierung numerischer Werte in Strings.

Ein String kann einen Zahlen enthalten, die in einem Graphen oder einem Diagramm dargestellt werden sollen. Beispielsweise können Sie eine Textdatei lesen, die einen Satz Zahlen enthält, die Sie in einem Diagramm zeichnen möchten. Allerdings weisen diese Zahlen das Format ASCII-Text auf, daher müssen Sie die Zahlen als String lesen und diesen String dann in Zahlen konvertieren, bevor eine Darstellung in einem Diagramm möglich ist.

Abbildung 10-2 zeigt einen String mit Zahlen. Sie sehen, wie der String in Zahlenwerte umgewandelt wird, ein Zahlen-Array erstellt wird und die Werte in ein Diagramm gezeichnet werden.

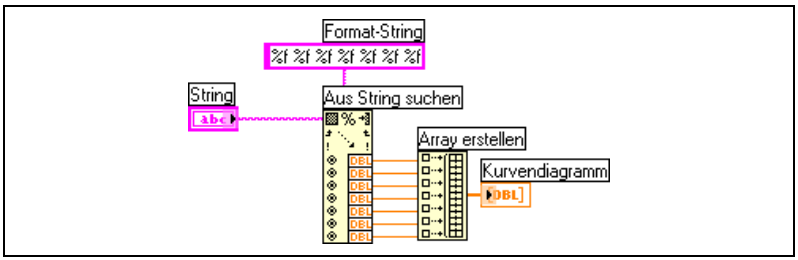


Abbildung 10-2. Konvertieren eines Strings in Zahlen

Konvertierung in das/aus dem XML-Format

XML (Extensible Markup Language) ist ein Formatierungsstandard, bei dem Daten mit Hilfe von Tags beschrieben werden. Im Gegensatz zu einem HTML-Tag enthält ein XML-Tag allerdings keine Formatierungsanweisungen, sondern dient nur zur Kennzeichnung der Daten.

Stellen Sie sich zum Beispiel vor, Sie sind Buchhändler und verkaufen Ihre Bücher im Internet. Sie möchten jedes Ihrer Bücher nach den folgenden Kriterien einteilen:

- Sparte (Belletristik oder Sachbuch)
- Titel
- Autor
- Herausgeber
- Preis
- Genre
- Kurzbeschreibung
- Seitenanzahl

Sie könnten demnach eine XML-Datei für jedes Buch erstellen. Die XML-Datei für ein Buch mit dem Titel *Deutschlands schönste Kirchen* könnte etwa so aussehen:

```
<Sachbuch>
<Titel>Deutschlands schönste Kirchen</Titel>
<Autor>Tony Walters</Autor>
<Herausgeber>Douglas Drive Publishing</Herausgeber>
<Preis>29,99 Euro<Preis>
<Genre>Reisen</Genre>
<Genre>Architektur</Genre>
```

```
<Genre>Geschichte</Genre>

<Kurzbeschreibung>In diesem Buch werden zwölf von
Deutschlands schönsten Kirchen mit Farbfotografien,
maßstabsgetreuen Übersichten und Informationen über ihre
Entstehungsgeschichte vorgestellt.</Kurzbeschreibung>

<Seiten>224</Seiten>

</Sachbuch>
```

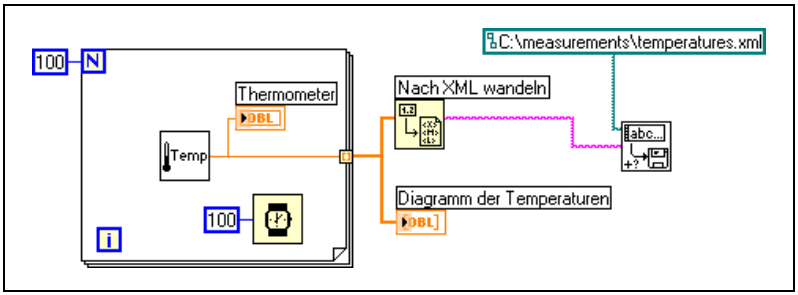
Daten können in LabVIEW ähnlich aufgeschlüsselt werden, und zwar jeweils nach Name, Wert und Typ. Ein String-Bedienelement für einen Benutzernamen kann in XML wie folgt dargestellt werden:

```
<String>

<Name>Benutzername</Name>

<Wert>Reggie Harmon</Wert>

</String>
```

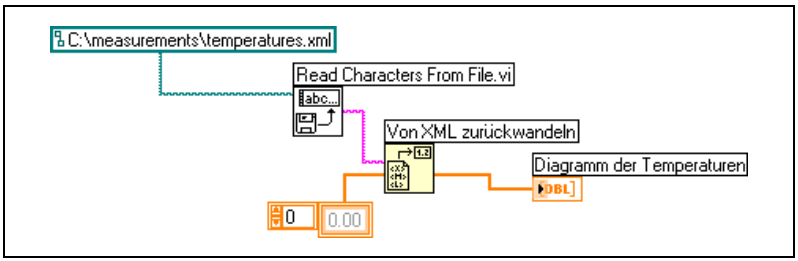
Verwendung von XML-basierten Datentypen

Beim Umwandeln von LabVIEW-Daten in XML-Daten werden diese so formatiert, dass Sie, falls Sie die Daten in einer Datei abspeichern, einfach zwischen Wert(en), Name(n) und dem Tag zur Beschreibung der Daten unterscheiden können. Wenn Sie zum Beispiel ein Array mit Temperaturmesswerten in XML umwandeln und diese in einer Datei speichern möchten, können Sie zur Ermittlung der Temperaturwerte einfach den `<Wert>`-Tag suchen, mit dem jeden einzelne Temperaturwert gekennzeichnet ist.

Um LabVIEW-Datentypen in das XML-Format zu konvertieren, verwenden Sie die Funktion “In XML konvertieren”. Im folgenden Beispiel-VI wird die Erzeugung von 100 Temperaturwerten simuliert, die als Array ausgegeben und in einem Diagramm dargestellt werden. Nach Konvertierung des Arrays in das XML-Format werden die Daten in die Datei `temperatures.xml` geschrieben.



Mit der Funktion “Aus XML zurückkonvertieren” können XML-Datentypen in LabVIEW-Datentypen umgewandelt werden. Im folgenden Beispiel werden die 100 Temperaturwerte aus der Datei `temperatures.xml` ausgelesen, und die Array-Werte werden in einem Diagramm dargestellt.



Hinweis Obwohl es möglich ist, LabVIEW-Variant-Daten in XML zu wandeln, würde der Versuch, XML-Daten in Variant-Daten zurück zu konvertieren zu einem leeren LabVIEW-Variant führen.

In der Bibliothek `examples\file\XMLex.llb` finden Sie Beispiele für die Konvertierung aus dem und in das XML-Format.

LabVIEW-XML-Schema

Daten werden in LabVIEW nach einem festgelegten XML-Schema konvertiert. Im Moment ist es allerdings noch nicht möglich, benutzerdefinierte Schemen zu erstellen oder zu steuern, wie LabVIEW die einzelnen

Daten mit den Tags verbindet. Es können auch keine vollständigen VIs oder Funktionen in XML umgewandelt werden.

Das in LabVIEW verwendete XML-Schema finden Sie im `LabVIEW\help`-Verzeichnis.

Gruppieren von Daten mit Arrays und Clustern

Die Array- und Cluster-Bedienelemente und -Funktionen dienen zur Gruppierung von Daten. Mit Arrays werden Datenelemente gleichen Typs und mit Clustern Datenelemente unterschiedlichen Typs zusammengefasst.

Arrays

Ein Array besteht aus Elementen und Dimensionen. Die Elemente sind die Daten, aus denen sich das Array zusammensetzt. Eine Dimension ist die Länge, Höhe oder Tiefe eines Arrays. Ein Array kann eine oder mehrere Dimensionen und je nach verfügbarem Speicher bis zu $2^{31} - 1$ Elemente pro Dimension enthalten.

Arrays können aus numerischen, booleschen, Pfad-, String-, Signalverlaufs- und Cluster-Daten erstellt werden. Der Einsatz von Arrays bietet sich insbesondere an, wenn ähnliche Daten verwendet oder Berechnungen wiederholt werden sollen. Arrays eignen sich ideal für das Speichern von Signalverlaufsdaten oder zum Speichern von Daten, die in Schleifen erzeugt werden. In diesem Fall wird bei jedem Schleifendurchlauf ein neues Array-Element erzeugt.

Es ist nicht möglich, Arrays bestehenden Arrays zu erzeugen. Sie können jedoch ein mehrdimensionales Array verwenden oder ein Array aus Clustern erzeugen, von denen jeder ein oder mehrere Arrays enthält. Weitere Informationen zu den Elementtypen, die ein Array enthalten kann, finden Sie in Abschnitt *Beschränkungen für Arrays* dieses Kapitels. Nähere Hinweise zu Clustern entnehmen Sie bitte dem Abschnitt *Cluster* dieses Kapitels.

Indizes

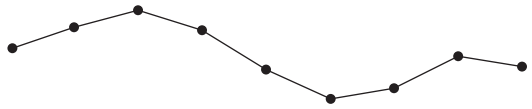
Um ein bestimmtes Element in einem Array zu finden, muss ein Index pro Dimension vorhanden sein. In LabVIEW können Sie mit Hilfe von Indizes Elemente, Zeilen, Spalten und Seiten aus einen Array im Blockdiagramm abrufen.

Beispiele für Arrays

Ein Beispiel für ein einfaches Array ist ein Text-Array, in dem die neun Planeten unseres Sonnensystems aufgeführt sind. Dazu wird ein aus Strings bestehendes 1D-Array mit neun Elementen verwendet.

Array-Elemente unterliegen einer Ordnung. Bei einem Array wird ein Index verwendet, so dass Sie auf jedes Element direkt zugreifen können. Der Index liegt im Bereich von 0 bis $n - 1$, wobei n für die Anzahl der Elemente im Array steht. Für die neun Planeten unseres Sonnensystems zum Beispiel wäre $n = 9$. Der Index umfasst also einen Bereich von 0 bis 8. Die Erde ist der dritte Planet und hätte somit den Index 2.

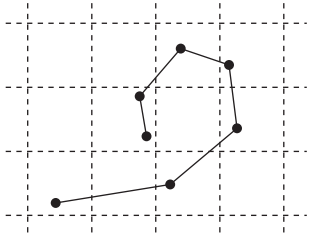
Ein weiteres Beispiel für ein Array sind die Abtastwerte eines Signalverlaufs, die in einem numerischen Array abgelegt sind – jede Zelle enthält einen Abtastwert bezogen auf jeweils einen Abtasttzeitpunkt, wie in Abbildung 10-3 dargestellt.



Index	0	1	2	3	4	5	6	7	8
Spannung (Volt)	0.4	0.9	1.4	0.8	–0.1	–0.7	–0.3	0.3	0.2

Abbildung 10-3. Signalverlauf als ein aus Zahlen bestehendes Array

Ein etwas komplexeres Beispiel für ein Array ist ein Graph, der als ein Array aus Punkten dargestellt wird. Jeder Punkt darin ist ein Cluster aus den x- und y-Koordinaten (siehe Abbildung 10-4).



Index	0	1	2	3	4	5	6
X–Koordinaten	0.4	2.2	3.3	3.2	2.4	1.8	1.9
Y–Koordinaten	0.2	0.5	1.3	2.3	2.6	1.9	1.2

Abbildung 10-4. Graph als ein aus Punkten bestehendes Array

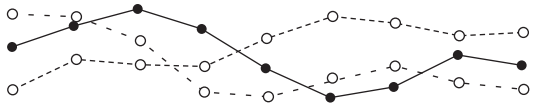
In den vorstehenden Beispielen wird ein 1D-Array verwendet. Bei einem 2D-Array werden die Elemente in einem Raster gespeichert. Dies setzt einen Spalten- und einen Zeilenindex zum Suchen eines Elements voraus, die beide jeweils nullbasiert sind. In Abbildung 10-5 ist ein aus 6 Spalten und 4 Zeilen bestehendes 2D-Array dargestellt, das $6 \times 4 = 24$ Elemente enthält.

		Spalten index					
		0	1	2	3	4	5
Reihenindex	0						
	1						
	2						
	3						

Abbildung 10-5. 2D-Array mit 6 Spalten und 4 Zeilen

Beispielsweise enthält ein Schachbrett acht Spalten und acht Zeilen für insgesamt 64 Positionen. Jede Position kann leer sein oder eine Schachfigur enthalten. Ein Schachbrett können Sie also als ein aus Strings bestehendes 2D-Array darstellen. Hierbei ist jeder String der Name der Figur, die die entsprechende Position auf dem Schachbrett einnimmt, oder ein leerer String, wenn die Position leer ist.

Sie können die Beispiele für 1D-Arrays in den Abbildungen 10-3 und 10-4 zweidimensional machen, indem Sie dem Array eine Zeile hinzufügen. Abbildung 10-6 zeigt drei Signalverläufe, die in Form eines 2D-Arrays aus Zahlen dargestellt sind. Mit dem Zeilenindex wird der Signalverlauf gewählt, mit dem Spaltenindex ein Punkt im Signalverlauf.



	0	1	2	3	4	5	6	7	8
0	0.4	0.9	1.4	0.8	–0.1	–0.7	–0.3	0.3	0.2
1	–0.1	0.6	0.4	0.2	0.8	1.6	1.4	0.9	1.1
2	1.6	1.4	0.7	0.5	–0.5	–0.6	–0.2	0.3	0.5

Abbildung 10-6. Mehrere Signalverläufe in einem aus Zahlen bestehenden 2D-Array

Weitere Beispiele für Arrays finden Sie in `examples\general\arrays.llb`. Für weitere Hinweise zu Arrays lesen Sie bitte im Abschnitt *Erstellen von Arrays mit Hilfe von Schleifen* des Kapitels 8, *Schleifen und Strukturen* nach.

Beschränkungen für Arrays

Sie können aus beinahe jedem Datentyp ein Array erstellen, wobei die folgenden Ausnahmen gelten:

- Es können keine aus Arrays bestehenden Arrays erzeugt werden. Sie können allerdings ein mehrdimensionales Array erstellen oder mit Hilfe der Funktion “Cluster-Array erstellen” ein Array aus Clustern erzeugen, in dem jeder Cluster ein oder mehrere Arrays enthält.
- Es ist nicht möglich, Arrays aus Unterpanel-Bedienelementen zu erzeugen.
- Es ist nicht möglich, Arrays aus Registerbedienelementen zu erzeugen.
- Es ist nicht möglich, Arrays aus ActiveX-Bedienelementen zu erzeugen.
- Es ist nicht möglich, Arrays aus Diagrammen zu erzeugen.
- Es ist nicht möglich, Arrays aus XY-Graphen mit mehreren Kurven zu erzeugen.

Erstellen von Array-Bedien- und Anzeigeelementen und -Konstanten

Zur Erzeugung eines Array-Elements platzieren Sie einen Array-Container (wie in Abbildung 10-7 dargestellt) in das Frontpanel. Ziehen Sie anschließend ein Datenobjekt oder Element hinein. Dabei kann es sich um ein digitales, boolesches Element oder ein String-, Pfad-, RefNum- oder Cluster-Element handeln.

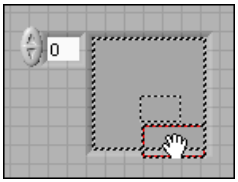


Abbildung 10-7. Array-Container

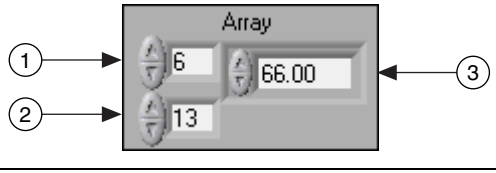
Array-Container passen sich automatisch an die Größe des neuen Objekts an. Um eine neue Dimension hinzuzufügen, klicken Sie mit der rechten Maustaste auf die Indexanzeige und wählen Sie im Kontextmenü die Option **Dimension hinzufügen** aus. Es ist aber auch möglich, die Indexanzeige auf die gewünschte Dimensionenanzahl zu erweitern. Zum Entfernen von Dimensionen klicken Sie mit der rechten Maustaste auf die Indexanzeige und wählen Sie im Kontextmenü den Befehl **Dimension entfernen** aus oder verkleinern Sie die Indexanzeige.

Um ein bestimmtes Element auf dem Frontpanel anzuzeigen, geben Sie entweder die Indexnummer in die Indexanzeige ein oder verwenden Sie die Pfeile der Indexanzeige.

Zum Erstellen einer Array-Konstante auf dem Blockdiagramm wählen Sie eine solche aus der Palette **Funktionen** aus, platzieren Sie den Array-Container auf das Frontpanel und ziehen Sie eine String-, Cluster- oder eine numerische Konstante hinein. Sie können eine Array-Konstante als Basis für den Vergleich mit einem anderen Array verwenden.

Array-Indexanzeige

Ein 2D-Array enthält Zeilen und Spalten. In der Abbildung 10-8 ist der Zeilenindex (die obere Anzeige mit den zwei Feldern links) und der Spaltenindex (die untere Anzeige) zu sehen. Die kombinierte Anzeige rechts daneben zeigt den Wert an der angegebenen Position an. In Abbildung 10-8 sehen Sie, dass der Wert an Zeile 6 und Spalte 13 **66** lautet.



1	Zeilenindex	2	Spaltenindex	3	Wert an Zeile/Spalte

Abbildung 10-8. Array-Bedienelement

Zeilen und Spalten sind nullbasiert, was bedeutet, dass die erste Spalte 0, die zweite Spalte 1 und so weiter ist. Wird die Indexanzeige des folgenden Arrays auf Zeile 1, Spalte 2 geändert, wird der Wert 6 angezeigt.

0	1	2	3
4	5	6	7
8	9	10	11

Wenn Sie versuchen, eine Spalte oder eine Zeile anzuzeigen, die außerhalb des Bereichs der Array-Dimensionen liegt, wird das Array-Bedienelement abgeblendet, um anzuzeigen, dass kein Wert definiert ist, und es wird der Standardwert des jeweiligen Datentyps angezeigt. Der Standardwert des Datentyps hängt vom Datentyp des Arrays ab.

Zur Anzeige mehrerer Zeilen oder Spalten verwenden Sie das Positionierwerkzeug.

Array-Funktionen

Mit den Array-Funktionen können Arrays erstellt und verändert werden, zum Beispiel durch folgende Operationen:

- Extrahieren von einzelnen Datenelementen aus einem Array.
- Einfügen, Löschen oder Ersetzen von Datenelementen in einem Array.
- Teilen von Arrays.

Automatisches Ändern der Größe von Array-Funktionen

Die Funktionen “Array indizieren”, “Teilarray ersetzen”, “In Array einfügen”, “Aus Array entfernen” und “Teilarray” passen sich automatisch an die Dimensionenanzahl des angeschlossenen Eingabe-Arrays an. Wenn Sie beispielsweise ein 1D-Array mit einer dieser Funktionen verbinden, hat die Funktion einen einzelnen Indexeingang. Wenn Sie ein 2D-Array mit der gleichen Funktion verbinden, werden zwei Indexeingänge angezeigt: einer für die Zeile und einer für die Spalte.

Sie können mit diesen Funktionen auf mehr als ein Element oder Teilarray (Zeile, Spalte oder Seite) zugreifen, indem Sie die Größe der Funktion mit Hilfe des Positionierwerkzeugs manuell ändern. Wenn Sie eine dieser Funktionen erweitern, werden die Funktionen in Inkrementen erweitert, die von den Dimensionen des Arrays bestimmt werden, das mit der Funktion verbunden wird. Wenn Sie ein 1D-Array mit einer dieser Funktionen verbinden, wird die Funktion um einen Indexeingang erweitert. Wenn Sie mit

der gleichen Funktion ein 2D-Array verbinden, wird die Funktion um zwei Indexeingänge erweitert, und zwar einen für die Zeile und einen für die Spalte.

Die Indexeingänge, die Sie verbinden, bestimmen die Form des Teilarrays, auf das Sie zugreifen oder das Sie ändern möchten. Wenn der Eingang einer Funktion des Typs “Array indizieren” ein 2D-Array ist und Sie nur den Eingang **Zeile** verbinden, wird eine vollständige 1D-Zeile des Arrays entnommen. Wenn Sie nur den Eingang **Spalte** verbinden, wird eine vollständige 1D-Spalte des Arrays extrahiert. Wenn Sie den Eingang **Zeile** und den Eingang **Spalte** verbinden, extrahieren Sie ein einzelnes Element des Arrays. Jede Eingangsgruppe ist unabhängig und kann auf einen beliebigen Teil jeder Dimension des Arrays zugreifen.

Im Blockdiagramm in Abbildung 10-9 werden mit Hilfe der Funktion “Array indizieren” eine Zeile und ein Element aus einem 2D-Array entnommen.

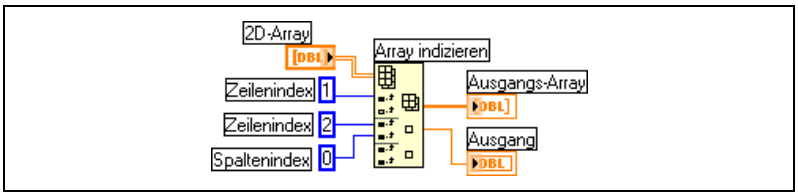


Abbildung 10-9. Indizieren eines 2D-Arrays

Um auf mehrere aufeinander folgende Werte in einem Array zuzugreifen, erweitern Sie die Funktion “Array indizieren” um drei Felder, verbinden jedoch mit den Index-Eingängen keine Werte. Um beispielsweise die erste, zweite und dritte Zeile eines 2D-Arrays zu extrahieren, erweitern Sie die Funktion um drei Felder und verbinden mit jedem **Teilarray**-Ausgang 1D-Array-Anzeigeelemente.

Weitere Einzelheiten dazu, wie durch den Einsatz von Arrays eine höhere Speichereffizienz erzielt werden kann, finden Sie auch in den *LabVIEW Development Guidelines* im Kapitel 6, *LabVIEW Style Guide*, unter *Memory and Speed Optimization*.

Cluster

In Clustern werden wie bei einem Kabelbündel Datenelemente unterschiedlichen Typs gruppiert, wobei jedes Kabel für ein unterschiedliches Element des Clusters steht. Ein Cluster entspricht einem Datensatz oder einer Struktur in textbasierten Programmiersprachen.

Durch das Bündeln mehrerer Datenelemente zu Clustern werden unübersichtliche Verbindungen im Blockdiagramm vermieden und es verringert sich die Anzahl der Anschlussfeldanschlüsse, die für SubVIs benötigt werden. Ein Anschlussfeld kann über maximal 28 Anschlüsse verfügen. Wenn das Frontpanel mehr als 28 Bedien- und Anzeigeelemente enthält, die programmgesteuert verwendet werden sollen, fassen Sie einige davon als Cluster zusammen und weisen diesem Cluster einen Anschluss des Anschlussfeldes zu.

Obwohl Cluster- und Array-Elemente jeweils geordnet sind, werden die Elemente immer gleichzeitig und nicht Index für Index aufgeschlüsselt. Um auf bestimmte Cluster-Elemente zuzugreifen, kann auch die Funktion “Nach Namen aufschlüsseln” verwendet werden. Cluster unterscheiden sich auch insofern von Arrays, als dass sie eine feste Größe aufweisen. Wie ein Array kann auch ein Cluster als Bedien- oder Anzeigeelement verwendet werden. Ein Cluster kann jedoch immer nur einen Elementetyp enthalten.

Bei den meisten Clustern im Blockdiagramm sind Verbindungsmuster und Symbol rosa dargestellt . Cluster aus Zahlenwerten (manchmal auch als Punkte bezeichnet) werden durch ein braunes Verbindungsmuster und Symbol gekennzeichnet. Dieser Clustertyp kann mit numerischen Funktionen wie beispielsweise “Addieren” oder “Quadratwurzel” verbunden werden, um dieselbe Rechenoperation gleichzeitig an allen Elementen durchzuführen.

Cluster-Elemente haben eine logische Reihenfolge, die in keiner Beziehung zu ihrer Position innerhalb des Containers steht. Das erste Objekt, das Sie in den Cluster einfügen, ist Element 0, das zweite ist Element 1 und so weiter. Wenn Sie ein Element löschen, wird die Reihenfolge automatisch angepasst. Durch die Cluster-Reihenfolge wird auch bestimmt, in welcher Reihenfolge die Elemente als Anschlüsse in den Funktionen “Elemente bündeln” und “Aufschlüsseln” im Blockdiagramm angezeigt werden. Zur Anzeige und zum Bearbeiten der Reihenfolge der Cluster-Elemente, klikken Sie mit der rechten Maustaste auf den Rand des Clusters und wählen aus dem Kontextmenü die Option **Neuanordnung der Bedienelemente in Cluster** aus.

Damit zwei Cluster miteinander verbunden werden können, müssen sie die gleiche Anzahl an Elementen enthalten. Außerdem müssen die Datentypen der einander entsprechenden Elemente (festgelegt durch die Reihenfolge im Cluster) zueinander kompatibel sein. Wenn beispielsweise zwei Cluster miteinander verbunden sind, jedoch das Element 1 in einem Cluster eine Fließkommazahl mit doppelter Genauigkeit und in dem anderen Cluster ein

String-Element ist, erscheint die Verbindung im Blockdiagramm als unterbrochen und das VI kann nicht ausgeführt werden. Bei Zahlenwerten mit unterschiedlicher Kommastellenanzahl wird von LabVIEW automatisch ein Wert umgewandelt, so dass die Anzahl übereinstimmt. Weitere Informationen zur Umwandlung von Zahlen finden Sie in Abschnitt *Numerische Konvertierungen* des Anhangs B, *Polymorphe Funktionen*.

Mit den Cluster-Funktionen können Sie Cluster erstellen und bearbeiten. So sind damit beispielsweise folgende Operationen möglich:

- Extrahieren von einzelnen Datenelementen aus einem Cluster.
- Hinzufügen von einzelnen Datenelementen zu einem Cluster.
- Aufteilen eines Clusters in die einzelnen Datenelemente.

11

Lokale und globale Variablen

In LabVIEW werden Daten über den Blockdiagrammanschluss eines Frontpanel-Elements vom Frontpanel in das Blockdiagramm geleitet und umgekehrt. Ein Frontpanel-Objekt verfügt jedoch immer nur über einen Blockdiagrammanschluss, und in Ihrer Applikation müssen Sie möglicherweise von mehr als einer Position aus auf die Daten dieses Anschlusses zugreifen können.

Mit lokalen und globalen Variablen werden Daten zwischen verschiedenen Teilen (Haupt-VIs, SubVIs) einer Applikation übertragen, bei denen keine direkte Verbindung herstellbar ist. Über lokale Variablen kann von mehreren Stellen eines VIs aus auf Frontpanel-Objekte zugegriffen werden. Globale Variablen dienen zum Datenaustausch zwischen VIs.

Weitere Informationen ...

Weitere Informationen zum Einsatz von lokalen und globalen Variablen finden Sie in der *LabVIEW-Hilfe*.

Lokale Variablen

Über lokale Variablen kann von mehreren Stellen eines VIs auf Frontpanel-Objekte zugegriffen werden, und es können Daten zwischen Strukturen des Blockdiagramms ausgetauscht werden, zwischen denen keine direkte Verbindung hergestellt werden kann.

Mit einer lokalen Variablen können Sie an ein Bedien- oder Anzeigeelement auf dem Frontpanel schreiben oder dieses auslesen. Das Schreiben in eine lokale Variable ist mit der Übergabe von Daten an einen beliebigen anderen Anschluss vergleichbar. Mit einer lokalen Variablen können Sie jedoch auch dann in den Anschluss schreiben, wenn es sich um ein Bedienelement handelt, und den Anschluss auslesen, wenn es sich um ein Anzeigeelement handelt. Frontpanel-Objekt können also sowohl als Eingang als auch als Ausgang verwendet werden.

Wenn sich die Benutzer einer Benutzeroberfläche beispielsweise anmelden müssen, können die Eingabefelder **`Anmeldename`** und **`Passwort`** mit Hilfe einer lokalen Variable ausgelesen werden, und sie können gelöscht werden, bevor sich ein neuer Benutzer anmeldet, indem jeweils ein leerer String hineingeschrieben wird.

Erstellen von lokalen Variablen

Um eine lokale Variable zu erzeugen, klicken Sie mit der rechten Maustaste auf ein Frontpanel-Objekt oder einen Blockdiagrammanschluss und wählen **Erstelle»Lokale Variable**. Im Blockdiagramm erscheint dann eine lokale Variable für das entsprechende Objekt.

Es ist aber auch möglich, eine lokale Variable aus der Palette **Funktionen** in das Blockdiagramm einzufügen. Der lokale Variablen-Knoten (siehe Abbildung links) ist dann allerdings noch nicht mit einem Bedien- oder Anzeigeelement verknüpft. Führen Sie einen Rechtsklick auf dem Knoten aus und wählen Sie aus dem Kontextmenü unter **Objekt wählen** das Frontpanel-Objekt, dem die Variable zugeordnet werden soll. Im Kontextmenü werden nur die Frontpanel-Objekte mit objektspezifischen Beschriftungen aufgeführt.

Die objektspezifischen Beschriftungen, mit denen lokale Variablen mit Frontpanel-Objekten verknüpft werden, sollten daher immer aussagekräftig sein. Weitere Informationen zu objektspezifischen und freien Beschriftungen finden Sie im Abschnitt *Beschriftungen* des Kapitels 4, *Erstellen des Frontpanels*.

Globale Variablen

Globale Variablen dienen zum Datenaustausch zwischen mehreren VIs. Diese müssen dazu gleichzeitig ausgeführt werden. Globale Variablen sind integrierte LabVIEW-Objekte. Bei der Erstellung einer globalen Variable erzeugt LabVIEW automatisch ein spezielles globales VI. Dieses enthält jedoch nur ein Frontpanel und kein Blockdiagramm. Um die Datentypen der globalen Variablen zu definieren, die es enthält, fügen Sie dem Frontpanel des globalen VIs Bedien- und Anzeigeelemente hinzu. Das Frontpanel ist also als Container zu verstehen, über das mehrere VIs auf Daten zugreifen können.

Nehmen wir beispielsweise an, Sie verfügen über zwei VIs, die gleichzeitig ausgeführt werden. Jedes VI enthält eine While-Schleife und gibt Daten an ein Signalverlaufsdiagramm weiter. Das erste VI enthält ein boolesches Bedienelement. Um beide Schleifen mit diesem Bedienelement beenden zu

können, sollte eine globale Variable verwendet werden. Wenn sich beide Schleifen in ein und demselben Blockdiagramm befänden, könnte dazu auch eine lokale Variable verwendet werden.

Erstellen von globalen Variablen

Wählen Sie eine globale Variable (siehe links) aus der Palette **Funktionen** und fügen Sie sie in das Blockdiagramm ein. Klicken Sie sie anschließend doppelt an, um das Frontpanel anzuzeigen. Platzieren Sie nun Bedien- und Anzeigeelemente in das Frontpanel.

Da globale Variablen mit den objektspezifischen Beschriftungen der betreffenden Bedien- und Anzeigeelemente gekennzeichnet werden, sollten diese Beschriftungen immer möglichst aussagekräftig sein. Weitere Informationen zu verknüpften und freien Beschriftungen finden Sie in Abschnitt *Beschriftungen* des Kapitels 4, *Erstellen des Frontpanels*.

Sie können entweder mehrere einzelne globale VIs mit jeweils einem Frontpanel-Objekt oder ein globales VI mit mehreren Frontpanel-Objekten erstellen. Ein globales VI mit mehreren Objekten arbeitet ökonomischer, da Sie zusammenhängende Variablen gruppieren können. Im Blockdiagramm eines VIs können mehrere globale Variablenknoten enthalten sein, die mit Bedien- und Anzeigeelementen eines globalen VIs verbunden sind. Dabei kann es sich entweder um Kopien der ersten globalen Variable handeln, die in das Blockdiagramm platziert wurde, oder um globale Variablen in SubVIs. Ein VI mit einer globalen Variable kann wie ein normales SubVI in andere VIs eingefügt werden. Jedesmal, wenn Sie einen neuen globalen Variablenknoten in ein Blockdiagramm einfügen, wird ein neues VI erstellt, das nur dieser Variablen und Kopien davon zugeordnet ist. Weitere Informationen zu SubVIs finden Sie in dem Abschnitt *SubVIs* des Kapitels 7, *Erstellen von VIs und SubVIs*.

Nachdem Sie mit dem Platzieren von Objekten auf dem Frontpanel des globalen VIs fertig sind, speichern Sie es, und kehren Sie zum Blockdiagramm des ursprünglichen VIs zurück. Anschließend ist im globalen VI ein Objekt auszuwählen, auf das zugegriffen werden soll. Klicken Sie dazu mit der rechten Maustaste auf die globale Variable und wählen Sie aus dem Kontextmenü unter **Objekt wählen** ein Frontpanel-Objekt aus. Im Kontextmenü werden nur die Frontpanel-Objekte mit objektspezifischen Beschriftungen aufgeführt.

Lesen und Schreiben von Variablen

Nachdem Sie eine lokale oder globale Variable erstellt haben, kann sie als Datenquelle oder -senke verwendet werden. Per Voreinstellung ist eine neue Variable immer als Datensenke konfiguriert. Solche lokale oder globale Variablen mit Schreibzugriff fungieren also wie ein Anzeigeelement. Wenn Sie neue Daten an die lokale oder globale Variable übergeben werden, wird das damit verbundene Bedien- oder Anzeigeelement jeweils aktualisiert.

Eine Variable kann auch so konfiguriert werden, dass sie als Datenquelle bzw. als lokale oder globale Variable mit Lesezugriff fungiert. Klicken Sie dazu mit der rechten Maustaste auf die Variable, und wählen Sie aus dem Kontextmenü **In Lesen ändern**, um die Variable als Bedienelement zu konfigurieren. Wenn der Knoten ausgeführt wird, wird das entsprechende Bedien- oder Anzeigeelement des Frontpanels ausgelesen.

Um die Variable als Datensenke zu konfigurieren, klicken Sie sie mit der rechten Maustaste an und wählen im Kontextmenü die Option **In Lesen ändern**.

Im Blockdiagramm unterscheiden sich lokale oder globale Lesevariablen von lokalen oder globalen Schreibvariablen genau wie Bedien- und Anzeigeelemente. Lesevariablen sind ähnlich wie Bedienelemente durch einen dicken Rand gekennzeichnet, während Schreibvariablen wie Anzeigeelemente einen dünnen Rand haben.

Beispiele für den Einsatz von lokalen und globalen Variablen finden Sie in `examples\general\locals.llb` und `example\general\globals.llb`.

Umsichtiger Einsatz von lokalen und globalen Variablen

Lokale und globale Variablen sind Komponenten der fortgeschrittenen VI-Entwicklung, da sie nicht in das datenflussorientierte Ausführungsmodell von LabVIEW passen. Da Blockdiagramme mit Variablen unter Umständen schwieriger nachvollziehbar sind, sollten Variablen möglichst sorgfältig eingesetzt werden. Wenn lokale oder globale Variablen falsch eingesetzt werden, zum Beispiel anstelle von SubVI-Anschlüssen oder zum Datenaustausch zwischen den einzelnen Rahmen von Sequenzstrukturen, kann das zu einem unerwarteten Verhalten des VIs führen. Bei der Verwendung zu vieler Variablen, beispielsweise um lange Verbindungen auf dem Blockdiagramm zu ersetzen, kann sich die Ausführungsgeschwin-

digkeit des VIs verlangsamen. Weitere Informationen zum datenflussorientieren Ausführungsmodell von LabVIEW finden Sie im Abschnitt *Datenflussprinzip Blockdiagramm* des Kapitels 5, *Erstellen des Blockdiagramms*.

Initialisieren von lokalen und globalen Variablen

Vergewissern Sie sich vor der Ausführung eines VIs, dass in den lokalen und globalen Variablen bekannte Daten enthalten sind, da das VI sonst unter Umständen nicht korrekt funktioniert.

Wenn Sie die Variable vor dem ersten Auslesen nicht initialisieren, enthält sie den Standardwert des verbundenen Frontpanel-Objekts.

Race Conditions

Eine Race Condition (Konkurrenzsituation) tritt immer dann auf, wenn zwei oder mehrere parallel ausgeführte Codeabschnitte versuchen, den Wert derselben Ressource (in der Regel einer globalen oder lokalen Variable) zu verändern. In Abbildung 11-1 finden Sie ein Beispiel für eine Race Condition.

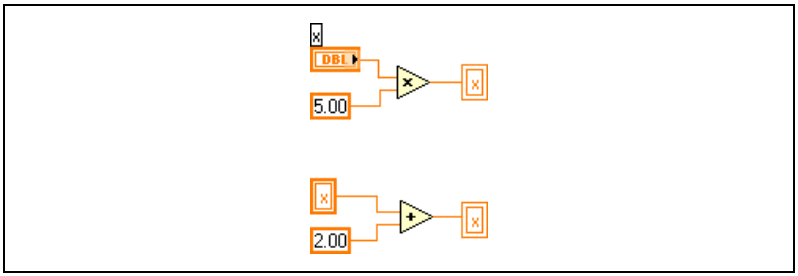


Abbildung 11-1. Race Condition

Die Ausgabe dieses VIs richtet sich nach der Ausführungsreihenfolge der Operationen. Da im Beispiel zwischen den Operationen keine Datenabhängigkeit besteht, kann nicht bestimmt werden, welche zuerst ausgeführt wird. Zur Vermeidung solcher Konkurrenzsituationen sollten keine Daten in eine Variable geschrieben werden, aus der auch Daten ausgelesen werden. Weitere Informationen zur Datenabhängigkeit finden Sie in Abschnitt *Datenabhängigkeit und künstliche Datenabhängigkeit* des Kapitels 5, *Erstellen des Blockdiagramms*.

Lokale Variablen und Speichernutzung

Lokale Variablen erstellen Kopien der Datenpuffer. Das heißt, beim Auslesen einer lokalen Variable wird für die Daten des damit verbundenen Bedienelements ein neuer Puffer erstellt.

Wenn große Datenmengen von einer Stelle im Blockdiagramm an eine andere übertragen werden sollen, ist im Allgemeinen mehr Speicherplatz erforderlich, und dementsprechend verlangsamt sich auch die Ausführungsgeschwindigkeit gegenüber der Datenübergabe über eine normale Verbindung. Zur Speicherung von Daten während der VI-Ausführung empfiehlt es sich, Schieberegister zu verwenden.

Globale Variablen und Speichernutzung

Beim Auslesen globaler Variablen wird eine Kopie der darin gespeicherten Daten erstellt.

Wenn große Arrays und Strings bearbeitet werden, kann die Zeit und die Speicherkapazität, die für das Bearbeiten von globalen Variablen benötigt wird, beträchtlich sein. Besonders die Bearbeitung von Arrays ist zeit- und speicherintensiv, da LabVIEW immer das gesamte Array speichert, auch wenn Sie nur ein einziges Element verändern. Beim Auslesen einer globalen Variable an mehreren Stellen einer Applikation werden mehrere Speicherpuffer erzeugt, was ineffizient ist und die Ausführungsgeschwindigkeit verringert.

12

Graphen und Diagramme

Mit Graphen und Diagrammen können Daten in grafischer Form dargestellt werden.

Graphen und Diagramme unterscheiden sich hinsichtlich der Anzeige und der Aktualisierung der Daten. VIs mit Graphen erfassen die Daten normalerweise in einem Array und stellen die Daten anschließend in einem Graphen dar, ähnlich wie in einer Tabellenkalkulation, bei der die Daten zuerst gespeichert und dann grafisch dargestellt werden. Bei einem Diagramm werden dagegen zu den bereits in der Anzeige vorhandenen Datenpunkten neue hinzugefügt, so dass eine Daten-Historie erzeugt wird. Deshalb ist es möglich, in Diagrammen die aktuellen Messwerte beziehungsweise die aktuelle Messung in Zusammenhang mit den zuvor erfassten Werten anzuzeigen.

Weitere Informationen ...

Weitere Informationen über die Verwendung von Graphen und Diagrammen finden Sie in der *LabVIEW-Hilfe*.

Verschiedene Typen von Graphen und Diagrammen

Es gibt folgende Arten von Graphen und Diagrammen:

- **Kurvendiagramm und Kurvengraph**—Zeigen die erfassten Daten mit konstanter Rate an.
- **XY-Graph**—Zeigt die erfassten Daten mit nicht konstanter Rate an, wie zum Beispiel Daten, die getriggert erfasst werden.
- **Intensitätsdiagramm und Intensitätsgraph**—Zeigen dreidimensionale Daten in einem zweidimensionalen Plot an, wobei die Werte der dritten Dimension anhand von Farben angezeigt werden.
- **Digitaler Kurvengraph**—Zeigt Daten als Impulse an oder als Gruppen digitaler Linien. Die Übertragung von Daten zwischen Computern erfolgt in Impulsform.

- **(Windows) 3D-Graphen**—Zeigen 3D-Daten anhand eines ActiveX-Objekts auf dem Frontpanel in dreidimensionaler Darstellung an.

Beispiele für Graphen und Diagramme finden Sie im Verzeichnis `examples\general\graphs`.

Optionen für Graphen und Diagramme

Obwohl die Darstellung von Daten in Graphen und Diagrammen unterschiedlich erfolgt, verfügen diese über mehrere gemeinsame Optionen, auf die Sie über das Kontextmenü zugreifen können. Weitere Informationen zu den nur für Graphen beziehungsweise Diagrammen verfügbaren Optionen finden Sie in den Abschnitten *Anpassen von Graphen* und *Diagramme benutzerspezifisch anpassen* dieses Kapitels.

Die Optionen von Signalverlaufs- und XY-Graphen bzw. -Diagrammen unterscheiden sich von den Optionen der digitalen, Intensitäts- und 3D-Graphen bzw.-Diagrammen. Weitere Informationen zu den Optionen für digitale, Intensitäts- und 3D-Graphen/Diagramme finden Sie in den Abschnitten *Intensitätsgraphen und -diagramme*, *3D-Graphen* und *Digitale Kurvengraphen* dieses Kapitels.

Verwendung mehrerer x- und y-Achsen in Graphen und Diagrammen

Graphen sind mit mehreren x- und y-Achsen darstellbar und Diagramme mit mehreren x-Achsen. Auf diese Weise können mehrere Kurven mit verschiedenen x- und y-Achsen gleichzeitig dargestellt werden. Klicken Sie mit der rechten Maustaste auf die x- oder y-Achse des Kurvengraphen oder -diagramms, und wählen Sie **Maßstab duplizieren** aus dem Kontextmenü, um mehrere Achsen zum Graphen oder Diagramm hinzuzufügen.

Kantengeglättete Kurven für Graphen und Diagramme

Mit Hilfe der Kantenglättung kann die Darstellung von Linien in Diagrammen und Graphen verbessert werden. Wenn diese Option aktiviert ist, wird der Rastereffekt ausgeschaltet, und die Kurven erscheinen glatter. Merkmale wie Linienstärke, Linienstil und Punktstil werden dadurch nicht beeinflusst.

Hinweis Bei digitalen Kurvengraphen ist keine Kantenglättung möglich.

Zum Aktivieren von kantengeglätteten Liniendarstellungen klicken Sie mit der rechten Maustaste auf die Diagrammlegende und wählen aus dem Kon-

textmenü die Option **Kantengeglättet**. Wenn die Legende nicht angezeigt wird, klicken Sie mit der rechten Maustaste auf den Graphen bzw. das Diagramm und wählen aus dem Kontextmenü **Sichtbare Objekte»Legende der Kurve**.

Hinweis Da die Darstellung von kantengeglätteten Linien sehr rechenintensiv sein kann, verringert sich dadurch unter Umständen die Systemleistung.

Anpassen der Darstellung von Graphen und Diagrammen

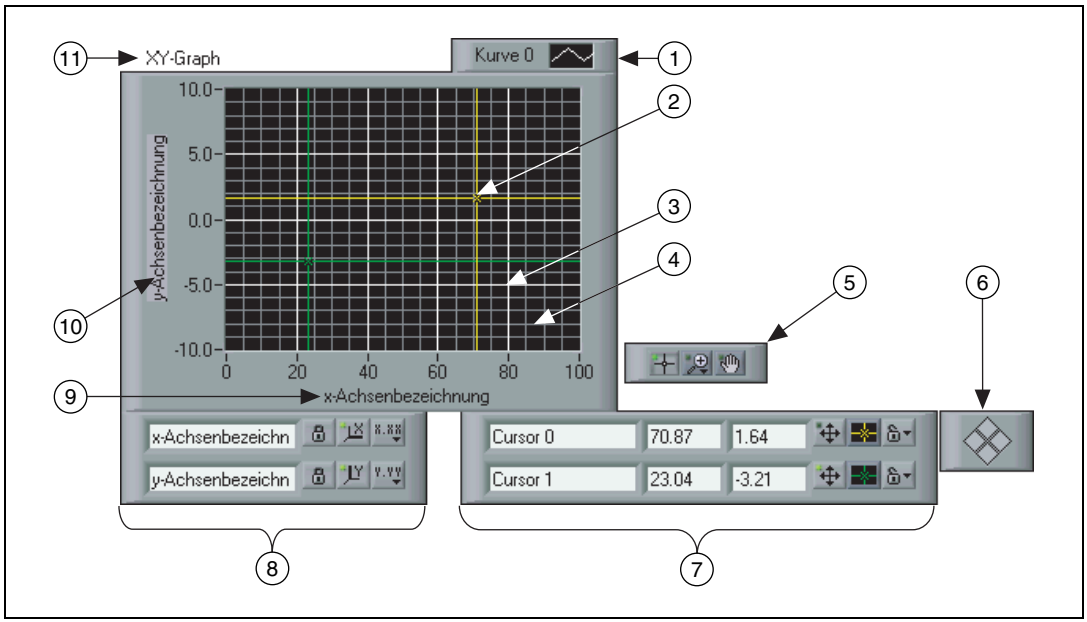
Sie können die Darstellung von Graphen und Diagrammen durch Ein- oder Ausblenden von Optionen anpassen. Klicken Sie mit der rechten Maustaste auf den Graphen oder das Diagramm und wählen aus dem Kontextmenü **Sichtbare Objekte**, um die folgenden Optionen ein- oder auszublenden:

- **Legende der Kurve**—Legt Farbe und Darstellungsart der Kurve fest. Wenn mehrere Kurven vorhanden sind, ziehen Sie die Legende auf, um alle Kurven anzuzeigen.
- **Achsenlegende**—Bestimmt die Achsenbeschriftungen und -eigenschaften.
- **Graphen-Palette**—Ändert Skalierung und Formatierung während der Ausführung eines VIs.
- **x- und y-Achse**—Dient zur Achsenformatierung. Weitere Informationen zu Achsen finden Sie im Abschnitt *Achsenformatierung* dieses Kapitels.
- **Cursor-Legende (nur bei Graphen)**—Zeigt an den angegebenen Koordinaten eine Markierung an. Es ist möglich, in einem Graphen mehrere Cursor anzuzeigen.
- **Bildlaufleiste**—Dient zum Scrollen in einem Graphen oder Diagramm.

Anpassen von Graphen

Das Verhalten des Graphen-Cursors, die Skalierungsoptionen und Achsen können verändert werden. Die Elemente eines Graphen sind in Abbildung 12-1 veranschaulicht.

Signalverlaufs- und Intensitätsgraphen können benutzerspezifisch angepasst werden, um sie nach bestimmten Kriterien oder mit Zusatzinformationen anzeigen zu lassen. Für Diagramme stehen eine Bildlaufleiste, eine Legende, eine Palette und eine Zahlenwertanzeige und Optionen für die Zeitdarstellung zur Verfügung.



1	Legende der Kurve	4	Teilgittermarkierung	7	Cursor-Legende	10	y-Achse
2	Cursor (nur bei Graphen)	5	Graphen-Palette	8	Achsenlegende	11	Beschriftung
3	Gittermarkierung	6	Cursor-Verschiebeelement	9	x-Achse		

Abbildung 12-1. Elemente eines Graphen

Die meisten der in der oben dargestellten Legende angezeigten Objekte lassen sich hinzufügen, indem Sie auf den Graphen klicken, aus dem Kontextmenü die Option **Sichtbare Objekte** und dann das entsprechende Element auswählen.

Graphen-Cursor

Mit Hilfe des Cursorkreuzes kann in Graphen der genaue Wert eines Punktes in einer Kurve angezeigt werden. Die Anzeige des Zahlenwertes erfolgt in der Cursor-Legende. Zur Anzeige des Cursorkreuzes klicken Sie mit der rechten Maustaste auf den Graphen, wählen im Kontextmenü die Option **Sichtbare Objekte»Cursor-Legende** und klicken auf eine beliebige Stelle in einer Zeile der Cursor-Legende, um das Cursorkreuz zu aktivieren. Wenn Sie mehrere Cursor verwenden möchten, erweitern Sie dazu mit dem Positionierwerkzeug die Cursor-Legende.

Cursorkreuze (die auch auf der Kurve beschriftet werden können) und eine entsprechende Zahlenwertanzeige sind zu allen Graphen verfügbar. Sie können einen Cursor auch in einer Darstellung verankern oder mehrere Cursor gleichzeitig verschieben. Hinsichtlich der Cursor-Anzahl bestehen keine Beschränkungen.

Automatische Skalierung

Die Achsen von Graphen können automatisch an die darzustellende Kurve angepasst werden. Ein solchen Verhalten wird als automatische Skalierung bezeichnet. Um die automatische Skalierung zu aktivieren bzw. zu deaktivieren, klicken Sie mit der rechten Maustaste auf den Graphen und wählen aus dem Kontextmenü **x-Achse»Autom. Skalierung x** oder **y-Achse» Autom. Skalierung y** aus. Standardmäßig ist die automatische Skalierung für Graphen aktiviert. Beachten Sie jedoch, dass durch die automatische Skalierung die Systemleistung beeinträchtigt werden kann.

Um die horizontale oder vertikale Achse direkt zu ändern, verwenden Sie das Bedien- bzw. das Beschriftungswerkzeug.

Achsenlegende für Kurvengraphen

Über die Achsenlegende sind Beschriftung und Eigenschaften der Achsen einstellbar.

Klicken Sie unter Verwendung des Bedienwerkzeugs auf die links angezeigte Schaltfläche **Achsenstil**, um Format, Genauigkeit und Abbildungsmodus zu konfigurieren.

Über die links dargestellte Schaltfläche **Skalierungssperre** kann die automatische Skalierung für jede Achse einzeln aktiviert bzw. deaktiviert werden.

Achsenformatierung

Um die Darstellungsart der x- und y-Achse eines Graphen oder Diagramms festzulegen, wählen Sie im Dialogfeld **Signalverlaufs-Eigenschaften** bzw. **Diagramm-Eigenschaften** die Seite **Format und Genauigkeit** aus.

Als Standardbeschriftung wird für die x-Achse `Zeit` und für die y-Achse `Amplitude` verwendet. Die Werte der x-Achse sind auf Fließkommadarstellung voreingestellt, und die der y-Achse passen sich automatisch an die Eingangswerte des Graphen an. Um die Konfiguration der Achsen zu ver-

ändern, klicken Sie den Graphen oder das Diagramm mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Eigenschaften**.

Das Zahlenformat für die Werte der Achsen kann im Dialogfeld **Signalverlaufs-Eigenschaften** bzw. **Diagramm-Eigenschaften** unter **Format und Genauigkeit** ausgewählt werden. Unter **Skalierungen** lassen sich Achsenbeschriftung und Skaleneinteilung verändern. Die Werte an der Achse sind standardmäßig so konfiguriert, dass ab der siebenten Stelle automatisch in die Exponentialschreibweise umgeschaltet wird.

Um die Textoptionen anzuzeigen, über die Format-Strings direkt eingegeben werden können, aktivieren Sie die Option **Fortgeschrittener Bearbeitungsmodus**. Mit der Eingabe der entsprechenden Format-Strings lassen sich nun die Achsen und das Zahlenformat der Achsenwerte benutzerspezifisch anpassen. Für nähere Hinweise zu diesem Thema lesen Sie bitte in der *LabVIEW-Hilfe* unter *Verwendung von Format-Strings* nach.

Dynamische Formatierung von Graphen

Zur automatischen Formatierung der Legende und des Zeitstempels der x-Achse verbinden Sie einen Signalverlaufsgraphen mit einem dynamischen Datentyp. Wenn Sie zum Beispiel das Express-VI “Signal simulieren” so konfigurieren, dass es ein Sinussignal erzeugt und mit absoluter Zeit arbeitet, und den Ausgang dieses VIs mit einem Signalverlaufsgraphen verbinden, stellt sich die Legende automatisch auf die Kurvenbeschriftung `Sinus` ein, und an der x-Achse werden Datum und Zeit der VI-Ausführung angezeigt.

Wenn die Beschriftung, die jeweils im dynamischen Datentyp enthalten ist, nicht verwendet werden soll, klicken Sie den Graphen mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Attribute ignorieren**. Um die Zeitstempel auszublenden, verfahren Sie analog und wählen die Option **Zeitstempel ignorieren**.

Hinweis Wenn in den im Graphen dargestellten Daten ein Zeitstempel für absolute Zeit enthalten ist, wählen Sie die Option **Zeitstempel ignorieren**, wenn dieser nicht verwendet werden soll. Weitere Erläuterungen zur Express-VIs und zur Verwendung des dynamischen Datentyps mit Graphen finden Sie im Handbuch *Erste Schritte mit LabVIEW*. Allgemeine Informationen zum dynamischen Datentyp erhalten Sie auch im Abschnitt *Dynamische Daten* des Kapitels 5, *Erstellen des Blockdiagramms*.

Glätten von Grafiken

Wenn ein Objekt aktualisiert wird und die Option “Grafik beim Zeichnen glätten” ist deaktiviert, wird der Inhalt des Objekts gelöscht und neu gezeichnet, was sich durch ein Flackern bemerkbar macht. Beim Glätten der Grafik wird dieses Flackern soweit wie möglich unterdrückt. Klicken Sie mit der rechten Maustaste auf den Graphen und wählen aus dem Kontextmenü **Fortgeschritten»Grafik glätten**. Zur Darstellung der Grafik wird dann ein Hintergrundspeicher verwendet. Dadurch kann jedoch – je nach verwendeten Computer und Grafiksystem – die Systemleistung beeinträchtigt werden.

Diagramme benutzerspezifisch anpassen

Im Gegensatz zu Graphen, die immer einen kompletten Signalverlauf anzeigen und damit bereits dargestellte und gespeicherte Daten immer wieder überschreiben, werden Diagramme periodisch aktualisiert und enthalten eine Historie der zuvor gespeicherten Daten.

Signalverlaufs- und Intensitätsdiagramme können benutzerspezifisch angepasst werden, um sie nach bestimmten Kriterien oder mit Zusatzinformationen anzeigen zu lassen. Für Diagramme stehen eine Bildlaufleiste, Legende, Palette und eine Zahlenwertanzeige sourie Optionen für die Zeitdarstellung zur Verfügung. Sie können das Verhalten von Historienlänge, Aktualisierungsmodus und Kurvenanzeige ändern.

Historienlänge

Datenwerte, die bereits zum Diagramm hinzugefügt wurden, werden in einen Puffer – die sogenannte Diagrammhistorie gespeichert. Die Standardgröße für den Puffer der Diagrammhistorie beträgt 1024 Datenwerte. Um den Historienpuffer zu konfigurieren, klicken Sie mit der rechten Maustaste auf das Diagramm und wählen aus dem Kontextmenü die Option **Historienlänge** aus. Um zurückliegende Daten anzuzeigen, kann eine Bildlaufleiste eingeblendet werden. Klicken Sie dazu mit der rechten Maustaste auf das Diagramm und wählen aus dem Kontextmenü die Option **Sichtbare Objekte»Bildlaufleiste** aus.

Aktualisierungsmodi von Diagrammen

Bei Diagrammen gibt es drei unterschiedliche Modi zur fortlaufenden Anzeige von Daten. Um einen Modus auszuwählen, klicken Sie mit der rechten Maustaste auf das Diagramm und wählen aus dem Kontextmenü

die Option **Fortgeschritten»Aktualisierungsmodus** aus. Hier stehen die Optionen **Streifendiagramm**, **Oszilloskopdiagramm** und **Laufdiagramm** zur Verfügung. Der Standardmodus ist **Streifendiagramm**.

- **Steifendiagramm**—Dient zur kontinuierlichen Datenanzeige, wobei die angezeigte Kurve von links nach rechts läuft. Neue Datenpunkte werden jeweils rechts an die bestehenden Datenpunkte angefügt. Ein Streifendiagramm funktioniert ähnlich wie ein XT-Schreiber.
- **Oszilloskopdiagramm**—Stellt einen Datensatz, wie zum Beispiel einen Impuls oder eine Kurve dar, wobei die Anzeige jeweils von links kontinuierlich wieder aufgebaut wird. Jeder neue Wert wird rechts an den vorhergehenden angefügt. Wenn die Kurve an der rechten Seite des Darstellungsbereiches angekommen ist, wird sie komplett gelöscht und von links nach rechts neu gezeichnet. Die Anzeige ähnelt der eines Oszilloskops.
- **Laufdiagramm**—Funktioniert ähnlich wie ein Oszilloskopdiagramm, jedoch mit dem Unterschied, dass die alten Datenwerte auf der rechten Seite und die neuen durch eine vertikale Linie getrennt auf der linken Seite angezeigt werden. Wenn die Kurve den rechten Rand des Darstellungsbereiches erreicht hat, wird sie nicht gelöscht, sondern läuft weiter. Ein Laufdiagramm ähnelt einer EKG-Anzeige.

Überlagerte Kurven und Stapelplots

Sie können in einem Diagramm mehrere Kurven unter Verwendung einer einzigen vertikalen Achse darstellen – den sogenannten überlagerten Kurven – oder unter Verwendung mehrerer vertikaler Achsen – den sogenannten Stapelplots. Abbildung 12-2 zeigt Beispiele für überlagerte Kurven und Stapelplots.

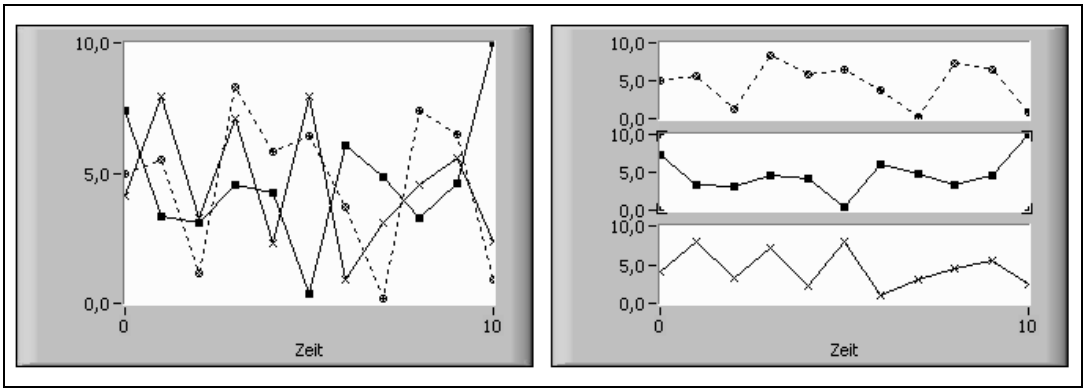


Abbildung 12-2. Diagramme mit überlagerten Kurven und Stapelplots

Um die Kurven eines Diagramms in Form mehrerer vertikaler Skalen anzuzeigen, klicken Sie das Diagramm mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Stapelplot** aus. Um die Kurve in Form einer vertikalen Skala anzuzeigen, klicken Sie das Diagramm mit der rechten Maustaste an und wählen Sie die Option **Überlagerte Kurven**.

Beispiele für die verschiedenen Diagramme und die jeweils zulässigen Datentypen finden Sie im VI “Charts” unter `examples\general\graphs\charts.llb`.

Kurven- und XY-Graphen

In Kurvengraphen werden in gleichen Abständen abgetastete Messungen angezeigt. XY-Graphen zeigen eine beliebige Wertemenge an, die gleichmäßig abgetastet sein kann oder nicht. In Abbildung 12-3 sehen Sie ein Beispiel für einen Kurvengraphen und einen XY-Graphen.

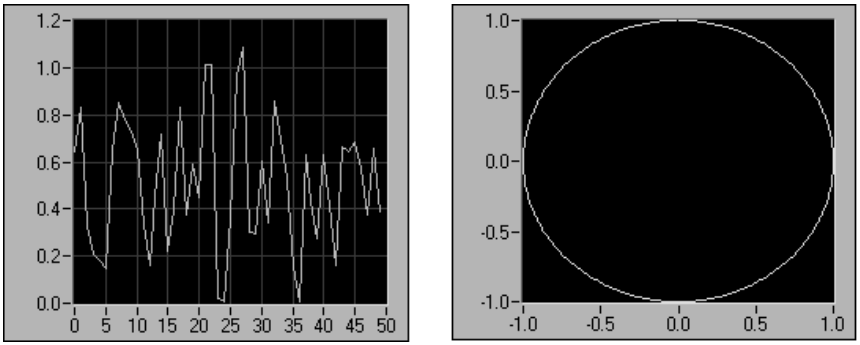


Abbildung 12-3. Kurven- und XY-Graph

Der Kurvengraph stellt nur einwertige Funktionen dar, wie in $y = f(x)$, wobei die Werte gleichmäßig über die x-Achse verteilt sind, wie zum Beispiel bei erfassten zeitabhängigen Signalen. Beim XY-Graph handelt es sich um ein kartesisches Diagrammobjekt zur grafischen Darstellung für allgemeine Zwecke, das mehrwertige Funktionen wie zum Beispiel Kreisformen oder Signalverläufe mit einer variablen Zeitbasis darstellt. Beide Graphen können Kurven mit einer beliebigen Anzahl von Werten anzeigen.

Beide Graphentypen arbeiten mit mehreren Datentypen, was den Zeitaufwand gering hält, da die Daten vor der Anzeige nicht umgewandelt werden müssen.

Datentypen für Kurvengraphen mit einer Kurve

Der Kurvengraph akzeptiert zwei Datentypen: zum einen ein einzelnes Array mit Werten, die als Y-Werte im Graph dargestellt werden. Der *X*-Index wird dabei beginnend bei $x = 0$ um jeweils um eins erhöht. Der Graph akzeptiert außerdem einen Cluster mit einem *x*-Anfangswert, einem Δx-Wert und einem Array von *Y*-Daten.

Beispiele für die Datentypen für Kurvengraphen zur Darstellung einer einzelnen Kurve finden Sie im VI Waveform Graph in `examples\general\graphs\gengraph.llb`.

Kurvengraph mit mehreren Kurven

An einen Kurvengraph für mehrere Kurven kann ein 2D-Array angeschlossen werden, wobei jede Zeile des Arrays die Daten eines einzelnen Signalverlaufs enthalten muss. Der Graph interpretiert die Zelleninhalte des Arrays als einzelne Datenwerte eines abgetasteten Signalverlaufs und erhöht den Index *x* beginnend bei $x = 0$ um jeweils eins. Verbinden Sie ein 2D-Array mit dem Graphen und klicken Sie ihn gegebenenfalls mit der rechten Maustaste an, um im Kontextmenü die Option **Array transponieren** auszuwählen, falls die Datenwerte eines Signalverlaufs in Spalten und nicht in Zeilen abgelegt sind. Dies ist insbesondere dann nützlich, wenn Sie mehrere Kanäle eines DAQ-Gerätes abtasten, da das Gerät die Daten als 2D-Arrays zurückgibt, wobei die Abtastwerte der einzelnen Kanäle in eigenständige Spalten gespeichert werden. Ein Beispiel für einen Graph, mit dem dieser Datentyp verwendbar ist, finden Sie im Graphen "(Y) Multi Plot 1" im VI "Waveform Graph" unter `examples\general\graphs\gengraph.llb`.

Ein Kurvengraph für mehrere Kurven akzeptiert außerdem einen Cluster mit einem *x*-Wert, einem Δx-Wert und einem 2D-Array von *y*-Werten. In diesem Fall interpretiert der Graph die *y*-Werte als Datenpunkte eines Signalverlaufes und inkrementiert den Index *x* beginnend bei $x = 0$ um jeweils Δx. Auf diese Weise lassen sich mehrere Signale skalieren, die alle mit der gleichen Abtastrate erfasst wurden. Ein Beispiel für einen Graph, der mit diesem Datentyp verwendbar ist, finden Sie im Graph "(Xo, dX, Y) Multi Plot 3" im VI "Waveform Graph" in der Datei `examples\general\graphs\gengraph.llb`.

Ein Kurvengraph mit mehreren Kurven arbeitet mit einen Kurvenarray, wobei das Array Cluster enthält. Jeder Cluster enthält ein Wertearray, das die *y*-Daten enthält. Das innere Array beschreibt die Werte einer Kurve, das äußere Array enthält einen Cluster für jede Kurve. Abbildung 12-4 zeigt dieses Array mit **y**-Clustern.

Abbildung 12-4. Array mit Clustern von y-Werten

Verwenden Sie einen Kurvengraphen mit mehreren Kurven anstelle eines 2D-Arrays, wenn die Anzahl der Elemente in den einzelnen Kurven nicht übereinstimmt. Verwenden Sie diese Datenstruktur anstelle eines 2D-Arrays, wenn Sie beispielsweise Daten mehrerer Kanäle abtasten und für jeden Kanal unterschiedliche Zeitintervalle verwenden, da jede Zeile eines 2D-Arrays die gleiche Anzahl von Elementen enthalten muss. Die Anzahl der Elemente in den inneren Arrays eines Arrays von Clustern kann variieren. Ein Beispiel für einen Graph, mit dem dieser Datentyp verwendbar ist, finden Sie im Graph “(Y) Multi Plot 2” im VI “Waveform Graph” in der Datei `examples\general\graphs\gengraph.llb`.

Für einen Kurvengraphen für mehrere Kurven ist ein Cluster mit einem x-Anfangswert, einem Δx-Wert und einem Array aus Clustern zulässig. Jeder Cluster enthält ein Wertearray, das die y-Daten enthält. Mit der Funktion “Elemente bündeln” können Sie die Arrays in Cluster bündeln, und mit der Funktion “Array erstellen” lässt sich aus den resultierenden Clustern ein Array erstellen. Alternativ kann aber auch die Funktion “Cluster-Array erstellen” verwendet werden, mit der Arrays aus Clustern erstellt werden, welche die angegebenen Eingänge enthalten. Ein Beispiel für einen Graph, der mit diesem Datentyp verwendbar ist, finden Sie im Graph “(Xo, dX, Y) Multi Plot 2” im VI “Waveform Graph” in der Datei `examples\general\graphs\gengraph.llb`.

Ein Kurvengraph für mehrere Kurven akzeptiert ein Array von Clustern mit einem x-Wert, einem Δx-Wert und einem Array von y-Daten. Dieser Datentyp bildet die universellste Form eines Kurvengraphen für mehrere Kurven, da Sie Anfangswert und Schrittweite für die x-Achse jeder Kurve eindeutig festlegen können. Ein Beispiel für einen Graph, der mit diesem Datentyp verwendbar ist, finden Sie im Graph “(Xo, dX, Y) Multi Plot 1” im VI “Waveform Graph” in der Datei `examples\general\graphs\gengraph.llb`.

Datentypen für XY-Graphen mit einer Kurve

Der XY-Graph für eine Kurve akzeptiert Cluster mit einem *x*- und einem *y*-Array sowie Wertearrays, in denen ein Wert aus einem Cluster besteht, der einen *x*-Wert und einen *y*-Wert enthält.

Beispiele für die Datentypen, die XY-Graphen mit einer Kurve akzeptieren, finden Sie in `examples\general\graph\gengraph.llb`.

Datentypen für XY-Graphen mit mehreren Kurven

XY-Graphen mit mehreren Kurven können mit einem Kurven-Array verbunden werden, in dem eine Kurve aus einem Cluster besteht, der ein *x*-Array und ein *y*-Array enthält. Statt dessen kann aber auch ein Array mit Clustern von Kurven verwendet werden, wobei eine Kurve jeweils einem Werte-Array entspricht. Ein Datenpunkt ist jeweils ein Cluster aus einem *x*- und einen *y*-Wert.

Beispiele für die Datentypen, die XY-Graphen mit mehreren Kurven akzeptieren, finden Sie im Verzeichnis `examples\general\graphs\gengraph.llb` befindet.

Kurvendiagramme

Bei einem Kurvendiagramm handelt es sich um einen speziellen Typ eines numerischen Anzeigeelements, das eine oder mehrere Kurven anzeigt. Beispiele für Kurvendiagramme finden Sie in `examples\general\graphs\charts.llb`.

Wenn an ein Diagramm ein Einzelwert oder mehrere Werte zur gleichen Zeit übergeben werden, interpretiert LabVIEW diese als Punkte des Diagramms und erhöht den *x*-Index beginnend bei $x = 0$ jeweils um eins, da die Daten als neue Werte für eine Einzelkurve verstanden werden. Bei Signalverlaufsdaten passt sich der *x*-Index an das angegebene Zeitformat an.

Wie oft das Diagramm neu gezeichnet wird, hängt davon ab, wie oft Daten übergeben werden.

Um Daten für mehrere Kurven an ein Kurvendiagramm zu übergeben, können Sie die Daten in einem Cluster skalarer Zahlen bündeln, wobei jede Zahl einen einzelnen Wert für die jeweilige Kurve darstellt.

Wenn Sie mehrere Werte für Kurven in einem einzigen Aktualisierungsvorgang übergeben möchten, verbinden Sie ein Array mit Clustern von Zahlen mit dem Diagramm. Jeder Zahlenwert stellt einen einzelnen *y*-Wert für die einzelnen Kurven dar.

Wenn die Anzahl der anzuzeigenden Kurven erst zur Laufzeit ermittelt werden kann oder wenn in einem einzigen Aktualisierungsvorgang mehrere Werte für verschiedene Kurven übergeben werden sollen, verbinden Sie das Diagramm mit einem 2D-Array mit numerischen Werten. Wie beim Kurvengraph interpretieren Kurvendiagramme Zeilen standardmäßig als neue Daten für die einzelnen Kurven. Verbinden Sie einen 2D-Array-Datentyp mit dem Diagramm, klicken Sie mit der rechten Maustaste auf das Diagramm und wählen Sie aus dem Kontextmenü **Array transponieren**, um die Spalten in dem Array für die einzelnen Kurven als neue Daten zu verwenden.

Intensitätsgraphen und -diagramme

Mit Intensitätsgraphen und -diagrammen können Sie 3D-Daten zweidimensional darstellen, indem Sie auf einer kartesischen Ebene Farbblöcke anzeigen. Intensitätsgraphen und -diagramme können zum Beispiel dazu genutzt werden, um Bitmuster darzustellen, wie beispielsweise Temperatur- oder Landkarten, auf denen Höhenunterschiede durch verschiedene Farben dargestellt werden. Intensitätsgraphen und -diagramme können mit 2D-Arrays mit numerischen Werten verbunden werden. Jede Zahl im Array stellt dann eine bestimmte Farbe dar, und die Indizes der Elemente bestimmen die Position der Farbpunkte. In Abbildung 12-5 ist die Funktionsweise eines Intensitätsgraphen dargestellt.

Eingangs-Array

Spalte = y

Reihe = x	0	1	2
0	50	50	13
1	45	61	10
2	6	13	5

Definitionstabelle

Array-Element = z	Farbe
5	blau
6	lila
10	hellrot
13	dunkelrot
45	orange
50	gelb
61	grün

Ergebnistabelle

dunkel-rot	hellrot	blau
gelb	grün	dunkel-rot
gelb	orange	lila

Abbildung 12-5. Farbzuordnung bei Intensitätsgraphen

Die Datenzeilen werden im Graph oder Diagramm zur Anzeige als neue Spalten übergeben. Wenn die Zeilen in der Anzeige als Zeilen angezeigt werden sollen, verbinden Sie ein 2D-Array mit dem Graph oder Diagramm, klicken mit der rechten Maustaste auf den Graph oder das Diagramm und wählen aus dem Kontextmenü **Array transponieren**.

Die Array-Indizes entsprechen dem unteren linken Eckpunkt des Farbblocks. Der Farbblock verfügt über einen Einheitsbereich. Dabei handelt es sich um den Bereich zwischen den beiden Punkten, wie durch die Array-Indizes festgelegt. Ein Intensitätsdiagramm oder -graph kann bis zu 256 diskrete Farben anzeigen.

Wenn Sie einen Datenblock auf einem Intensitätsdiagramm dargestellt haben, verschiebt sich der Ursprung der kartesischen Ebene auf die rechte Seite des letzten Datenblocks. Werden in dem Diagramm neue Daten verarbeitet, werden die neuen Daten rechts von den alten Daten angezeigt. Ist eine Diagrammanzeige voll, laufen die ältesten Daten auf der linken Seite aus dem Diagramm. Dieses Verhalten ähnelt dem Verhalten von Streifen-

schreiberdiagrammen. Weitere Informationen zu diesen Diagrammen finden Sie im Abschnitt *Aktualisierungsmodi von Diagrammen* dieses Kapitels.

Beispiele für Intensitätsgraphen und -diagramme finden Sie unter `examples\general\graphs\intgraph.llb`.

Farbzuordnung

Sie können die Farbzuordnung für Intensitätsgraphen und -diagramme genauso interaktiv festlegen wie Sie die Farben für einen numerischen Farbverlaufsbalken definieren. Weitere Informationen zu Farbrampen finden Sie in Abschnitt *Farbrampen* des Kapitels 4, *Erstellen des Frontpanels*.

Um die Farbzuordnung für Intensitätsgraphen und -diagramme programmatisch festzulegen, verwenden Sie den Eigenschaftsknoten auf zweierlei Art und Weise. Normalerweise wird die Zuordnung von Wert und Farbe im Eigenschaftsknoten definiert. Geben Sie dazu die Eigenschaft “Z-Achse: Marker-Werte” an. Diese Eigenschaft besteht aus einem Array mit Clustern, in dem jeder Cluster einen numerischen Grenzwert und die für den betreffenden Wert anzuzeigende Farbe enthält. Wenn Sie die Farbzuordnung auf diese Weise festlegen, können Sie mit der Eigenschaft “Z-Achse: High Color (16-Bit)” eine Farbe für eine Bereichsüberschreitung und mit der Eigenschaft “Z-Achse: Untere Farbe” eine Farbe für eine Bereichsunterschreitung festlegen. Intensitätsgraph und -diagramm sind auf insgesamt 254 Farben beschränkt. Zusammen mit den Farben für eine Bereichsüber- und -unterschreitung ergeben sich damit insgesamt 256 Farben. Wenn Sie mehr als 254 Farben angeben, erstellt der Intensitätsgraph beziehungsweise das Intensitätsdiagramm die Tabelle aus 254 Farben durch Interpolation der festgelegten Farben.

Wenn im Intensitätsgraphen eine Bitmap anzeigt wird, sollte die Farbtabelle mit Hilfe der Eigenschaft “Farbtabelle” festgelegt werden. Bei dieser Methode kann ein Array mit bis zu 256 Farben angegeben werden. Die an das Diagramm übergebenen Daten werden anhand der Farbskala des Intensitätsdiagramms den Indizes in dieser Farbtabelle zugeordnet. Wenn die Farbskalierung beispielsweise von 0 bis 100 reicht, würde der Datenwert 0 auf den Index 1 abgebildet werden und der Datenwert 100 auf den Index 254. Die Zwischenwerte würden auf 1 bis 254 aufgeteilt werden. Allen Werten kleiner 0 wird die für Low Color (Index 0) definierte Farbe zugewiesen und allen Werte größer 0 die Farbe für High Color (Index 255).

Hinweis Die Anzahl der in einem Intensitätsgraphen oder -diagramm darstellbaren Farben richtet sich nach der verwendeten Grafikkarte und Bildschirmauflösung.

Optionen für Intensitätsdiagramme

Das Intensitätsdiagramm verwendet viele der optionalen Bestandteile der Kurvendiagramme, die Sie ein- oder ausblenden können, indem Sie mit der rechten Maustaste auf das Diagramm klicken und **Sichtbare Objekte** auswählen. Da das Intensitätsdiagramm als dritte Dimension eine Farbe enthält, definiert darüber hinaus eine Skala (ähnlich wie beim Farbverlaufsbalken) den Bereich und die Zuordnungen von Werten und Farben.

Wie bei Kurvendiagrammen speichert das Intensitätsdiagramm den Datenverlauf, oder Puffer, der vorherigen Aktualisierungen. Sie können diesen Puffer konfigurieren, indem Sie mit der rechten Maustaste auf das Diagramm klicken und aus dem Kontextmenü **Historienlänge** auswählen. Die Standardgröße für ein Intensitätsdiagramm beträgt 128 Datenwerte. Die Anzeige eines Intensitätsdiagramms kann sehr speicherintensiv sein. Für ein Diagramm mit einfacher Genauigkeit, einer Historie von 512 Werten und 128 *y*-Werten sind beispielsweise 512 * 128 * 4 Bytes (Größe einer Zahl mit einfacher Genauigkeit) oder 256 KB erforderlich.

Optionen für Intensitätsgraphen

Der Intensitätsgraph funktioniert genauso wie das Intensitätsdiagramm, mit der Ausnahme, dass vorhergehende Daten nicht gespeichert werden und keine Aktualisierungsmodi enthalten sind. Immer, wenn neue Daten an einen Intensitätsgraph übergeben werden, ersetzen diese neuen Daten die alten.

Der Intensitätsgraph kann wie andere Graphen über Cursor verfügen. Jeder Cursor zeigt den *x*-, *y*- und *z*-Wert für einen bestimmten Punkt an.

Digitale Kurvengraphen

Digitale Signalverlaufsgraphen dienen zur Darstellung digitaler Daten und eignen sich insbesondere dann, wenn mit Zeitdiagrammen oder Logikanalysatoren gearbeitet wird. Weitere Informationen zur Erfassung digitaler Daten finden Sie im *LabVIEW Measurements Manual*.

An digitale Signalverlaufsgraphen können digitale Signalverlaufsdaten, digitale Daten oder Arrays aus Daten dieses Typs übergeben werden. Standardmäßig werden mehrere digitale Leitungen, die an einen Signalverlaufsgraphen übergeben werden, zusammengefasst, so dass nur ein einziger

Signalverlauf gezeichnet wird. Wenn der Graph mit einem Array aus digitalen Daten verbunden wird, stellt der Graph jedes Element als einzelne Kurve dar, wobei die Reihenfolge der Elemente im Array berücksichtigt wird.

Bei dem digitalen Signalverlaufsgraphen in Abbildung 12-6 werden die Daten in Form eines einzelnen Plots dargestellt. Die Werte im **Zahlen**-Array werden dazu in Digitaldaten umgewandelt und im Element **Binäre Darstellungen** angezeigt. Im digitalen Graphen erscheint die Zahl 0 ohne obere Linie, um anzuzeigen, dass alle Bitwerte 0 sind, und die Zahl 255 ohne untere Linie, um anzuzeigen, dass alle Bitwerte 1 sind.

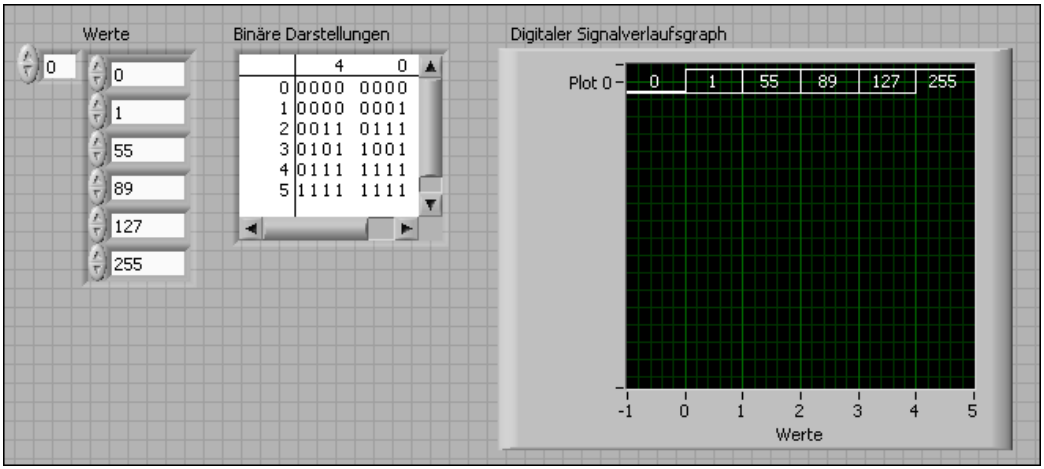


Abbildung 12-6. Darstellung digitaler Daten in einem einzelnen Plot

Bitte lesen Sie zur Konfiguration von Plots in digitalen Signalverlaufsgraphen auch die *LabVIEW-Hilfe*.

Um für jeden Digitalkanal einen Plot zu erstellen, klicken Sie mit der rechten Maustaste auf die y-Achse und wählen Sie aus dem Kontextmenü die Option **Digitalen Bus erweitern** aus. Jeder Plot steht dann für einen bestimmten Kanal.

Im Graphen in Abbildung 12-7 sind im digitalen Array-Bedienelement **Zahlen** sechs Werte dargestellt.

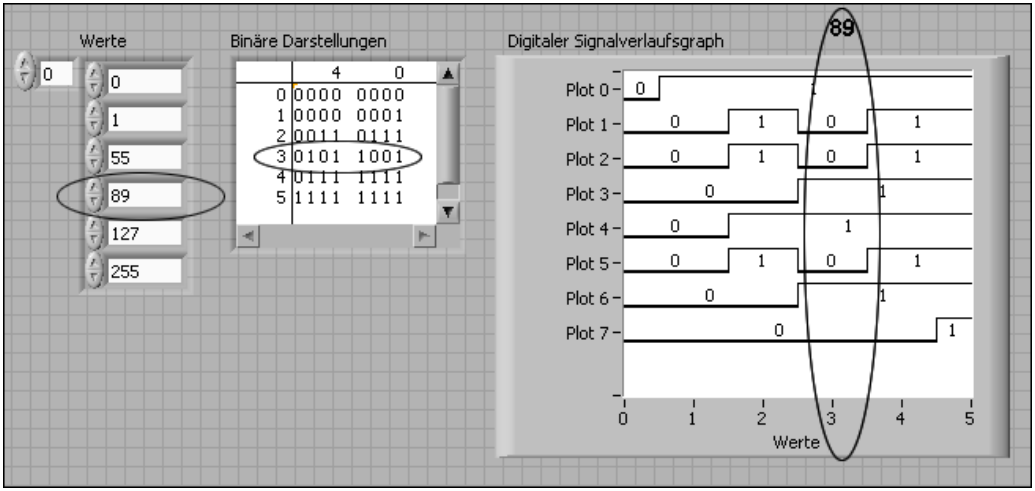


Abbildung 12-7. Digitale Grafikdarstellung ganzer Zahlen

Im digitalen Bedienelement **Binäre Darstellungen** werden Zahlen in Binärschreibweise dargestellt. Jede Spalte der Tabelle steht hierbei für ein Bit. So sind beispielsweise für die Zahl 89 sieben Bit Speicher erforderlich (der Wert `0` in Spalte 7 zeigt an, dass das Bit nicht verwendet wird). Im Punkt 3 des digitalen Signalverlaufsgraphen werden die sieben Bits zur Darstellung der Zahl 89 angezeigt und der Wert Null für das nicht benötigte achte Bit im Plot 7.

Im VI in Abbildung 12-8 wird ein Array mit numerischen Werten in digitale Daten umgewandelt. Mit Hilfe der Funktion “Signalverlauf erstellen” werden Startzeit, Delta t und die Zahlen festgelegt, die in das digitale Bedienelement eingegeben werden, und die digitalen Daten angezeigt.

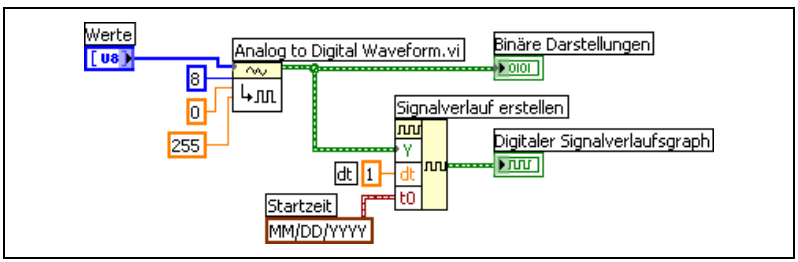


Abbildung 12-8. Umwandlung eines Zahlen-Arrays in digitale Daten

Für weitere Informationen zum digitalen Bedienelement, den Datentyp “digitaler Signalverlauf” und zur Umwandlung in digitale Daten lesen Sie bitte die Abschnitte *Bedienelement für digitale Signalverläufe* und *Bedienelement für digitale Daten* im Kapitel 4, *Erstellen des Frontpanels*.

Maskieren von Daten

Mit dem VI in Abbildung 12-7 wird ein Graph erzeugt, in dem jede Kurve ein Bit der Daten darstellt. Sie können vor der Anzeige auch Daten-Bits auswählen, neu anordnen und kombinieren. Das Auswählen, Anordnen und Kombinieren von Bits wird als Maskieren der Daten bezeichnet.

Mit Hilfe einer Maske können Sie die Kurven von zwei oder mehr verschiedenen Bits kombinieren und auf einer einzelnen Kurve darstellen. Bei einem Array von 8-Bit-Integern können bis zu 8 Bit in einer einzigen Kurve dargestellt werden. Bei einem Array von 16-Bit-Integern können bis zu 16 Bit in einer einzigen Kurve angezeigt werden und so weiter. Dasselbe Bit kann in einem einzelnen Plot auch mehrmals dargestellt werden.

3D-Graphen

Hinweis Bedienelemente für 3D-Graphen stehen unter Windows nur im Full Development System und Professional Development System von LabVIEW zur Verfügung.

Bei vielen in der Realität vorkommenden Datenmengen, wie zum Beispiel einer Oberflächentemperaturverteilung, verknüpften Zeit-/Frequenzanalysen und der Bewegung eines Flugzeugs, müssen Daten in drei Dimensionen visualisiert werden. Mit den 3D-Graphen können Sie dreidimensionale Daten visualisieren und die Anzeige der Daten ändern, indem Sie die Eigenschaften des 3D-Graphen bearbeiten.

Es stehen die folgenden 3D-Graphen zur Verfügung:

- **3D-Oberfläche**—Zeichnet in den dreidimensionalen Raum eine Oberfläche. Wenn Sie dieses Bedienelement auf dem Frontpanel ablegen, wird es mit einem SubVI verbunden, das die Daten für die Oberfläche enthält.
- **3D-Oberfläche (parametrisch)**—Zeichnet eine komplexe Oberfläche in einen 3D-Raum. Wenn Sie dieses Bedienelement auf dem Frontpanel ablegen, wird es mit einem SubVI verbunden, das die Daten für die Oberfläche enthält.

- **3D-Kurve**—Zeichnet eine Linie in einen 3D-Raum. Wenn Sie dieses Bedienelement auf dem Frontpanel ablegen, wird es von LabVIEW mit einem SubVI verbunden, das die Daten erhält, welche die Linie darstellen.

Zur Darstellung von Kurven und Oberflächen sollten die 3D-Graphen zusammen mit den 3D-Graph-VIs verwendet werden. Eine Kurve enthält einzelne Punkte im Graph, wobei jeder Punkt durch eine *x*-, *y*- und *z*-Koordinate definiert ist. Das VI verbindet diese Punkte dann zu einer Linie. Eine Kurve eignet sich ideal für die Visualisierung des Weges eines sich bewegenden Objekts, wie zum Beispiel des Flugwegs eines Flugzeugs.

3D-Graphen arbeiten mit ActiveX-Technologie und VIs zur 3D-Darstellung. Für die VIs für 3D-Graph-Eigenschaften sind verschiedene Einstellungen für die Ausführung wählbar (Grundeigenschaften, Eigenschaften für Achsen, Gitter, Projektionseigenschaften).

Bei einem Oberflächengraphen werden zur Darstellung der Punkte *x*-, *y*- und *z*-Werte benötigt. Der Oberflächengraph verbindet anschließend diese Punkte und bildet eine dreidimensionale Oberflächenansicht der Daten. Sie können einen Oberflächengraph beispielsweise zur topologischen Darstellung verwenden.

Bei Auswahl eines 3D-Graphen fügt LabVIEW auf dem Frontpanel einen ActiveX-Container ein, der ein Bedienelement für einen 3D-Graphen enthält. LabVIEW erstellt im Blockdiagramm außerdem eine Referenz auf diesen 3D-Graphen. LabVIEW verbindet diese Referenz mit einem der drei 3D-Graph-VIs.

Datentyp “Signalverlauf”

Der Datentyp “Signalverlauf” enthält Anfangszeit, Δx und die Daten eines Signalverlaufs. Mit Hilfe der Funktion “Signalverlauf erstellen” können Signalverläufe erzeugt werden. Viele VIs und Funktionen zur Erfassung und Analyse von Signalverläufen arbeiten standardmäßig mit dem Datentyp “Signalverlauf”. Wenn Sie den Datentyp ”Signalverlauf” mit einem Kurvengraph oder -diagramm verbinden, stellt der Graph beziehungsweise das Diagramm automatisch eine Kurve dar, die auf den Daten, der Anfangszeit und Δx dieses übergebenen Signalverlaufs basiert. Wenn Sie ein Array des Datentyps ”Signalverlauf” mit einem Kurvengraph oder -diagramm verbinden, stellt der Graph beziehungsweise das Diagramm automatisch alle Signalverläufe dar. Informationen zur Verwendung des Datentyps “Signalverlauf” finden Sie in Kapitel 5, *Creating a Typical Measurement Application* im *LabVIEW Measurements Manual*.

Datentyp “digitaler Signalverlauf”

Im Datentyp “digitaler Signalverlauf” sind Anfangszeit, Delta x und die Daten sowie die Attribute eines digitalen Signalverlaufs enthalten. Mit Hilfe der Funktion “Signalverlauf erstellen” können digitale Signalverläufe erzeugt werden. Wenn der Datentyp “digitaler Signalverlauf” mit einem entsprechenden Graphen verbunden wird, stellt dieser den Signalverlauf den mitgelieferten Zeitinformationen und Daten entsprechend dar. Zur Anzeige von Abtastwerten und Signalen des digitalen Signalverlaufes verbinden Sie den Datentyp mit einem Anzeigeelement für Digitaldaten. Weitere Informationen zum Datentyp “Signalverlauf” finden Sie in Kapitel 5, *Typical Measurement Applications*, des *LabVIEW Measurements Manual*.

13

VIs für Grafiken und Sound

Verwenden Sie die Grafik-VIs zum Anzeigen und Bearbeiten von Grafiken und die Sound-VIs zur Wiedergabe und Editierung von Sounds.

Weitere Informationen …

Weitere Informationen zur Verwendung von Grafiken und Sound in VIs finden Sie in der *LabVIEW-Hilfe*.

Verwenden des Bildanzeigeelements

Das Bildanzeigeelement dient zur Darstellung verschiedener Bildtypen. Diese können Linien, Kreise, Text und andere Arten grafischer Formen enthalten. Da das Bildanzeigeelement auf Pixelebene gesteuert werden kann, können Sie nahezu jedes beliebige Grafikobjekt erstellen. Mit Hilfe des Bildanzeigeelements und der Grafik-VIs können Grafiken in LabVIEW ähnlich wie in einem Bildverarbeitungsprogramm erstellt, bearbeitet und angezeigt werden.

Das Bildanzeigeelement besitzt ein pixelbasiertes Koordinatensystem, dessen Ursprung (0, 0) vom Pixel in der linken oberen Ecke gebildet wird. Der Wert für die horizontale Koordinate (x) nimmt nach rechts hin und für die vertikale Koordinate (y) nach unten hin zu.

Ist ein Bild zu groß, so dass es vom Bildanzeigeelement nicht angezeigt werden kann, wird es so zugeschnitten, dass nur die Pixel angezeigt werden, die in den Anzeigebereich des Elements passen. Mit dem Positionierwerkzeug können Sie die Größe des Anzeigeelements ändern und das VI erneut ausführen, damit das gesamte Bild angezeigt wird. Darüber hinaus ist es möglich, Bildlaufleisten einzublenden, mit denen alle Bildteile angezeigt werden können, die sich außerhalb des sichtbaren Bereichs des Anzeigeelements befinden. Klicken Sie dazu mit der rechten Maustaste auf Anzeigeelement und wählen aus dem Kontextmenü die Option **Sichtbare Objekte»Bildlaufleiste** aus.

Für programmatische Änderungen des Bildanzeigeelements (beispielsweise für eine Größenänderung des Elements oder Darstellungsbereichs) stehen Ihnen die VI-Server-Eigenschaften in der Bildklasse zur Verfügung. Weitere Informationen zu VI-Server-Eigenschaften finden Sie in Kapitel 17, *Programmatische Steuerung von VIs*.

Wenn Sie auf dem Frontpanel ein Bildanzeigeelement einfügen, wird es als leerer rechteckiger Bereich angezeigt. Im Blockdiagramm wird ein dazugehöriger Anschluss, wie links dargestellt, angezeigt.

Eine mit dem Bildanzeigeelement darzustellende Grafik muss immer programmatisch an das Element übergeben werden. Es ist also nicht möglich, beispielsweise ein Bild in die Zwischenablage zu kopieren und in das Element einzufügen. Für Zeichenanweisungen zur Grafik verwenden Sie die VIs für Bildfunktionen. Jedes VI übernimmt mehrere Eingaben, die eine Zeichenanweisung beschreiben. Diese werden vom VI zusammengefasst und anschließend zur Anzeige an das Bildanzeigeelement übergeben. Nähere Einzelheiten zur Verwendung der VIs für Bildfunktionen zur Darstellung von Grafiken in einem Bildanzeigeelement entnehmen Sie bitte dem Abschnitt *Bildfunktionen-VIs* dieses Kapitels.

VIs für Bildplots

Verwenden Sie die Bildplot-VIs, um mit dem Bildanzeigeelement gängige Typen von Graphen zu erstellen. Diese Graphen umfassen ein Polardiagramm, eine Kurvengraphanzeige, ein Smith-Diagramm und einen Graphenmaßstab.

Bei den Bildplot-VIs werden zur grafischen Anzeige der Daten systemnahe Zeichenfunktionen verwendet. Dadurch, dass sich mit den VIs der Zeichencode benutzerspezifisch anpassen lässt, sind die VIs vielfältig einsetzbar. Diese grafischen Anzeigen gestatten zwar eine geringere Interaktivität als die integrierten LabVIEW-Bedienelemente; jedoch können Daten damit auf eine andere Art und Weise visualisiert werden als momentan mit den integrierten Bedienelementen möglich ist. So kann mit dem VI “Signalverlauf zeichnen” ein Plot mit einer geringfügig anderen Funktion erstellt werden als der von LabVIEW-Graphen für Signalverläufe.

Verwendung des VIs “Polar Plot” als SubVI

Das VI “Polarplot” dient zur Darstellung eines polaren Graphen oder bestimmter zusammenhängender Quadranten dieses Graphen. Wie bei den integrierten LabVIEW-Graphen können Sie die Farbe der Komponenten angeben, Gitter einfügen sowie Bereich und Format der Achsen festlegen.

Dieses VI vereinigt eine Vielzahl von Funktionen. Daher sind seine Eingänge komplexe Cluster. Für eine einfache Handhabung des VIs empfiehlt es sich, Standardwerte und benutzerspezifische Bedienelemente zu verwenden. Anstatt im Blockdiagramm einen eigenen Eingangs-Cluster zu erstellen, kann beispielsweise ein benutzerspezifisches Bedienelement aus dem VI "Polar Plot Demo" unter `examples\picture\demos.llb` kopiert und in das Frontpanel eingefügt werden.

Verwendung der VIs "Signalverlauf zeichnen" und "XY-Graph" als SubVIs

Das VI "Signalverlauf zeichnen", das das Verhalten des LabVIEW-Kurvengraphen emuliert, ermöglicht die Erstellung verschiedener Kurventypen, wie zum Beispiel aus Punkten, Verbindungslinien oder Balken bestehende Kurven. Wie bei den LabVIEW-Graphen können Sie die Farbe der Komponenten festlegen, Gitter einfügen sowie Bereich und Format der Achsen festlegen.

Das VI "Signalverlauf zeichnen" bietet eine Vielzahl von Funktionen. Die Eingänge des VIs bilden komplexe Cluster. Für eine einfache Handhabung des VIs empfiehlt es sich, Standardwerte und benutzerspezifische Bedienelemente zu verwenden. Anstatt einen eigenen Eingangs-Cluster zu erstellen, können Sie im VI "Waveform and XY Plots" unter `examples\picture\demos.llb` ein benutzerdefiniertes Bedienelement kopieren und dieses in das Frontpanel des VIs "Signalverlauf zeichnen" einfügen.

Das VI "Signalverlauf zeichnen" ähnelt den VIs "XY-Plot" und "Multi-XY-Plot". Sie können mit verschiedenen Bedienelementen die Darstellung ändern, da die VIs für die XY-Darstellung drei Darstellungsarten bieten: zwei Streudiagrammarten und eine Darstellungsart, bei der an jeder eindeutigen x-Position eine Linie gezeichnet wird, um die minimalen und maximalen y-Werte für den betreffenden x-Wert zu markieren.

Verwendung des VIs "Smith-Plot" als SubVI

Mit Hilfe des VIs "Smith-Plot" kann, zum Beispiel in der Telekommunikationsbranche, das Verhalten eines Übertragungsmediums untersucht werden. Ein Übertragungsmedium dient zur Übermittlung von Energie und Signalen. Dabei kann es sich zum Beispiel um ein Kabel handeln oder die Atmosphäre, über die das Signal übertragen wird. Übertragungsleitungen beeinflussen das übertragene Signal. Dieser Effekt wird als Impedanz bezeichnet und äußert sich als Dämpfung oder Phasenverschiebung von Wechselstromsignalen.

Die Impedanz (der Scheinwiderstand) einer Übertragungsleitung bildet ein Maß für den Wirk- und den Blindwiderstand der Leitung. Die Impedanz, Z, ist in der Regel eine komplexe Zahl, die über die Gleichung $Z = R + jX$ berechnet wird, wobei R der Wirkwiderstand und X der Blindwiderstand ist.

Mit dem VI “Smith-Plot” können die Impedanzen von Übertragungsleitungen grafisch dargestellt werden. Das Diagramm besteht aus Kreisen mit konstantem Wirk- und Blindwiderstand.

Eine gegebene Impedanz, $R + jX$, kann dargestellt werden, indem der Schnittpunkt des entsprechenden R- und X-Kreises ermittelt wird. Nach der Anzeige der Impedanz können Sie das VI “Smith-Plot” als visuelle Hilfe verwenden, um die Impedanz anzupassen und den Reflektionsfaktor einer Übertragungsleitung zu berechnen.

Jedes einzelne VI des VIs “Smith-Plot” bietet eine Vielzahl von Funktionen. Die Eingänge vieler dieser VIs sind daher komplexe Cluster. Um das VI überschaubar zu gestalten, stehen jedoch Standardwerte zur Verfügung und es können benutzerdefinierte Bedienelemente verwendet werden. Anstatt einen eigenen Eingangs-Cluster zu erstellen, kann beispielsweise ein benutzerdefiniertes Bedienelement aus dem VI “Smith Plot Example” unter `examples\picture\demos.llb` kopiert und in das Frontpanel des betreffenden VIs eingefügt werden.

Bei der grafischen Anzeige von Lastimpedanzen lässt sich eine Impedanz als komplexe Zahl der Form $R + jX$ darstellen.

Damit im Smith-Plot keine Einzelheiten verloren gehen, empfiehlt es sich, die Daten mit dem VI “Smith-Plot normieren” zu normalisieren. Die von diesem VI ausgegebenen Daten können direkt an das VI “Smith-Plot” übergeben werden. Die Normalisierung erfolgt üblicherweise in Bezug auf die charakteristische Impedanz (Z_0) des Systems.

Bildfunktionen-VIs

Zum Zeichnen oder zur Texteingabe in ein Bildanzeigeelement stehen Ihnen die VIs für Bildfunktionen zur Verfügung. Damit können Punkte, Linien, geometrische Figuren und Pixmaps gezeichnet werden. Bei Pixmaps umgewandelter Daten handelt es sich um 2D-Arrays mit Farben. Je nach Farbtiefe entspricht jeder darin enthaltene Wert einer Farbe oder einem Array-Index, der einem RGB-Farbwert zugeordnet ist.

Die erste Zeile der Palette **Bildfunktionen** enthält VIs, mit denen Punkte und Linien gezeichnet werden können. Ein Punkt besteht aus einem Cluster aus 16-Bit-Ganzzahlen mit Vorzeichen, welche die x- und y-Koordinaten eines Pixels darstellen.

Bei Verwendung der VIs für Bildfunktionen wird die Position des Grafikstifts für das Bild gespeichert. Bei den meisten Bildfunktionen-VIs müssen die Koordinaten absolut angegeben werden, also relativ zum Koordinatenursprung (0,0). Bei den VIs “Linie zeichnen” und “Stift bewegen” ist die Angabe der Koordinaten sowohl absolut als auch relativ möglich. Bei relativen Koordinaten dient die aktuelle Position des Stifts als Bezugspunkt. Wenn Sie die Position des Zeichenstifts verändern möchten, ohne dabei zu zeichnen, verwenden Sie das VI “Stift bewegen”. Ansonsten kann die Stiftposition auch mit den VIs “Punkt zeichnen”, “Linie zeichnen” und “Mehrere Linien zeichnen” geändert werden.

In der zweiten Zeile der Palette **Bildfunktionen** befinden sich VIs, mit denen ausgefüllte geometrische Figuren gezeichnet werden können. Jedes einzelne VI zeichnet eine Figur in einem rechteckigen Bereich eines Bilds. Dazu legen Sie über einen Cluster mit vier Werten ein Rechteck fest. Diese Werte stehen für das obere, untere, rechte und linke Pixel.

Mit den VIs in der dritten Zeile der Palette **Bildfunktionen** kann Text in ein Bild gezeichnet werden. Ausgenommen davon ist das VI “Textrechteck lesen”, mit dem die Größe der Umrandung eines Strings berechnet wird.

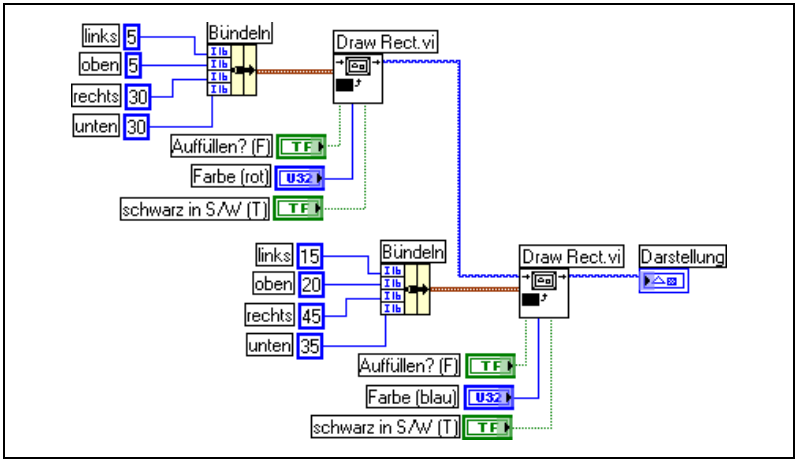
In der vierten Reihe der Palette **Bildfunktionen** finden Sie VIs zum Einzeichnen von Pixmaps in Bilder, zum Maskieren eines Bilds, zum Erfassen eines Bildausschnitts oder zum Konvertieren eines Bilddatentyps in einen eindimensionalen Bilddaten-Cluster.

Die letzte Zeile der Palette Bildfunktionen enthält die Konstante “Leeres Bild”, die immer dann zu verwenden ist, wenn mit einem leeren Bild begonnen werden oder ein leeres Bild verändert werden muss. Darüber hinaus befinden sich hier die VIs, mit denen die Werte für die Farben Rot, Grün und Blau in die entsprechende RGB-Farbe umgewandelt oder Farben in ihre RGB-Bestandteile aufgeschlüsselt werden können.

Die mit den VIs für Bildfunktionen erstellten Grafiken können ausschließlich an ein Bildanzeigeelement oder den Eingang **Bild** eines Bildfunktionen-VIs übergeben werden. Das Bild wird gezeichnet, wenn das Bildanzeigeelement bei geöffnetem Frontpanel aktualisiert wird.

Jedes Bildfunktionen-VI verkettet seine Zeichenanweisungen mit denen am Eingang **Bild**. Die Ausgabe der neuen Anweisungen erfolgt am Ausgang **Neues Bild**.

Im folgenden Beispiel werden mit Hilfe des VIs “Rechteck zeichnen” zwei sich überlappende Rechtecke gezeichnet.



Erstellen und Ändern von Farben mit den Bildfunktionen-VIs

Viele Bildfunktionen-VIs haben einen Eingang mit der Bezeichnung **Farbe**, über den die Farbe von Text und anderen Teilen einer Grafik verändert wird. Es empfiehlt sich, diesen Eingang mit einer Farbfeldkonstante zu verbinden, denn so kann die Farbe durch einen einfachen Mausklick ausgewählt werden.

Um Farben als Ergebnis von Berechnungen und nicht mit den Farbfeldkonstanten zu erstellen, müssen Sie wissen, wie ein Farbfeld eine Farbe mit einem numerischen Wert festlegt.

Eine 32-Bit-Ganzzahl mit Vorzeichen stellt eine Farbe dar, wobei die drei niederwertigen Bytes für den roten, grünen und blauen Bestandteil der Farbe stehen. Für einen Bereich blauer Farben würden Sie ein Array mit 32-Bit-Ganzzahlen erstellen, in dem jedes Element einen anderen Blau-Wert besitzt, der jedoch jeweils größer als der Rot- und Grün-Anteil ist. Erstellen Sie für einen grauen Farbbereich ein Array mit 32-Bit-Ganzzahlen, in dem die roten, grünen und blauen Werte jedes Elements gleich sind.

VIs für Grafikformate

Verwenden Sie die VIs für Grafikformate, wenn Daten aus Dateien verschiedener Standardformate für Grafiken gelesen und in Dateien geschrieben werden sollen, wie zum Beispiel `.bmp` (Bitmap), `.png` (Portable Network Graphics) oder `.jpg` (Joint Photographic Experts Group). Die VIs eignen sich:

- Zum Lesen von Daten aus einer Grafikdatei zur Anzeige in einem Bildanzeigeelement.
- Zum Lesen von Daten für die Bildbearbeitung.
- Zum Schreiben einer Grafik zur Anzeige in anderen Applikationen.

Bei Bitmap-Daten handelt es sich um 2D-Arrays, in denen jeder Punkt je nach Farbtiefe variiert. Bei einem Schwarz-weiß-Bild (1-Bit-Bild) ist jeder Punkt vom Typ “boolesch”. Bei 4-Bit- und 8-Bit-Bildern stellt jeder Punkt einen Index in einer Farbtabelle dar. Bei 24-Bit-Bildern mit Echtfarben setzt sich jeder Punkt aus drei Werten zusammen, nämlich denen für den Rot-, Grün- und Blau-Anteil (RGB).

Die VIs, die Grafikdateien lesen und schreiben, arbeiten mit Daten in einem einfachen, unstrukturierten Format, das eher der Art und Weise entspricht, in der Grafikdateien auf einen Datenträger geschrieben werden, wobei die Daten in ein eindimensionales Array gespeichert werden. Bei diesen Grafikdateien handelt es sich um Pixmaps, deren Konzept dem von Bitmaps ähnelt. Die erzeugten Daten lassen sich direkt mit dem VI “Geglättetes Pixmap zeichnen” darstellen.

Um die Daten als 2D-Array zu bearbeiten, konvertieren Sie sie mit Hilfe der VIs “1D-Pixmap nach 2D” und “2D-Pixmap nach 1D”.

Sound-VIs

Um Sounddateien und -funktionen in Ihre VIs einzubeziehen, stehen Ihnen die Sound-VIs zur Verfügung. Diese VIs eignen sich für folgende Aufgaben:

- Erstellen von VIs, die Audiodateien wiedergeben, wie beispielsweise eine aufgezeichnete Warnung, wenn Anwender bestimmte Aktionen ausführen.
- Erstellen eines VIs, das zu Beginn, am Ende oder einer sonstigen Stelle des VIs eine Audiodatei wiedergibt.

- Konfigurieren eines Geräts zur Erfassung von Sounddaten mit Hilfe der VIs zur Sound-Aufnahme, wobei beliebige Audiodaten gelesen werden können.
- Konfigurieren eines Wiedergabegeräts, so dass es Daten anderer Sound-VIs verarbeiten kann. Neben Funktionen wie Abspielen, Anhalten oder Löschen von Audiodateien ist auch eine Lautstärkeregelung möglich.

14

Datei-I/O

Bei Datei-I/O-Operationen werden Daten aus Dateien ausgelesen oder in Dateien geschrieben. Die VIs zur Datei-I/O eignen sich für alle mit der Datei-I/O in Verbindung stehenden Vorgänge, wie beispielsweise:

- Öffnen und Schließen von Dateien.
- Lesen von Daten aus und Schreiben von Daten in Dateien.
- Lesen von Daten aus und Schreiben von Daten in Dateien im Tabellenkalkulationsformat.
- Verschieben und Umbenennen von Dateien und Verzeichnissen.
- Ändern von Dateieigenschaften.
- Erstellen, Ändern und Lesen einer Konfigurationsdatei.

High-Level-VIs sollten für häufig durchzuführende I/O-Operationen verwendet werden und Low-Level-VIs sowie -Funktionen, wenn jede Operation einzeln gesteuert werden soll.

Weitere Informationen ...

Weitere Informationen zu Datei-I/O-Operationen finden Sie in der *LabVIEW-Hilfe*.

Grundlagen der Datei-I/O

Bei einer Datei-I/O-Operation sind gewöhnlich folgende Schritte erforderlich:

1. Erstellen oder Öffnen einer Datei. Dazu muss über einen Pfad oder ein Dialogfeld angegeben werden, wo sich die entsprechende Datei befindet oder wo eine neue Datei erstellt werden soll. Nach dem Öffnen erhält die Datei eine RefNum, über die auf die Datei verwiesen wird. Weitere Informationen zu RefNums entnehmen Sie bitte dem Abschnitt *Referenzen auf Objekte oder Applikationen* des Kapitels 4, *Erstellen des Frontpanels*.
2. Durchführen des Lese- oder Schreibvorgangs.
3. Schließen der Datei.

Die meisten Datei-I/O-VIs und -Funktionen führen immer nur einen Schritt innerhalb einer Operation aus. Es gibt jedoch auch High-Level-VIs, bei denen alle drei Arbeitsvorgänge auf einmal ausgeführt werden. Obwohl diese VIs nicht immer so leistungsfähig sind wie die Low-Level-Funktionen, sind sie eventuell einfacher zu verwenden.

Auswahl eines Datei-I/O-Formats

Welches VI zur Datei-I/O eingesetzt wird, hängt davon ab, ob es sich um Text-, Binär- oder Datenprotokolldateien handelt. Das verwendete Format hängt sowohl von den erfassten oder erstellten Daten als auch von den Applikationen ab, die auf die Daten zugreifen.

Grundsätzlich lässt sich das zu verwendende Format anhand folgender Richtlinien ermitteln:

- Wenn die Daten anderen Applikationen wie zum Beispiel Microsoft Excel zur Verfügung gestellt werden sollen, verwenden Sie Textdateien, da diese am häufigsten verwendet werden und am einfachsten portiert werden können.
- Wenn wahlfreie Schreib- oder Lesezugriffe durchgeführt werden müssen oder Geschwindigkeit und Festplattenspeicherplatz von Bedeutung sind, verwenden Sie Binärdateien, da sie weniger Festplattenspeicherplatz erfordern und schneller arbeiten als Textdateien.
- Wenn in LabVIEW komplexe Datensätze oder unterschiedliche Datentypen bearbeitet werden müssen, sollten Datenprotokolldateien verwendet werden, da sie sich optimal zum Speichern von Daten eignen, sofern der Zugriff auf die Daten lediglich von LabVIEW aus erfolgen soll und komplexe Datenstrukturen gespeichert werden müssen.

Einsatz von Textdateien

Dateien im Textformat sollten immer dann verwendet werden, wenn die Daten anderen Anwendern oder Applikationen zur Verfügung gestellt werden sollen, kein wahlfreier Schreib- bzw. Lesezugriff erfolgen soll oder numerische Genauigkeit, Festplattenspeicherplatz und Geschwindigkeit der Datei-I/O nicht von Bedeutung sind.

Für eine gemeinsame Nutzung sind Textdateien am besten geeignet. Sie können von nahezu jedem Computer gelesen und geschrieben werden. Eine Vielzahl von textbasierten Programmen kann textbasierte Dateien lesen. Auch die meisten Applikationen zur Gerätesteuerung verwenden Text-Strings.

Daten, auf die von einer anderen Applikation aus zugegriffen werden soll, beispielsweise von einer Textverarbeitungs- oder Tabellenkalkulationsanwendung, sollten in Textdateien gespeichert werden. Konvertieren Sie dazu mit Hilfe der String-Funktionen alle Daten in Textstrings. Textdateien können Daten unterschiedlicher Typen enthalten.

Textdateien belegen normalerweise mehr Speicherplatz als Binär- oder Datenprotokolldateien, wenn die Daten ursprünglich nicht als Text, sondern zum Beispiel als Graphen- oder Diagrammdaten vorliegen, da die ASCII-Darstellung von Daten in der Regel eine größere Bitbreite erfordert. Sie können beispielsweise die Zahl –123,4567 als Fließkommazahl einfacher Genauigkeit in 4 Bytes speichern. In der ASCII-Darstellung werden dagegen 9 Byte benötigt (ein Byte für jedes Zeichen).

Darüber hinaus erweist sich der wahlfreie Zugriff auf numerische Daten in Textdateien als schwierig. Obwohl jedes Zeichen in einer Zeichenfolge genau 1 Byte belegt, gibt es keine feste Breite, mit der zum Beispiel eine Zahl ausgedrückt wird. Um also beispielsweise die neunte Zahl in einer Textdatei zu finden, muss LabVIEW zuerst die vorhergehenden acht Zahlen lesen und konvertieren.

Beim Speichern von numerischen Daten in Textdateien kann sich die Genauigkeit verringern, da solche Daten, die normalerweise in Binärschreibweise gespeichert werden, in Textdateien in dezimaler Darstellung geschrieben werden. Die Genauigkeit kann sich auch beim Schreiben der Daten in eine Textdatei verringern. Dieses Problem tritt bei Binärdateien nicht auf.

Beispiele für die Verwendung der Datei-I/O mit Textdateien finden Sie unter `examples\file\smplfile.llb` und `examples\file\sprdsht.llb`.

Einsatz von Binärdateien

Beim Speichern binärer Daten, beispielsweise eines ganzzahligen Werts, wird eine feste Byteanzahl verwendet. Wenn beispielsweise eine Zahl von 0 bis 4 Milliarden im Binärformat gespeichert wird (1, 1000 oder 1000000), belegt jede Zahl 4 Byte Speicherplatz.

Die Verwendung von Binärdateien bietet sich besonders zum Speichern numerischer Daten an oder, um Zufallszahlen auszulesen bzw. gezielt auf Zahlen in einer Datei zuzugreifen. Im Gegensatz zu Textdateien sind Binärdateien nur von Rechnern lesbar. Sie bieten das kompakteste und am schnellsten zu speichernde Datenformat. Die Verwendung unterschiedlicher Datentypen ist bei diesem Dateityp zwar möglich, jedoch unüblich.

Binärdateien sind weniger speicher- und zeitaufwendig als Textdateien, da sie weniger Festplattenspeicherplatz belegen und beim Speichern und Abrufen von Daten nicht in eine oder aus einer Textdarstellung konvertiert werden müssen. Auf einen Speicherplatz von einem Byte können bei Binärdateien 256 Werte gespeichert werden. Außer bei erweiterten und komplexen Zahlen enthalten Binärdateien oftmals ein Byte-für-Byte-Abbild der Daten im Speicher. In diesem Fall erfolgt das Lesen der Daten schneller, da keine Konvertierung notwendig ist. In der Application Note *LabVIEW Data Storage* finden Sie weitere Informationen über die in LabVIEW eingesetzten Speichermechanismen.

Hinweis Text- und Binärdateien werden auch als Bytestream-Dateien bezeichnet, da Daten darin als Zeichen- oder Bytefolge gespeichert sind.

Beispiele zum Datenaustausch zwischen Arrays mit Fließkommazahlen doppelter Genauigkeit und Binärdateien finden Sie in den VIs “Read Binary File” und “Write Binary File” unter `examples\file\smplfile.llb`.

Einsatz von Datenprotokolldateien

Verwenden Sie Datenprotokolldateien, um nur in LabVIEW auf Daten zuzugreifen und diese zu bearbeiten sowie komplexe Datenstrukturen schnell und einfach zu speichern.

Bei einer Datenprotokolldatei werden Daten als Folge identisch aufgebauter Datensätze gespeichert, ähnlich wie bei einer Tabellenkalkulationsdatei, bei der jede Zeile einen Datensatz darstellt. Jedem Datensatz in einer Datenprotokolldatei müssen dieselben Datentypen zugeordnet sein. Geschrieben werden die Datensätze als Cluster, in dem die zu speichernden Daten enthalten sind. Die Komponenten eines Datenprotokolldatensatzes können jedoch beliebige Datentypen aufweisen. Diese werden beim Erstellen der Datei festgelegt.

So kann beispielsweise ein Datenprotokoll erstellt werden, dessen Datentyp ein Cluster aus einem String und einer Zahl ist. Jeder Datensatz des Datenprotokolls bildet dann einen Cluster aus einem String und einer Zahl. Die Reihenfolge ist jedoch nicht vorgegeben. So könnte der erste Datensatz zum Beispiel (`“abc”,1`) lauten und der zweite (`“xyz”,7`).

Die Daten in Datenprotokolldateien müssen kaum bearbeitet werden, so dass Schreib- und Lesevorgänge wesentlich schneller erfolgen. Auch das Abrufen von Daten ist einfacher, da die ursprünglichen Datenblöcke als Datensatz gelesen werden können, wobei ein Auslesen der davor liegenden

Datensätze entfällt. Ein wahlfreier Zugriff ist bei Datenprotokolldateien schnell und einfach möglich, da dazu lediglich die Datensatznummer benötigt wird. Beim Erstellen der Datenprotokolldatei werden die Datensätze automatisch fortlaufend nummeriert.

In LabVIEW kann sowohl über das Frontpanel als auch über das Blockdiagramm auf Datenprotokolldateien zugegriffen werden. Weitere Informationen über den Zugriff auf Datenprotokolldateien vom Frontpanel aus finden Sie im Abschnitt *Protokollieren von Frontpanel-Daten* dieses Kapitels.

LabVIEW schreibt immer dann einen Datensatz in die Datenprotokolldatei, wenn das entsprechende VI ausgeführt wird. Diese Datensätze können nicht überschrieben werden. Das Lesen einer Datenprotokolldatei kann datensatzweise oder auch mit mehreren Datensätzen gleichzeitig erfolgen.

Beispiele zum Lesen und Schreiben von Datenprotokolldateien finden Sie im Verzeichnis `examples\file\datalog.llb`.

Verwenden von High-Level-Datei-I/O-VIs

Die High-Level-VIs zur Datei-I/O bieten sich insbesondere für häufig durchzuführende Operationen der Datei-I/O an, wie beispielsweise:

- Lesen und Schreiben von Zeichen aus bzw. in Textdateien.
- Auslesen von Zeilen aus Textdateien.
- Lesen bzw. Schreiben von 1D- oder 2D-Arrays aus Zahlen einfacher Genauigkeit aus bzw. in Textdateien einer Tabellenkalkulation.
- Lesen bzw. Schreiben von 1D- oder 2D-Arrays aus Zahlen einfacher Genauigkeit oder aus vorzeichenbehafteten 16-Bit-Ganzzahlen aus bzw. in Binärdateien.

Die High-Level-VIs sind zeitsparend und erfordern einen geringen Programmieraufwand, da mit einem VI alle für den Lese- bzw. Schreibvorgang notwendigen Schritte ausgeführt werden (Öffnen, Lesen/Schreiben, Schließen). Da die Funktionen “Öffnen” und “Schließen” nur jeweils einmal ausgeführt werden müssen, sollten diese VIs möglichst nicht in Schleifen verwendet werden. Informationen darüber, wie Dateien für mehrere Operationen geöffnet gelassen werden, finden Sie im Abschnitt *Datenträger-Streaming*.

Beim Einsatz von High-Level-VIs muss immer der Speicherort der Datei für den Schreib- oder Lesevorgang angegeben werden. Wenn Sie das VI nicht mit einem Pfadnamen verbinden, wird ein Dialogfenster angezeigt,

in dem Sie eine Datei auswählen können. Bei High-Level-VIs aufgetretene Fehler werden in einem Dialogfeld beschrieben. Sie können die Ausführung beenden oder fortsetzen.

In Abbildung 14-1 sehen Sie ein Beispiel für die Anwendung des High-Level-VIs “In Spreadsheet-Datei schreiben”, um Zahlen an eine Microsoft-Excel-Datei zu übergeben. Bei Ausführung dieses VIs werden Sie von LabVIEW aufgefordert, die Daten in eine vorhandene Datei zu schreiben oder eine neue Datei zu erstellen.

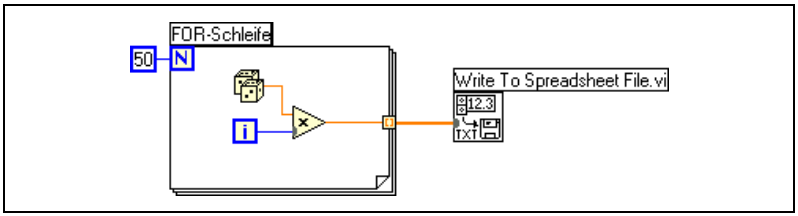


Abbildung 14-1. Schreiben in eine Tabelle mit Hilfe eines High-Level-VIs

Die Binärdatei-VIs werden zum Schreiben und Lesen von Binärdaten verwendet. Bei den Daten kann es sich um Ganzzahlen oder Fließkommazahlen einfacher Genauigkeit handeln.

Verwenden von Low-Level- und erweiterten Datei-I/O-VIs und -Funktionen

Wenn Sie jeden Datei-I/O-Vorgang einzeln steuern möchten, verwenden Sie dazu die Low-Level- sowie die erweiterten VIs und Funktionen zur Datei-I/O.

Mit den Low-Level-Hauptfunktionen können Sie eine Datei erstellen oder öffnen, Daten in die Datei schreiben beziehungsweise Daten aus der Datei lesen sowie die Datei schließen. Mit den anderen Low-Level-Funktionen können folgende Aufgaben durchgeführt werden:

- Erstellen von Verzeichnissen.
- Verschieben, Kopieren oder Löschen von Dateien.
- Auflisten von Verzeichnisinhalten.
- Ändern von Dateieigenschaften.
- Bearbeiten von Pfaden.

Bei einem Pfad (siehe Symbol links) handelt es sich um einen LabVIEW-Datentyp, der den Speicherort einer Datei auf dem Datenträger bezeichnet. Der Pfad beschreibt den Datenträger, der die Datei enthält, die Verzeichnisse zwischen der obersten Ebene des Dateisystems und der Datei sowie den Namen der Datei. Verwenden Sie zur Eingabe von Pfaden das Pfadbedienelement und zur Darstellung von Pfaden das -anzeigeelement. Achten Sie bei der Angabe auf die Syntax der jeweiligen Plattform. Weitere Einzelheiten zu Pfadbedien- und -anzeigeelementen finden Sie im Abschnitt *Pfad-Bedien- und Anzeigeelemente* des Kapitels 4, *Erstellen des Frontpanels*.

In Abbildung 14-2 sehen Sie ein Beispiel für die Anwendung von Low-Level-VIs und -Funktionen zur Übergabe von Zahlen an eine Excel-Datei. Wenn Sie dieses VI ausführen, öffnet das VI “Öffnen/Erstellen/Ersetzen einer Datei” zunächst die Datei `numbers.xls`. Anschließend schreibt die Funktion “Datei schreiben” die Zahlen in die Datei. Mit der Funktion “Datei schließen” wird die Datei wieder geschlossen. Wenn die Datei nicht geschlossen wird, bleibt sie im Arbeitsspeicher und kann von anderen Anwendungen oder Benutzern nicht geöffnet werden.

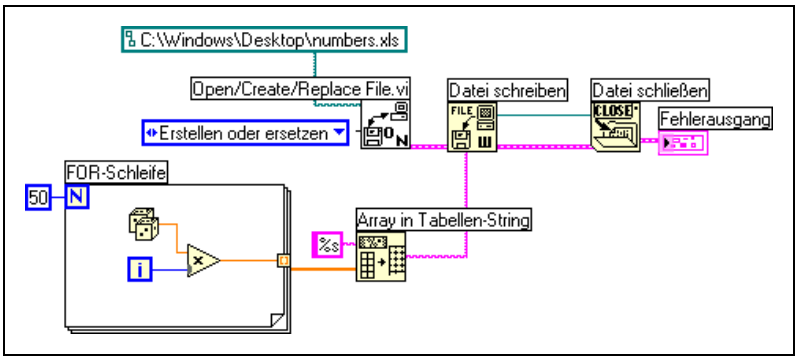


Abbildung 14-2. Anwendung von Low-Level-VIs zum Schreiben in eine Tabelle

Vergleichen Sie das VI in Abbildung 14-2 mit dem VI in Abbildung 14-1, das die gleiche Aufgabe durchführt. In Abbildung 14-2 ist das Array aus Zahlen mit der Funktion “Array in Tabellen-String” zunächst als String zu formatieren. Das VI “In Spreadsheet-Datei schreiben” in Abbildung 14-1 öffnet die Datei, konvertiert das Array aus Zahlen in einen String und schließt die Datei.

Ein Beispiel zur Verwendung von Low-Level-VIs und -Funktionen zur Datei-I/O finden Sie im VI “Write Datalog File Example” im Verzeichnis `examples\file\datalog.llb`.

Datenträger-Streaming

Um Speicherressourcen zu sparen, können Low-Level-VIs und -Funktionen zur Datei-I/O auch zum Datenträger-Streaming genutzt werden. Dabei handelt es sich um ein Verfahren, bei dem Dateien geöffnet bleiben, wenn mehrere Schreiboperationen (beispielsweise in einer Schleife) durchgeführt werden. Die einfach zu verwendenden High-Level-Schreiboperationen haben diesbezüglich den Nachteil, dass die Dateien bei jeder Ausführung geöffnet und geschlossen werden. VIs sind jedoch leistungsfähiger, wenn ein häufiges Öffnen und Schließen der Datei vermieden wird.

Durch das Datenträger-Streaming werden Systemressourcen gespart, da das System die Datei nicht für jeden Zugriff erneut öffnen und schließen muss. Fügen Sie für eine typische Datenträger-Streaming-Operation vor der Schleife das VI “Öffnen/Erstellen/Ersetzen einer Datei” und hinter der Schleife die Funktion “Datei schließen” ein. In der Schleife kann die Datei ohne die zusätzliche Systemverwaltungszeit für das Öffnen und Schließen der Datei kontinuierlich beschrieben werden.

Das Datenträger-Streaming eignet sich ideal für langwierige Datenerfassungsoperationen, bei denen die Geschwindigkeit von Bedeutung ist. Die Daten können bei laufender Datenerfassung kontinuierlich in eine Datei geschrieben werden. Für optimale Ergebnisse sollten bis zum Ende der Erfassung keine anderen VIs und Funktionen, wie zum Beispiel zur Analyse, ausgeführt werden.

Erstellen von Text- und Tabellenkalkulationsdateien

Damit Daten in eine Textdatei geschrieben werden können, müssen sie in Strings konvertiert werden. Um die Daten in eine Tabellenkalkulations-Datei zu schreiben, ist der String in einen Tabellen-String (einen String mit Trennzeichen, beispielsweise Tabulatorzeichen) zu formatieren. Weitere Informationen zum Formatieren von Strings finden Sie im Abschnitt *Formatieren von Strings* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Beim Schreiben von Text in Textdateien ist keine Formatierung notwendig, da eine solche für die meisten Textverarbeitungsprogramme nicht erforderlich ist. Mit dem VI “Zeichen in Datei schreiben” können Text-Strings in eine Textdatei geschrieben werden. Das VI öffnet und schließt die Datei automatisch.

Mit dem VI “In Spreadsheet-Datei schreiben” oder der Funktion “Array in Tabellen-String” können Zahlen aus einem Graphen, einem Diagramm oder Werte von einer Erfassung in einen Tabellen-String konvertiert werden. Weitere Informationen über die Verwendung dieser VIs und Funktionen erhalten Sie in den Abschnitten *Verwenden von High-Level-Datei-I/O-VIs* und *Verwenden von Low-Level- und erweiterten Datei-I/O-VIs und -Funktionen* dieses Kapitels.

Beim Lesen von Text eines Textverarbeitungsprogramms kann es zu Fehlern kommen, da der Text unter Umständen unterschiedliche Schriftarten, Farben, Formatierungen und Größen enthält, die von den High-Level-VIs zur Datei-I/O nicht verarbeitet werden können.

Zur Übergabe von Zahlen oder Text an Tabellenkalkulations- oder Textverarbeitungsprogramme sollten zum Formatieren und Zusammenfassen der Daten die String- und Array-Funktionen verwendet werden. Anschließend sind die Daten in eine Datei zu schreiben. Weitere Informationen zum Formatieren und Zusammenfassen der Daten mit Hilfe dieser Funktionen finden Sie in Kapitel 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Formatieren und Schreiben von Daten in Dateien

Mit der Funktion “In Datei formatieren” können String-, Pfad-, numerische und boolesche Daten als Text formatiert und dieser Text in eine Datei geschrieben werden. Bei Verwendung dieser Funktion muss der String nicht extra mit der Funktion “In String formatieren” formatiert und dann mit dem VI “Zeichen in Datei schreiben” oder der Funktion “Datei schreiben” übergeben werden.

Weitere Informationen zum Formatieren von Strings finden Sie im Abschnitt *Formatieren von Strings* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*.

Einlesen von Daten aus Dateien

Mit der Funktion “Aus Datei einlesen” kann Text in einer Datei nach String-, Zahlen-, Pfad- oder booleschen Werten abgesucht und anschließend in einen Datentyp konvertiert werden. In vielen Fällen können Sie diese Funktion verwenden, anstatt Daten mit der Funktion “Datei lesen” oder mit dem VI “Zeichen aus Datei lesen” zu lesen und den resultierenden String mit der Funktion “Aus String suchen” zu durchzusuchen.

Erstellen von Binärdateien

Zum Erstellen einer Binärdatei sind Zahlenwerte zu erfassen und in eine Datei zu schreiben. Verwenden Sie zum Speichern ein- und zweidimensionaler Arrays aus 16-Bit-Ganzzahlen oder Fließkommazahlen einfacher Genauigkeit in eine Datei die VIs “In I16-Datei schreiben” und “In SGL-Datei schreiben.” Mit den VIs “Aus I16-Datei lesen” und “Aus SGL-Datei lesen” können die erstellten Dateien gelesen werden.

Zum Schreiben von numerischen Werten verschiedener Datentypen, wie zum Beispiel von Fließkommazahlen oder vorzeichenlosen 32-Bit-Ganzzahlen, sollten die erweiterten oder die Low-Level-Dateifunktionen verwendet werden. Verwenden Sie beim Lesen der Datei die Funktion “Datei lesen” und geben Sie den Datentyp der Zahl an.

Beispiele zum Lesen und Schreiben von Fließkommazahlen aus einer bzw. in eine Binärdatei sind die VIs “Read Binary File” und “Write Binary File” unter `examples\file\smplfile.llb`.

Erstellen von Datenprotokolldateien

Zum Erstellen und Lesen von Datenprotokolldateien aktivieren Sie die Datenprotokollierung des Frontpanels oder verwenden Sie die erweiterten Dateifunktionen, um die Daten zu erfassen und in eine Datei zu schreiben. Weitere Informationen zur Erstellung von und zum Zugriff auf Datenprotokolldateien vom Frontpanel aus finden Sie im Abschnitt *Protokollieren von Frontpanel-Daten* dieses Kapitels.

Die Daten in einer Datenprotokolldatei müssen nicht formatiert werden. Beim Schreiben oder Lesen muss jedoch der Datentyp angegeben werden. Wenn beispielsweise Temperaturmesswerte mit Datum und Uhrzeit der Erfassung vorliegen und diese in eine Datenprotokolldatei geschrieben werden sollen, sind die Daten als Cluster aus einer Zahl und zwei Strings anzugeben. Ein Beispiel für das Schreiben von Daten in eine Datenprotokolldatei finden Sie im VI “Simple Temp Datalogger” im Verzeichnis `examples\file\datalog.llb`.

Beim Lesen einer Datei, die Temperaturmesswerte mit Datum und Uhrzeit enthält, ist anzugeben, dass ein Cluster aus einer Zahl und zwei Strings erfasst werden soll. Ein Beispiel für das Lesen einer Datenprotokolldatei ist das VI “Simple Temp Datalog Reader” unter `examples\file\datalog.llb`.

Schreiben von Signalverläufen in Dateien

Mit Hilfe der VIs “Signalverläufe in Datei schreiben” und “Signalverläufe in Spreadsheet-Datei exportieren” können Signalverläufe an Dateien übergeben werden. Signalverläufe können dabei in Tabellen-, Text- oder Datenprotokolldateien geschrieben werden.

Wenn der erstellte Signalverlauf nur in einem VI verwendet werden soll, speichern Sie ihn als Datenprotokolldatei (`.log`). Weitere Informationen zur Datenprotokollierung finden Sie in dem Abschnitt *Einsatz von Datenprotokolldateien* dieses Kapitels.

Mit dem VI in Abbildung 14-3 werden mehrere Signalverläufe erfasst, in einem Graphen angezeigt und in eine Datei im Tabellenformat von Microsoft Excel geschrieben.

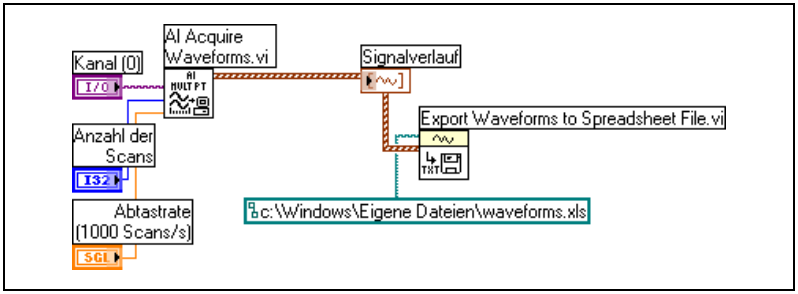


Abbildung 14-3. Schreiben mehrerer Signalverläufe in eine Tabellenkalkulationsdatei

Lesen von Signalverläufen aus Dateien

Mit Hilfe des VIs “Signalverlauf aus Datei lesen” können mehrere Signalverläufe aus einer Datei ausgelesen werden. Nach dem Lesen eines einzelnen Signalverlaufs können mit Hilfe der Funktion “Signalverlauf erstellen” Komponenten dieses Datentyps hinzugefügt bzw. bearbeitet werden oder mit der Funktion “Signalverlaufsattribut lesen” bestimmte Komponenten herausgefiltert werden.

Bei dem VI in Abbildung 14-4 wird ein Signalverlauf aus einer Datei gelesen, die **dt**-Komponente bearbeitet und der resultierende Signalverlauf in einem Graphen dargestellt.

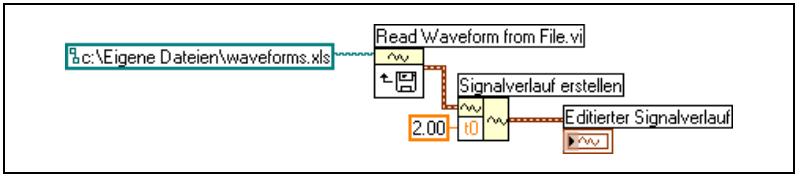


Abbildung 14-4. Lesen eines Signalverlaufs aus einer Datei

Mit dem VI “Signalverlauf aus Datei lesen” können auch mehrere Signalverläufe aus einer Datei ausgelesen werden. Das VI gibt ein Array aus Daten des Signalverlaufstyps zurück, die in einem Graphen mit mehreren Kurven dargestellt werden können. Wenn Sie auf einen einzelnen Signalverlauf einer Datei zugreifen möchten, müssen Sie das Array aus Signalverlaufdatentypen indizieren (siehe Abbildung 14-5). Weitere Informationen zum Indizieren von Arrays finden Sie im Abschnitt *Arrays* des Kapitels 10, *Gruppieren von Daten mit Hilfe von Strings, Arrays und Clustern*. Das VI greift auf eine Datei zu, die mehrere Signalverläufe umfasst. Mit der Funktion “Index-Array” werden der erste und der dritte Signalverlauf in der Datei ausgelesen und in zwei Kurvengraphen dargestellt. Weitere Informationen zu Graphen finden Sie im Kapitel 8, *Schleifen und Strukturen*.

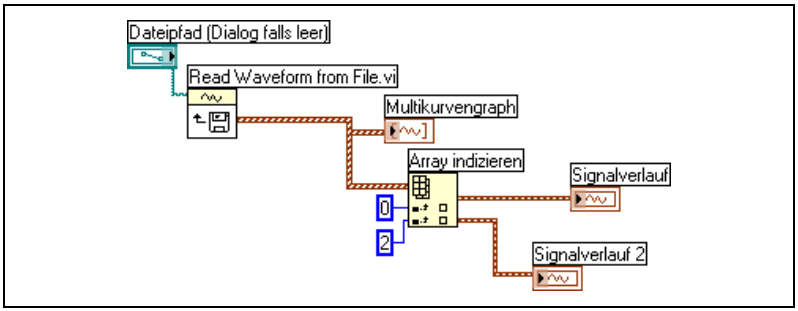


Abbildung 14-5. Lesen mehrerer Signalverläufe aus einer Datei

Durchgeschliffene Parameter

Viele VIs und Funktionen zur Datei-I/O enthalten durchgeschliffene Parameter (normalerweise eine RefNum oder einen Pfad) mit denen der vom entsprechenden Eingangsparameter eingelesene Wert ausgegeben wird. Sie können diese Parameter einsetzen, um die Ausführungsreihenfolge der Funktionen zu steuern. Durch Verbinden der Ausgabeseite des durchgeschliffenen Parameters des zuerst auszuführenden Knotens mit dem entsprechenden Eingang des nächsten Knotens kann eine künstliche Daten-

abhängigkeit geschaffen werden. Ansonsten müssten Sequenzstrukturen verwendet werden, um sicherzustellen, dass Datei-I/O-Operationen in der gewünschten Reihenfolge stattfinden. Weitere Informationen zur künstlichen Datenabhängigkeit finden Sie im Abschnitt *Datenabhängigkeit und künstliche Datenabhängigkeit* des Kapitels 5, *Erstellen des Blockdiagramms*.

Erstellen von Konfigurationsdateien

Mit den VIs für Konfigurationsdateien können Standardkonfigurationsdateien von Windows (`.ini`-Dateien) erstellt und gelesen sowie plattformspezifische Daten wie Pfade in einem plattformunabhängigen Format geschrieben werden, damit die von den jeweiligen VIs erstellten Dateien auf mehreren Plattformen einsetzbar sind. Bei diesen VIs wird jedoch für die Konfigurationsdateien kein Standarddateiformat verwendet. Ein Lesen und Schreiben der von den VIs erzeugten Dateien ist auf jeder Plattform möglich. Jedoch können keine Konfigurationsdateien im Mac- oder UNIX-Format erstellt bzw. verändert werden.

Beispiele zur Anwendung der Konfigurationsdatei-VIs finden Sie unter `examples\file\config.llb`.

Hinweis Die Standarderweiterung für Windows-Konfigurationsdateien lautet `.ini`. Die VIs arbeiten jedoch mit Dateien beliebiger Erweiterung zusammen, sofern der Inhalt im richtigen Format vorliegt. Weitere Informationen zum Konfigurieren des Inhalts finden Sie im Abschnitt *Windows-Dateiformat für Konfigurationseinstellungen* dieses Kapitels.

Einsatz von Konfigurationsdateien

In Windows handelt es sich bei einer Standarddatei mit Konfigurationseinstellungen um ein spezielles Format zum Speichern von Daten in einer Textdatei. Aufgrund dieses Formates kann auf einfache Weise programmatisch auf die Daten in der `.ini`-Datei zugegriffen werden.

So könnte eine Konfigurationsdatei beispielsweise folgenden Inhalt haben:

```
[Data]
Value=7.2
```

Diese Daten können mit den VIs für Konfigurationsdateien – wie in Abbildung 14-6 dargestellt – gelesen werden. Das vorliegende VI verwendet das VI “Schlüssel lesen”, um den **Schlüssel** mit dem Namen `Wert` aus dem **Abschnitt** mit dem Namen `Data` auszulesen. Es funktioniert unabhän-

gig von Änderungen der Datei, sofern das Windows-Dateiformat für Konfigurationseinstellungen beibehalten wird.

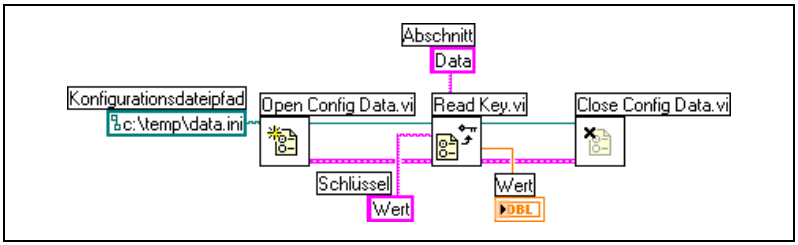


Abbildung 14-6. Lesen von Daten aus einer .ini-Datei

Windows-Dateiformat für Konfigurationseinstellungen

Bei Windows-Konfigurationsdateien handelt es sich um Textdateien, die in benannte Abschnitte unterteilt sind. Abschnittsnamen stehen in eckigen Klammern. Alle Abschnittsnamen in einer Datei müssen eindeutig sein. Die Abschnitte enthalten Schlüssel/Wert-Paare, die durch ein Gleichheitszeichen (=) getrennt sind. Innerhalb eines Abschnitts müssen alle Schlüsselnamen eindeutig sein. Der Schlüsselname stellt einen Konfigurationsparameter dar und der Wertname die Einstellung des jeweiligen Parameters. Das folgende Beispiel zeigt den Aufbau der Datei:

```
[Abschnitt 1]
Schlüssel1=Wert
Schlüssel2=Wert

[Abschnitt 2]
Schlüssel1=Wert
Schlüssel2=Wert
```

Verwenden Sie bei Konfigurationsdatei-VIs für den Wert des Parameters **Schlüssel** die folgenden Datentypen:

- String
- Pfad
- Boolesch
- 64-Bit-Fließkommazahl mit doppelter Genauigkeit
- 32-Bit-Ganzzahl mit Vorzeichen
- 32-Bit-Ganzzahl ohne Vorzeichen

Mit den Konfigurationsdatei-VIs können unformatierte oder konvertierte String-Daten gelesen und geschrieben werden. Die VIs lesen unformatierte Daten Byte für Byte, ohne die Daten in den ASCII-Code zu konvertieren.

Bei konvertierten Strings werden alle Textzeichen, die nicht angezeigt werden können, in der Konfigurationsdatei als entsprechende Hexadezimalcodes gespeichert, wie zum Beispiel `\0D` für einen Zeilenumbruch. Umgekehrte Schrägstriche werden bei diesem Dateityp immer verdoppelt, wie zum Beispiel `\\` für `\`. Setzen Sie die Eingänge **Originalstring lesen?** oder **Originalstring schreiben?** für unformatierte Daten auf TRUE und für konvertierte Daten auf FALSE.

Wenn VIs in eine Konfigurationsdatei schreiben, werden Strings oder Pfaddaten, die ein Leerzeichen enthalten, in Anführungszeichen eingeschlossen. Enthält ein String Anführungszeichen, speichert LabVIEW sie als `\"`. Bei Verwendung eines Texteditors stellen Sie eventuell fest, dass Anführungszeichen durch `\"` ersetzt werden.

Pfade werden in `.ini`-Dateien in einem plattformunabhängigen Format, dem UNIX-Standardformat für Pfade, gespeichert. Die VIs interpretieren den in einer Konfigurationsdatei gespeicherten, absoluten Pfad `/c/temp/data.dat` wie folgt:

- **(Windows)** `c:\temp\data.dat`
- **(Mac OS)** `c:temp:data.dat`
- **(UNIX)** `/c/temp/data.dat`

Ein relativer Pfad wie `temp/data.dat` wird wie folgt interpretiert:

- **(Windows)** `temp\data.dat`
- **(Mac OS)** `:temp:data.dat`
- **(UNIX)** `temp/data.dat`

Protokollieren von Frontpanel-Daten

Über die Frontpanel-Datenprotokollierung können Daten für die Verwendung in anderen VIs und Reports aufgezeichnet werden. So können beispielsweise Daten aus einem Graphen protokolliert und im Graphen eines anderen VIs dargestellt werden.

Die Frontpanel-Daten jeder VI-Ausführung werden in einer separaten Datei in einem Textformat mit Trennzeichen gespeichert. Zum Abrufen der Daten gehen Sie folgenderweise vor:

- Verwenden Sie dieselben VIs, aus dem die Daten aufgezeichnet wurden, um die Daten interaktiv abzurufen.
- Verwenden Sie VIs als SubVIs, um die Daten programmatisch abzurufen.
- Verwenden Sie die VIs und Funktionen zur Datei-I/O.

Jedes VI unterhält eine Protokolldateibindung, welche den Speicherort der Datenprotokolldatei aufzeichnet, in der LabVIEW die protokollierten Frontpanel-Daten verwaltet. Die Protokolldateibindung ist die Zuordnung zwischen einem VI und der Datenprotokolldatei, in der die VI-Daten protokolliert werden.

Eine Datenprotokolldatei enthält Datensätze mit einer Zeitmarkierung und den Daten der jeweiligen Ausführung des VIs. Beim Zugriff auf eine Datenprotokolldatei wählen Sie den gewünschten Datensatz aus, indem Sie das VI im Abrufmodus ausführen und die Frontpanel-Bedienelemente zum Anzeigen der Daten verwenden. Im Abrufmodus wird oben im Frontpanel ein numerisches Bedienelement angezeigt, mit dem Sie Datensätze auswählen können (siehe Abbildung 14-7).

Automatische und interaktive Frontpanel-Datenprotokollierung

Wählen Sie zum Aktivieren der automatischen Protokollierung die Option **Ausführen»Protokoll nach Beendigung**. Bei der ersten Aufzeichnung von Frontpanel-Daten für ein VI werden Sie aufgefordert, einen Namen für die Datenprotokolldatei anzugeben. LabVIEW protokolliert die Daten bei jeder Ausführung des VIs und hängt bei jeder weiteren Ausführung des VIs an die entsprechende Datei einen Datensatz an. Diese Datensätze können nicht überschrieben werden.

Um die Daten interaktiv zu speichern, wählen Sie **Ausführen»Datenprotokollierung»Protokoll**. Die Daten werden immer unverzüglich an die Datenprotokolldatei angehängt. Die interaktive Variante bietet den Vorteil, dass Sie auswählen können, wann die Daten protokolliert werden sollen. Bei automatischer Protokollierung hingegen werden die Daten bei jeder Ausführung des entsprechenden VIs aufgezeichnet.

Hinweis Bei Kurvendiagrammen wird bei der Frontpanel-Datenprotokollierung immer nur jeweils ein Datenpunkt aufgezeichnet. Wenn das Anzeigeelement für Diagramme mit einem Array verbunden wird, enthält das Datenprotokoll einen Teil des Arrays.

Interaktive Anzeige der protokollierten Frontpanel-Daten

Um die Daten nach dem Protokollieren interaktiv anzuzeigen, wählen Sie **Ausführen»Datenprotokollierung»Protokolldaten lesen**. Sie sehen dann die Datenabrufsymbolleiste (siehe Abbildung 14-7).

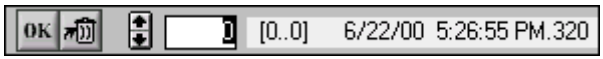


Abbildung 14-7. Datenabrufsymbolleiste

Die markierte Zahl kennzeichnet den angezeigten Datensatz. Die Zahlen in eckigen Klammern zeigen den Datensatzbereich an, der für das aktuelle VI protokolliert wurde. Eine Protokollierung findet bei jeder Ausführung des entsprechenden VIs statt. Datum und Uhrzeit der Aufzeichnung zeigen an, wann der jeweilige Datensatz aufgezeichnet wurde. Um zum nächsten oder vorherigen Datensatz zu gelangen, klicken Sie auf den entsprechenden Pfeil oder drücken Sie die Taste “Nach oben” bzw. “Nach unten”.

Je nach angezeigtem Datensatz ändert sich auch das Erscheinungsbild des Frontpanels. Wenn Sie beispielsweise zum nächsten Datensatz wechseln, indem Sie auf den entsprechenden Pfeil klicken, werden in den Bedien- und Anzeigeelementen die Daten für den betreffenden Datensatz zum Zeitpunkt der Protokollierung angezeigt. Um den Abrufmodus zu verlassen und zu dem VI zurückzukehren, dessen Datenprotokolldatei angezeigt wurde, klicken Sie auf die Schaltfläche **OK**.

Löschen eines Datensatzes

Im Abrufmodus können Sie bestimmte Datensätze entfernen. Markieren Sie dazu den betreffenden Datensatz und klicken auf die Schaltfläche **Datensatz löschen**. Durch erneutes Anklicken der Schaltfläche kann die Löschmarkierung wieder entfernt werden.

Um im Abrufmodus alle markierten Datensätze zu entfernen, wählen Sie **Ausführen»Datenprotokollierung»Daten löschen**.

Wenn Sie die Datensätze vor dem Betätigen der **OK**-Schaltfläche nicht entfernen, werden Sie gefragt, ob die markierten Datensätze entfernt werden sollen.

Löschen der Protokolldateibindung

Mit der Protokolldateibindung ordnen Sie einem VI die Datenprotokolldatei zu, die beim Aufzeichnen oder Abrufen von Frontpanel-Daten verwendet werden soll. Ein VI kann auch mehrere Dateien umfassen.

Dies ist eventuell beim Testen oder Vergleichen der VI-Daten hilfreich. So können beispielsweise die Daten, die bei der ersten Ausführung des VIs protokolliert wurden, mit den Daten verglichen werden, die bei der zweiten Ausführung vorlagen. Um einem VI mehr als eine Datenprotokolldatei zuzuordnen, müssen Sie die Protokolldateibindung löschen, indem Sie **Ausführen»Datenprotokollierung»Protokolldatei-Bindung löschen** wählen. Unabhängig davon, ob die Aufzeichnung automatisch oder interaktiv erfolgen soll, werden Sie bei der nächsten Ausführung des VIs aufgefordert, eine Datei auszuwählen.

Ändern der Protokolldateibindung

Um bei der Frontpanel-Aufzeichnung eine andere Protokolldatei zu verwenden oder Daten aus einer anderen Datei abzurufen, wählen Sie **Ausführen»Datenprotokollierung»Protokolldatei-Bindung ändern**. Daraufhin werden Sie aufgefordert, eine andere Protokolldatei auszuwählen oder eine neue zu erstellen. Sie können die Protokolldateibindung auch ändern, wenn Sie andere Daten in ein VI einlesen oder Daten aus dem VI an ein anderes Protokoll anhängen möchten.

Programmatischer Abruf von Frontpanel-Daten

Protokollierte Daten können auch mit einem SubVI beziehungsweise den VIs und Funktionen zur Datei-I/O abgerufen werden.

Abrufen von Frontpanel-Daten mit Hilfe eines SubVIs

Wenn Sie mit der rechten Maustaste auf ein SubVI klicken und aus dem Kontextmenü **Datenbankzugriff aktivieren** wählen, wird um das SubVI ein gelbes Kästchen angezeigt (siehe Abbildung 14-8).

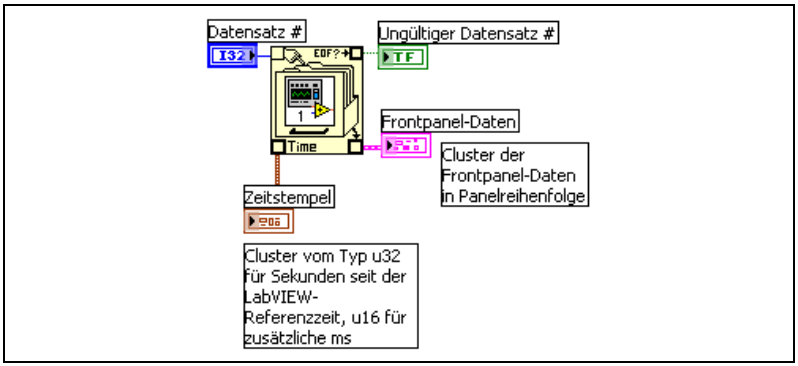


Abbildung 14-8. Abrufen protokollierter Daten

Das gelbe Kästchen, das einen Aktenschrank darstellt, enthält Anschlüsse für den Zugriff auf die Daten in der Datenprotokolldatei. Wenn Sie den Datenbankzugriff für das SubVI aktivieren, funktionieren die Ein- und Ausgänge des SubVIs als Ausgänge, die die protokollierten Daten zurückgeben. **Datensatz #** zeigt den abzurufenden Datensatz an, **ungültiger Datensatz #** zeigt an, ob eine Datensatznummer existiert, **Zeitstempel** ist die Zeit, zu der ein Datensatz erstellt wurde, und **Frontpaneldaten** ist ein Cluster der Frontpanel-Objekte. Um auf die Daten eines Frontpanel-Objekts zuzugreifen, verbinden Sie den Cluster **Frontpaneldaten** mit der Funktion “Aufschlüsseln”.

Sie können außerdem Werte für bestimmte Ein- und Ausgänge abrufen, indem Sie den entsprechenden Anschluss des SubVIs anschließen (siehe Abbildung 14-9).

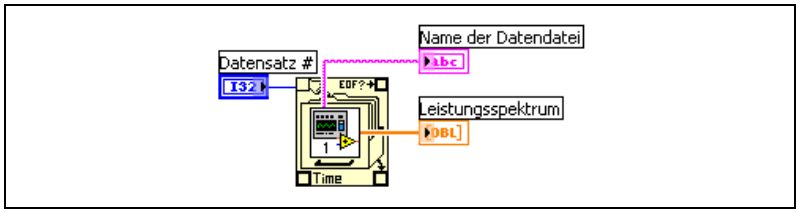


Abbildung 14-9. Abrufen protokollierter Daten über SubVI-Anschlüsse

Wenn Sie das VI ausführen, wird das SubVI nicht ausgeführt. Statt dessen werden die protokollierten Daten vom entsprechenden Frontpanel als Cluster an das VI-Frontpanel zurückgegeben.

Hinweis Wenn Sie ein SubVI oder Express-VI als erweiterbar konfigurieren, ist eine Aktivierung des Datenbankzugriffs für diesen Knoten nicht möglich.

Festlegen von Datensätzen

Wenn zu einem SubVI *n* protokollierte Datensätze vorliegen, können Sie eine beliebige Zahl von –*n* bis *n* – 1 mit dem Anschluss **Datensatz #** des SubVIs verbinden. Um auf Datensätze relativ zum ersten protokollierten Satz zuzugreifen, sind nichtnegative Zahlen zu verwenden. Die Zahl 0 steht für den ersten Datensatz, 1 für den zweiten, und so weiter bis zum letzten Datensatz, *n* – 1.

Es ist auch möglich, auf eine Datensatznummer relativ zum letzten protokollierten Datensatz zuzugreifen. In diesem Fall kann die Datensatznummer auch negativ sein. –1 steht dabei für den letzten Datensatz, –2 für den vorletzten, und so weiter bis –*n*, wodurch der erste Datensatz darge-

stellt wird. Wenn Sie eine Zahl außerhalb des Bereichs –*n* bis *n* – 1 mit dem Anschluss **Datensatz #** verbinden, wird der Ausgang **ungültiger Datensatz #** TRUE, und das SubVI ruft keine Daten ab.

Abrufen von Frontpanel-Daten mit Datei-I/O-Funktionen

Protokollierte Daten können auch mit den VIs und Funktionen zur Datei-I/O, wie zum Beispiel der Funktion “Datei lesen”, abgerufen werden. Der Datentyp jedes Datensatzes in der Datenprotokolldatei erstellt zwei Cluster: einen mit einer Zeitmarkierung und einen zweiten, in dem die Frontpanel-Daten enthalten sind. Der Cluster mit der Zeitmarkierung besteht aus einer vorzeichenlosen 32-Bit-Ganzzahl für die Sekunden und einer vorzeichenlosen 16-Bit-Ganzzahl für die Millisekunden, die seit der LabVIEW-Referenzzeit (1. Januar 1904, 00:00 Uhr Weltzeit) verstrichen sind.

Die Datei-I/O-Funktionen eignen sich sowohl für programmatisch erstellte als auch für Frontpanel-Datenprotokolldateien. Der Datensatztyp kann am Eingang **Typ des Protokolldatensatzes** der Funktion “Datei öffnen” festgelegt werden (siehe Abbildung 14-10).

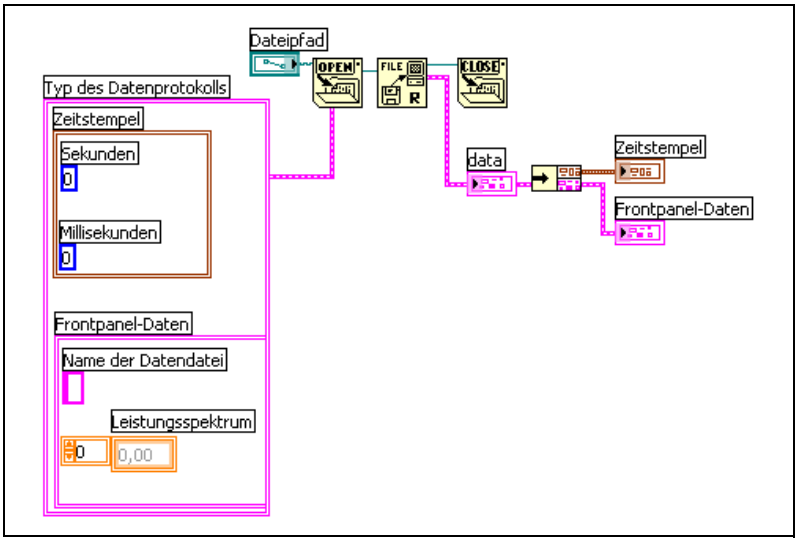


Abbildung 14-10. Abrufen protokollierter Daten mit der Funktion “Datei öffnen”

LabVIEW-Datenverzeichnis

Im Standardverzeichnis `LabVIEW Data` können Dateien gespeichert werden, die von LabVIEW erzeugt werden, wie beispielsweise `.lvm`- oder `.txt`-Dateien. Dieser Ordner wird im Standardverzeichnis Ihres Betriebssystems angelegt und soll Ihnen dabei helfen, von LabVIEW erzeugte Daten zu organisieren und einfach wiederzufinden. Die vom Express-VI "LabVIEW-Messdaten in Datei schreiben" erzeugten `.lvm`-Dateien werden standardmäßig in dieses Verzeichnis gespeichert. Das VI "LabVIEW-Messdaten aus Datei lesen" verwendet ebenfalls per Voreinstellung diesen Ordner. Die links abgebildete Konstante "Standard-Datenverzeichnis" und die gleichnamige Eigenschaft verweisen standardmäßig auf das Verzeichnis `LabVIEW Data`.

Um ein anderes Standardverzeichnis festzulegen, wählen Sie **Werkzeuge»Optionen** und klicken im oberen Pulldown-Menü auf **Pfade**. Das Standarddatenverzeichnis ist nicht mit dem Standardverzeichnis zu verwechseln, das zur Speicherung von neuen VIs, benutzerdefinierten Bedienelementen, VI-Vorlagen oder anderen von Ihnen erstellten LabVIEW-Komponenten dient.

LabVIEW-Dateien für Messdaten

In einer LabVIEW-Datei für Messdaten (`.lvm`-Datei) werden beispielsweise die vom Express-VI "LabVIEW-Messdaten in Datei schreiben" erzeugten Daten gespeichert. Dabei handelt es sich um eine mit Tabulatoren versehene Textdatei, die mit einem Tabellenkalkulationsprogramm oder einem Texteditor geöffnet werden kann. Abgesehen von den durch ein Express-VI erzeugten Daten, enthält die `.lvm`-Datei zusätzliche Angaben wie Erstellungsdatum und Zeit.

Weitere Hinweise zum Speichern und Abrufen von Daten mit Express-VIs entnehmen Sie bitte dem Handbuch *Erste Schritte mit LabVIEW*.

Eine LabVIEW-Datei für Messdaten kann auch mit NI-DIAdem verwendet werden.

15

Dokumentieren und Drucken von VIs

Mit Hilfe von LabVIEW können Sie VIs dokumentieren und drucken.

Im Abschnitt *Dokumentieren von VIs* dieses Kapitels wird die Aufzeichnung von Informationen zum Blockdiagramm und/oder Frontpanel in einem beliebigen Entwicklungsstadium des VIs in einer gedruckten VI-Dokumentation erörtert.

Optionen zum Drucken von VIs finden Sie im Abschnitt *Drucken von VIs* beschrieben. Einige Optionen eignen sich besser für das Drucken von Informationen über VIs, und andere für die Erstellung von Reports über die ausgegebenen Daten und Ergebnisse. Die verwendete Methode wird von mehreren Faktoren beeinflusst und richtet sich zum Beispiel danach, ob der Druckvorgang manuell oder programmatisch erfolgen soll, wie viele Optionen für das Berichtsformat benötigt werden, ob diese Funktion auch in der entsprechenden ausführbaren Anwendung benötigt wird oder auf welchen Plattformen die VIs verwendet werden sollen.

Weitere Informationen ...

Weitere Informationen zum Dokumentieren und Drucken von VIs und zur Reporterstellung finden Sie in der *LabVIEW-Hilfe*.

Dokumentieren von VIs

Mit LabVIEW können Sie die Entwicklung eines VIs verfolgen, ein fertiggestelltes VI dokumentieren und Anweisungen für Anwender erstellen. Die erstellte Dokumentation kann im HTML- oder RTF-Format gespeichert und ausgedruckt werden.

Um die Dokumentation möglichst aussagekräftig zu gestalten, fügen Sie Kommentare zur VI-Versionshistorie hinzu und erstellen Sie VI- und Objektbeschreibungen.

Erstellen der VI-Versionshistorie

Im Fenster **Historie** finden Sie die Versionsnummer und die Entwicklungsgeschichte eines VIs. Wenn Sie Veränderungen an einem VI vornehmen, sollten Sie diese im **Historie**-Fenster aufzeichnen und verfolgen. Wählen Sie **Werkzeuge»VI-Revisions-Historie**, um das Fenster **Historie** anzuzeigen. Die Versionshistorie eines VIs kann als HTML-, RTF- oder Textdatei gespeichert werden. Weitere Informationen zum Drucken der Versionshistorie finden Sie in Abschnitt *Drucken der Dokumentation* dieses Kapitels.

Versionsnummern

Die Versionsnummer bildet eine einfache Möglichkeit, Änderungen eines VIs zu verfolgen. Die Versionsnummer beginnt bei 0 und wird bei jeder Speicherung des VIs jeweils um eins erhöht. Um die aktuelle Versionsnummer in der Titelleiste des VIs und des **Historie**-Fensters anzuzeigen, wählen Sie **Werkzeuge»Optionen**, und dann **Revisions-Historie** aus dem Pulldown-Menü und aktivieren das Kontrollkästchen **Versionsnummer in der Titelleiste anzeigen**.

Im **Historie**-Fenster wird immer die Nummer der nächsten Version angezeigt, also die aktuelle Versionsnummer plus eins. Wenn Sie einen Kommentar einfügen, enthält die Kopfzeile des Kommentars die nächste Versionsnummer. Wenn an der Historie lediglich Änderungen vorgenommen wurden, ändert sich die Versionsnummer beim Speichern nicht.

Die Versionsnummern sind von den Kommentaren im Fenster **Historie** unabhängig. Lücken in den Versionsnummern zwischen Kommentaren zeigen an, dass das entsprechende VI ohne Eingabe von Kommentaren gespeichert wurde.

Da es sich bei der Historie genau genommen um ein Entwicklungswerkzeug handelt, entfernt LabVIEW die Historie automatisch, wenn das Blockdiagramm eines VIs gelöscht wird. Weitere Informationen zum Löschen des Blockdiagramms finden Sie in Abschnitt *Bereitstellen von VIs* des Kapitels 7, *Erstellen von VIs und SubVIs*. Das Fenster **Historie** kann nicht im Laufzeitmodus aufgerufen werden. Auf der Seite **Allgemein** des Dialogfelds **VI-Einstellungen** wird die Versionsnummer auch für VIs ohne Blockdiagramm angezeigt. Klicken Sie im Fenster **Historie** auf die Schaltfläche **Zurücksetzen**, um die Versionshistorie zu löschen und die Versionsnummer auf Null zurückzusetzen.

Erstellen von VI- und Objektbeschreibungen

Mit Hilfe der Beschreibungen für VIs und die dazugehörigen Objekte, wie zum Beispiel Bedien- und Anzeigeelemente können Sie deren Zweck erläutern und Anweisungen zu deren Verwendung geben. Die Beschreibungen können in LabVIEW angezeigt, ausgedruckt oder als HTML-, RTF- oder Textdateien gespeichert werden.

Um VI-Beschreibungen zu erstellen, zu bearbeiten und anzuzeigen, wählen Sie **Datei»VI-Einstellungen** und klicken Sie anschließend im Pull-down-Menü **Kategorie** die Option **Dokumentation** an. Objekt- und SubVI-Beschreibungen können auch erstellt, bearbeitet und angezeigt werden, indem Sie mit der rechten Maustaste auf das entsprechende Objekt klicken und aus dem Kontextmenü **Beschreibung und Tipp** auswählen. Hinweisstreifen sind kurze Beschreibungen, die angezeigt werden, wenn während der VI-Ausführung der Cursor über ein Objekt bewegt wird. Den Inhalt des Hinweisstreifens legen Sie im Dialogfenster **Beschreibung und Tipp** fest. Wenn Sie keinen Tipp eingeben, erscheint kein Hinweisstreifen. Wenn Sie den Cursor über das VI-Symbol bewegen, wird die Objekt- bzw. VI-Beschreibung im Fenster **Kontexthilfe** angezeigt.

Drucken der Dokumentation

Zum Drucken oder Speichern der VI-Dokumentation in einer HTML- oder RTF-Datei wählen Sie **Datei»Drucken**. Sie können entweder ein integriertes Format verwenden oder für die Dokumentation ein benutzerspezifisches Format erstellen. Folgende Elemente können in LabVIEW beschrieben werden:

- Symbol und Anschlussfeld
- Frontpanel und Blockdiagramm
- Bedien-, Anzeigeelemente und Datentypanschlüsse
- VIs und Objekte
- VI-Hierarchie
- Liste der SubVIs
- Versionshistorie

Hinweis Bei bestimmten VIs stehen Ihnen nicht alle dieser Möglichkeiten zur Verfügung. So hat zum Beispiel ein polymorphes VI weder ein Frontpanel noch ein Blockdiagramm, so dass zu diesen Elementen keine Dokumentation möglich ist.

Speichern im HTML-, RTF- oder Textformat

Die VI-Dokumentation kann im HTML- oder RTF-Format gespeichert werden. HTML- und RTF-Dateien können in die meisten Textverarbeitungsprogramme importiert und als Hilfedateien kompiliert werden. In LabVIEW erstellte HTML-Dateien können auch im Internet veröffentlicht werden. Weitere Informationen zum Erstellen von Hilfedateien mit HTML- und RTF-Dateien finden Sie in Abschnitt *Erstellen eigener Hilfedateien* dieses Kapitels. In Abschnitt *Programmatisches Drucken von VIs* dieses Kapitels erfahren Sie mehr darüber, wie Sie Ihre HTML-, RTF- und Textdateien programmatisch drucken können.

Wenn Sie Ihre Dokumentation als RTF-Datei speichern, können Sie festlegen, ob die Datei später als Online-Hilfedatei oder zur Textverarbeitung dienen soll. Im Hilfedateiformat werden Grafiken in externen Bitmap-Dateien gespeichert. Im Format für die Textverarbeitung werden Grafiken dagegen in das Dokument eingebettet. Bei HTML-Dateien werden alle Grafiken extern im JPEG-, PNG- oder GIF-Format gespeichert.

Auswahl des Grafikformats für HTML-Dateien

Wenn Sie die Dokumentation in einer HTML-Datei speichern, können Sie das Format der Grafikdateien und die Farbtiefe auswählen.

In JPEG wird gewöhnlich eine gute Komprimierung von Grafiken erzielt. Jedoch können beim Speichern einige Grafikdetails verloren gehen. Dieses Format eignet sich am besten für Fotos. Bei Strichzeichnungen, Frontpanels und Blockdiagrammen können durch die Komprimierung unter Umständen verschwommene Bilder oder Farbunterschiede auftreten. JPEG-Grafiken sind immer 24-Bit-Grafiken. Wenn Sie eine geringere Farbtiefe wie zum Beispiel schwarzweiß auswählen, werden Grafiken mit der erforderlichen Tiefe gespeichert, das Ergebnis ist aber trotzdem eine 24-Bit-Grafik.

Im PNG-Format werden Grafiken ebenfalls komprimiert, wenn auch nicht immer so gut wie in JPEG. Bei der Komprimierung in PNG gehen jedoch keine Details verloren. Daneben werden 1-Bit-, 4-Bit-, 8-Bit- und 24-Bit-Grafiken unterstützt. Bei einer geringeren Bit-Tiefe lässt sich die resultierende Grafik sehr viel besser komprimieren als das JPEG-Format. Das PNG-Format ersetzt das herkömmliche GIF-Format. Obwohl das PNG-Format eine Reihe von Vorteilen im Vergleich zu JPEG und GIF bietet, wird es jedoch von Webbrowsern nicht optimal unterstützt.

Mit dem GIF-Format kann ebenfalls eine gute Kompression erreicht werden. Das GIF-Format wird von den meisten Webbrowsern unterstützt. Aus lizenzrechtlichen Gründen speichert LabVIEW Grafiken zur Zeit nicht als GIF-Dateien; dies kann jedoch in Zukunft möglich sein. Mit Grafikkonvertern lassen sich unkomprimierte GIF-Dateien, wie sie von LabVIEW gespeichert werden, komprimieren. Um die Qualität der komprimierten GIF-Dateien zu verbessern, wählen Sie daher beim Speichern das PNG-Format und wandeln Sie die PNG-Datei in eine GIF-Datei um. Wenn die Grafik zunächst als PNG-Datei gespeichert wird, lässt sich auch eine höhere Qualität erzielen, da mit diesem Format ein exaktes Abbild der Originalgrafik möglich ist. Damit die Dateien mit der Endung `.gif` in die HTML-Datei eingebettet werden, muss diese entsprechend angepasst werden.

Bezeichnungskonventionen für Grafikdateien

Bei der Speicherung von Grafikdateien, die in HTML- oder RTF-Dokumente eingebettet werden sollen, werden die Anschlüsse der Bedien- und Anzeigeelemente mit einheitlichen Namen versehen. Wenn in einem VI mehrere Anschlüsse desselben Typs auftreten, wird nur eine entsprechende Grafikdatei erstellt. So würde LabVIEW beispielsweise bei einem VI mit drei Anschlüssen für 32-Bit-Integer mit Vorzeichen eine einzige `ci32.`*x*-Datei erstellen, wobei *x* für das entsprechende Grafikformat steht.

Erstellen eigener Hilfedateien

Mit den in LabVIEW erstellten HTML- oder RTF-Dateien können Sie auch eine eigene kompilierte Hilfedatei erstellen. **(Windows)** Die einzelnen HTML-Dateien können zu einer HTML-Hilfedatei kompiliert werden.

Die von LabVIEW erstellten Hilfedateien können zu einer **(Windows)** WinHelp-, **(Mac OS)** QuickHelp- oder **(UNIX)** HyperHelp-Datei kompiliert werden.

Um in HTML- oder kompilierten Hilfedateien Links zu erstellen, wählen Sie **Datei»VI-Eigenschaften** und rufen aus dem Menü **Kategorie** die Option **Dokumentation** auf.

Drucken von VIs

Zum Ausdrucken von VIs in LabVIEW gibt es folgende Möglichkeiten:

- Wählen Sie **Datei»Fenster drucken**, um den Inhalt des aktiven Fensters zu drucken.
- Wählen Sie **Datei»Drucken**, wenn umfassendere VI-Daten wie beispielsweise Daten zum Frontpanel, Blockdiagramm, zu SubVIs, Bedienelementen oder die VI-Historie ausgedruckt werden sollen. Weitere Informationen zum Verwenden dieser Methode zum Drucken von VIs finden Sie in Abschnitt *Drucken der Dokumentation* dieses Kapitels.
- Wählen Sie zum programmatischen Drucken den VI-Server. Auf diese Weise ist jederzeit ein Drucken jedes VI-Fensters und jeder VI-Dokumentation möglich. Weitere Informationen hierzu finden Sie im Kapitel 17, *Programmatische Steuerung von VIs*.

Drucken des aktiven Fensters

Die direkteste und zeitsparendste Methode, das aktive Frontpanel- oder Blockdiagrammfenster zu drucken, ist die Option **Datei»Fenster drucken**. Über diese Option wird der Arbeitsbereich des aktiven Fensters einschließlich aller Objekte gedruckt, die sich außerhalb des angezeigten Bereiches befinden. Titel-, Menü-, Symbol- oder Bildlaufleisten werden nicht dargestellt.

Wählen Sie **Datei»VI-Einstellungen** und klicken Sie im Pulldown-Menü **Kategorie** auf **Druckoptionen**. Hier können genauere Angaben zum programmatischen Drucken eines VIs oder zum Drucken über die Option **Datei»Fenster drucken** vorgenommen werden. Weitere Informationen zum programmatischen Drucken finden Sie in Abschnitt *Programmatisches Drucken von VIs* dieses Kapitels.

Programmatisches Drucken von VIs

Mit den folgenden Methoden können Sie VIs programmatisch drucken, anstatt die Dialogfelder zu verwenden, die bei der Auswahl der Option **Datei»Fenster drucken** und **Datei»Drucken** angezeigt werden:

- Festlegen, dass das Frontpanel eines VIs nach jeder Ausführung automatisch gedruckt wird.
- Erstellen eines SubVIs, um das VI zu drucken.

- Verwendung der VIs zur Reporterstellung, um Reports auszudrucken oder HTML-Reports mit Informationen über ein VI oder mit vom VI ausgegebenen Daten zu speichern.
- Verwendung der Druckmethoden der VI-Klasse, um ein VI-Fenster oder Dokumentation programmatisch zu drucken bzw. diese jederzeit als HTML-, RTF- oder Textdatei abzuspeichern. Weitere Informationen hierzu finden Sie im Kapitel 17, *Programmatische Steuerung von VIs*.

Hinweis Bei ausführbaren Anwendungen kann nur das Frontpanel ausgedruckt werden.

Drucken nach Ausführung

Zum Ausdrucken des Frontpanels eines VIs nach Ausführungsende wählen Sie **Ausführen»Nach Ausführung drucken**. Sie können auch **Datei»VI-Einstellungen** wählen, aus dem Pulldown-Menü **Kategorie** den Eintrag **Druck-Optionen** auswählen und dann das Kontrollkästchen **Panel automatisch drucken, wenn die Ausführung des VIs beendet ist** aktivieren.

Diese Vorgehensweise entspricht in etwa der Auswahl der Option **Datei»Fenster drucken**, wenn das Frontpanel das aktive Fenster ist.

Wenn das VI als SubVI verwendet wird, druckt LabVIEW, wenn die Ausführung des SubVIs beendet wird, bevor es zum aufrufenden VI zurückkehrt.

Ausdrucken von VI-Daten mit Hilfe eines SubVIs

In einigen Fällen soll ein VI eventuell nicht nach jeder Ausführung ausgedruckt werden. Der Druckvorgang soll möglicherweise nur dann erfolgen, wenn der Benutzer auf eine Schaltfläche klickt oder ein bestimmter Zustand eintritt, wie zum Beispiel ein Testfehler. Unter Umständen soll das Format des Ausdrucks besonders präzise angegeben werden oder es soll nur ein Teil der Bedienelemente gedruckt werden. In diesen Fällen können Sie ein SubVI verwenden, das einen Druckvorgang nach der Ausführung auslöst.

Erstellen Sie ein SubVI und formatieren Sie das Frontpanel so, wie es von LabVIEW gedruckt werden soll. Wählen Sie die Option **Ausführen»Nach Ausführung drucken** nicht im übergeordneten, sondern im SubVI aus. Wenn Sie drucken möchten, rufen Sie das SubVI auf und übergeben die zu druckenden Daten.

Erstellen und Drucken von Reports

Mit Hilfe der VIs zur Reporterstellung können Sie Reports ausdrucken oder HTML-Reports mit Informationen über ein VI oder mit vom VI ausgegebenen Daten speichern. Verwenden Sie das VI “Einfach VI-Panel oder -Dokumentationen drucken” um einen allgemeinen Report mit Angaben zu einem VI zu erstellen. Mit dem VI “Einfacher Text-Report” lassen sich allgemeine Reports mit den Daten erzeugen, die von einem VI ausgegeben werden. Für komplexere Reports verwenden Sie die anderen Reporterstellungs-VIs.

Hinweis Die Berichterzeugung ist nur mit dem LabVIEW Full und Professional Development System möglich.

Mit Hilfe der Berichterzeugungs-VIs können folgenden Aufgaben durchgeführt werden:

- Anhängen von Text, Grafiken oder Tabellen an einen Report.
- Festlegen von Textschriftart, -größe, -stil und -farbe.
- Festlegen der Ausrichtung des Reports: Hoch- oder Querformat.
- Festlegen von Kopf- und Fußzeilen.
- Festlegen von Rändern und Tabulatoren.

Weitere Vorgehensweisen zum Drucken

Erfüllen die Standarddruckmethoden von LabVIEW nicht Ihre Anforderungen, können Sie die folgenden zusätzlichen Verfahren verwenden:

- Zeilenweises Drucken von Daten. Wenn Sie einen Zeilendrucker besitzen, der an die serielle Schnittstelle angeschlossen ist, können Sie mit den Kompatibilitäts-VIs Text an den Drucker senden. Dazu sind in der Regel einige Kenntnisse über die Befehlssprache des Druckers erforderlich.
- Exportieren von Daten in andere Applikationen, wie zum Beispiel Microsoft Excel, Speichern der Daten in einer Datei und Drucken aus der anderen Applikation.
- **(Windows, Mac OS X und UNIX)** Verwenden Sie das VI “System Exec”.
- **(Mac OS)** Verwenden Sie das VI “AESend Print Document”.
- **(Windows)** Verwenden Sie die ActiveX-Elemente, um Daten von anderen Applikationen drucken zu lassen. Weitere Informationen zu ActiveX finden Sie in Kapitel 19, *Windows-Konnektivität*.

16

Anpassen von VIs

Die Funktionsweise von VIs und SubVIs kann jeweils den Anforderungen einer bestimmten Applikation angepasst werden. Wenn ein VI beispielsweise als SubVI verwendet werden soll, bei dem eine Eingabe des Benutzers erforderlich ist, muss es so konfiguriert werden, dass bei jedem Aufruf das dazugehörige Frontpanel angezeigt wird.

Es gibt verschiedene Vorgehensweisen, die Konfiguration eines VI zu verändern. So kann das VI selbst bearbeitet werden oder programmatisch unter Verwendung des VI-Servers. Für nähere Hinweise, wie das Verhalten von VIs mit Hilfe des VI-Servers konfiguriert werden kann, lesen Sie bitte Kapitel 17, *Programmatische Steuerung von VIs*.

Weitere Informationen ...

Weitere Informationen über das Anpassen von VIs finden Sie in der *LabVIEW-Hilfe*.

Verhalten und Erscheinungsbild von VIs konfigurieren

Wählen Sie **Datei»VI-Einstellungen**, um Erscheinungsbild und Verhalten eines VIs zu konfigurieren. Klicken Sie auf das Pulldown-Menü **Kategorie** im oberen Bereich des Dialogfelds, um zwischen verschiedenen Kategorien von Optionen auszuwählen, wie zum Beispiel:

- **Allgemein**—Zeigt den aktuellen Pfad an, unter dem ein VI gespeichert ist, die Versionsnummer, die Versionshistorie und alle seit dem letzten Speichern des VIs durchgeführten Änderungen. Hier kann auch das Symbol bearbeitet und die Größe des Rasters für das VI festgelegt werden.
- **Dokumentation**—Auf dieser Seite können Sie eine Beschreibung des VIs und eine Verknüpfung zu einem Thema in einer Hilfedatei erstellen. Weitere Informationen zu den Dokumentationsmöglichkeiten finden Sie im Abschnitt *Dokumentieren von VIs* des Kapitels 15, *Dokumentieren und Drucken von VIs*.
- **Sicherheit**—Auf dieser Seite können Sie ein VI sperren oder mit einem Kennwort schützen.

- **Fenstererscheinungsbild**—Ermöglicht die Konfiguration verschiedener Fenstereinstellungen.
- **Fenstergröße**—Hier kann die Fenstergröße festgelegt werden.
- **Ausführung**—Enthält verschiedene Optionen zur Ausführung eines VIs. Sie können ein VI beispielsweise so konfigurieren, dass es beim Öffnen sofort gestartet wird, oder anhält, wenn es als SubVI aufgerufen wird. Darüber hinaus können VIs unterschiedliche Prioritäten zugewiesen werden. Wenn ein VI beispielsweise ausgeführt werden soll, ohne auf den Abschluss einer anderen Operation zu warten, konfigurieren Sie das VI zur Ausführung mit zeitkritischer (höchster) Priorität. Weitere Informationen zum Erstellen von Multithread-VIs finden Sie in der Application Note *Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability*.
- **Bearbeitungsoptionen**—Hier kann die Gittergröße des aktuellen VIs eingestellt und die Darstellungsart der Bedien- und Anzeigelemente zu verändert werden, die LabVIEW erzeugt, wenn Sie einen Anschluss mit der rechten Maustaste anklicken und **Erstelle»Bedienelement** oder **Erstelle»Anzeigeelement** aus dem Kontextmenü wählen. Weitere Informationen zum Ausrichtungsgitter entnehmen Sie bitte dem Abschnitt *Ausrichten und Einteilen von Objekten* des Kapitels 4, *Erstellen des Frontpanels*.

Anpassen von Menüs

Sie können für jedes erstellte VI angepasste Menüs erstellen und die VIs so konfigurieren, dass Menüleisten ein- oder ausgeblendet werden. Zum Ein- und Ausblenden der Menüleisten wählen Sie unter **Datei»VI-Einstellungen** aus dem Menü **Kategorie** die Option **Fenstererscheinungsbild** aus, klicken auf die Schaltfläche **Anpassen** und aktivieren bzw. deaktivieren das Kontrollkästchen **Menüleiste anzeigen**.

Zum Konfigurieren von Menüs gehört das Erstellen des Menüs und das Bereitstellen des Blockdiagrammcodes, der jeweils auszuführen ist, wenn ein Menüpunkt ausgewählt wird.

Hinweis Die benutzerdefinierten Menüs werden erst bei der Ausführung eines VIs angezeigt.

Erstellen von Menüs

Zum Erstellen benutzerdefinierter Menüs oder zum Verändern der Standardmenüs von LabVIEW können Sie entweder das betreffende VI manuell bearbeiten oder programmatisch, wenn es ausgeführt wird. Wenn Sie **Bearbeiten»Laufzeitmenü** auswählen und im Dialogfeld **Menü-Editor** ein Menü erzeugen, erstellt LabVIEW eine Laufzeitmenüdatei (`.rtm`). Im betreffenden VI wird dann anstelle der Standardmenüleiste eine benutzerspezifische Menüleiste angezeigt. Der relative Pfad zwischen dem VI und der `.rtm`-Datei muss beibehalten werden.

Im Dialogfeld **Menü-Editor** kann einem VI eine benutzerspezifische `.rtm`-Datei zugeordnet werden. Bei Ausführung des VIs wird das Menü aus der `.rtm`-Datei geladen. Benutzerdefinierte Objekte können auch über das Dialogfeld **Menü-Editor** mit Hilfe von Applikationsobjekten – Objekten, die von LabVIEW im Standardmenü zur Verfügung gestellt werden – oder Benutzerobjekten, die man zusätzlich hinzufügen kann, erstellt werden. Das Verhalten der Applikationsobjekte ist von LabVIEW definiert; das der Benutzerobjekte können Sie bestimmen. Weitere Informationen zur programmatischen Bearbeitung von Menüs finden Sie in dem Abschnitt *Menüverarbeitung* dieses Kapitels.

Benutzerdefinierte Menüs können aber auch programmatisch über die Menüfunktionen erstellt werden. Anhand dieser Funktionen können Sie verschiedene Attribute der Benutzerobjekte einfügen, löschen oder verändern. Standardelemente können ebenfalls hinzugefügt oder entfernt werden. Ihre Funktion ist allerdings unveränderlich, da diese von LabVIEW festgelegt ist.

Menüverarbeitung

Beim Erstellen von benutzerspezifischen Menüs ist zur Unterscheidung zwischen Groß- und Kleinschreibung jedem Menüobjekt ein Tag, also ein eindeutiger Stringbezeichner, zuzuweisen. Wenn der Benutzer einen Menüpunkt auswählt, wird mit der Funktion “Menüauswahl lesen” der dazugehörige Tag programmatisch abgerufen. Auf Grundlage des Tag-Wertes wird im Blockdiagramm für jedes Menüelement ein Handler zur Verfügung gestellt. Der Handler ist eine Kombination aus While-Schleife und Case-Struktur, mit der festgestellt werden kann, ob eine Menüauswahl vorgenommen wurde, und wenn ja, welche, so dass der entsprechende Code ausgeführt werden.

Nach Erstellung eines benutzerdefinierten Menüs muss im Blockdiagramm eine Case-Struktur erstellt werden, in der für jeden Menüpunkt ein

bestimmter Cases ausgeführt wird. Dieser Vorgang wird Menüverarbeitung genannt. Bei Standardmenüpunkten erfolgt die Verarbeitung automatisch.

In Abbildung 16-1 wird der ausgewählte Menüpunkt von der Funktion “Menüauswahl lesen” erfasst und an die Case-Struktur übergeben, in der der entsprechende Case ausgeführt wird.

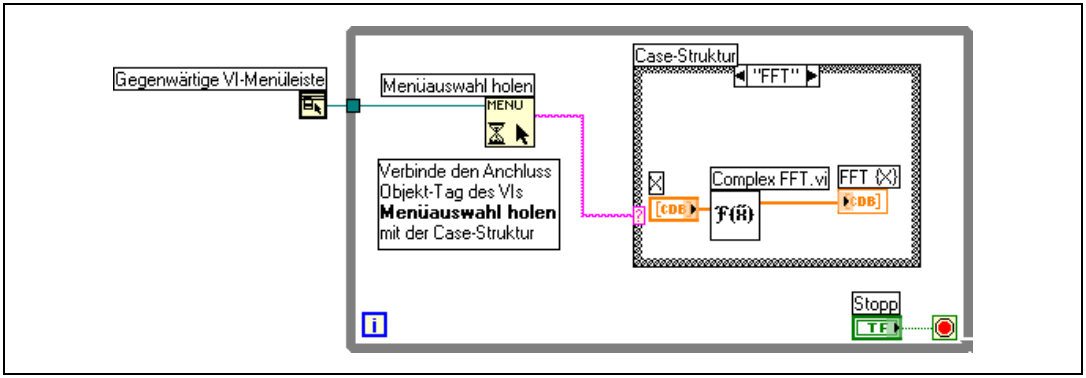


Abbildung 16-1. Blockdiagramm mit Menüverarbeitung

Wenn absehbar ist, dass die Verarbeitung eines bestimmten Menüelementes längere Zeit beansprucht, verbinden Sie den Eingang **Menü fixieren** der Funktion “Menüauswahl lesen” mit einem booleschen Bedienelement und setzen dieses auf TRUE. Dadurch kann der Benutzer im Menü keine andere Auswahl vornehmen, bis die Verarbeitung des Menüpunkts abgeschlossen ist. Um die Menüleiste anschließend wieder zu aktivieren, ist an die Funktion “Aktivieren der Menüüberwachung” mit einem TRUE-Wert zu verbinden.

Menüpunkte können aber auch mit Hilfe von Ereignisstrukturen verarbeitet werden. Für weitere Informationen zu Ereignisstrukturen lesen Sie bitte das Kapitel 9, *Ereignisgesteuerte Programmierung*.

17

Programmatische Steuerung von VIs

Mit Hilfe des VI-Servers, auf den über das Blockdiagramm, die ActiveX-Technologie oder das TCP-Protokoll zugegriffen werden kann, ist eine Kommunikation zwischen VIs und anderen Instanzen von LabVIEW möglich. Auf diese Weise können VIs und LabVIEW programmatisch gesteuert werden. VI-Server-Operationen können sowohl auf einem lokalen Computer als auch über ein Netzwerk ausgeführt werden.

Weitere Informationen ...

Weitere Informationen zur programmatischen Steuerung von VIs finden Sie in der *LabVIEW-Hilfe*.

Leistungsmerkmale des VI-Servers

Mit dem VI-Server können Sie folgende programmatische Operationen durchführen:

- Aufrufen eines VIs über das Netzwerk.
- Konfigurieren einer Instanz von LabVIEW als Server, um VIs zu exportieren, die von anderen Instanzen von LabVIEW im Netzwerk aufgerufen werden können. Wenn Sie beispielsweise an einem anderen Standort auf einem Netzwerkcomputer eine Applikation zur Datenerfassung betreiben, mit der Daten erfasst und protokolliert werden, können Sie diese Daten gelegentlich von Ihrem lokalen Computer aus abrufen. Durch Ändern der LabVIEW-Voreinstellungen können VIs im Netzwerk zugänglich gemacht werden. Dadurch gestaltet sich die Übertragung aktueller Daten so einfach wie ein SubVI-Aufruf. Alle notwendigen Operationen für die Übertragung über das Netzwerk werden vom VI-Server übernommen. Der VI-Server arbeitet sogar plattformübergreifend, das heißt, Client und Server können auf verschiedenen Plattformen ausgeführt werden.

- Bearbeiten der Eigenschaften von LabVIEW und VIs. Sie können beispielsweise die Position eines VI-Fensters dynamisch ermitteln oder einen Bildlauf im Frontpanel durchführen, um bestimmte Teile davon sichtbar zu machen. Darüber hinaus ist es möglich, Änderungen programmatisch zu speichern.
- Aktualisieren der Eigenschaften mehrerer VIs, anstatt für jedes VI manuell das Dialogfenster **Datei»VI-Einstellungen** zu verwenden.
- Abrufen von Informationen über eine Instanz von LabVIEW, wie zum Beispiel Versionsnummer und Ausgabe. Sie können auch Umgebungsinformationen abrufen, wie zum Beispiel die Plattform, auf der LabVIEW ausgeführt wird.
- Dynamisches Laden von VIs in den Speicher, wenn sie von einem anderen VI aufgerufen werden müssen, anstatt alle SubVIs beim Öffnen eines VIs zu laden.
- Erstellen einer Modularchitektur für die Applikation, so dass die Funktion der Applikation nach der Auslieferung an Kunden erweitert werden kann. Wenn Sie beispielsweise über mehrere VIs verfügen, die Daten filtern und alle dieselben Parameter verwenden, und Sie die Applikation so gestalten, dass diese VIs aus einem Modulverzeichnis dynamisch lädt, können Sie die Applikation mit einem Teil dieser VIs ausliefern und den Benutzern später zusätzliche Filter-VIs zur Verfügung stellen, indem Sie diese in das Modulverzeichnis speichern.

Erstellen von VI-Server-Applikationen

Das Programmiermodell für VI-Server-Applikationen basiert auf RefNums. Diese werden unter anderem auch bei der Datei-I/O, bei Netzwerkverbindungen und anderen Objekten in LabVIEW verwendet. Weitere Informationen zu RefNums finden Sie im Abschnitt *Referenzen auf Objekte oder Applikationen* des Kapitels 4, *Erstellen des Frontpanels*.

Normalerweise öffnen Sie eine RefNum für eine Instanz von LabVIEW oder ein VI. Die RefNum dient dann als Parameter für andere VIs zum Lesen oder Schreiben von Eigenschaften, Ausführen von Methoden oder zum dynamischen Laden des VIs, auf das über die RefNum verwiesen wird. Anschließend muss die RefNum geschlossen werden, um den Speicherplatz des entsprechenden VIs freizugeben.

Verwenden Sie zum Erstellen einer VI-Serverapplikationen folgende Funktionen und Knoten zur Applikationssteuerung:

- **Applikationsreferenz öffnen**—Erstellt eine RefNum auf eine lokale oder Netzwerkapplikation, auf die über den Server zugegriffen werden soll, oder für den Zugriff auf eine Netzwerkinstanz von LabVIEW.
- **VI-Referenz öffnen**—Erstellt eine Referenz auf ein VI auf dem lokalen oder dem Netzwerkcomputer oder zum dynamischen Laden eines VIs.
- **Eigenschaftsknoten**—Dient zum Festlegen oder Abfragen von VI-, Objekt- oder Applikationseigenschaften. Weitere Informationen zu Eigenschaften finden Sie im Abschnitt *Eigenschaftsknoten* dieses Kapitels.
- **Methodenknoten**—Ruft Methoden für ein VI, Objekt oder eine Applikation auf. Weitere Informationen zu Methoden finden Sie in dem Abschnitt *Methodenknoten* dieses Kapitels.
- **Aufruf über Referenz**—Ruft ein dynamisch geladenes VI auf.
- **LV-Objektreferenz schließen**—Schließt geöffnete RefNums zum VI, zum Objekt oder zur Applikation, auf die über den VI-Server zugegriffen wurde.

Applikations- und VI-Referenzen

Der Zugriff auf die verschiedenen Optionen des VI-Servers erfolgt über Referenzen auf zwei Hauptklassen von Objekten: das Applikationsobjekt und das VI-Objekt. Nachdem auf eines dieser Objekte eine Referenz erstellt wurde, kann sie an ein VI oder eine Funktion übergeben werden, das/die eine Operation für das Objekt durchführt.

Eine Applikations-RefNum verweist auf eine lokale oder eine Netzwerkinstanz von LabVIEW. Mit Hilfe von Applikationseigenschaften und -methoden können die LabVIEW-Voreinstellungen geändert und Systeminformationen angezeigt werden. Eine VI-RefNum verweist auf ein VI in einer Instanz von LabVIEW.

Mit einer RefNum auf eine Instanz von LabVIEW können Sie Informationen zur LabVIEW-Umgebung abrufen, wie zum Beispiel die Plattform, auf der LabVIEW ausgeführt wird, die Versionsnummer oder eine Liste aller VIs, die sich im Speicher befinden. Sie können auch Informationen, wie zum Beispiel den aktuellen Benutzernamen oder die Liste der in andere Instanzen von LabVIEW exportierten VIs festlegen.

Wenn Sie eine RefNum zu einem VI erstellen, wird das VI in den Arbeitsspeicher geladen. Dort verbleibt es, bis die RefNum geschlossen wird. Sind

gleichzeitig mehrere RefNums zu einem geöffneten VI vorhanden, bleibt das VI so lange im Arbeitsspeicher, bis alle RefNums geschlossen wurden. Über eine RefNum zu einem VI können alle Eigenschaften eines VIs, die im Dialogfeld **Datei»VI-Einstellungen** verfügbar sind, und dynamische Eigenschaften, wie zum Beispiel die Fensterposition des Frontpanels, aktualisiert werden. Daneben können Sie das VI auch programmatisch drucken, an einem anderen Ort speichern oder darin enthaltenen Text für Übersetzungszwecke in eine andere Sprache exportieren bzw. importieren.

Bearbeiten von Applikations- und VI-Einstellungen

Mit dem VI-Server können Sie Applikations- und VI-Einstellungen mit Hilfe der Eigenschafts- und Methodenknoten abrufen und festlegen. Viele Applikations- und VI-Einstellungen können nur über die Eigenschafts- und Methodenknoten abgerufen und festgelegt werden.

Beispiele zur Verwendung der Applikations- und VI-Klasseneigenschaften und -methoden finden Sie unter `examples\viserver`.

Eigenschaftsknoten

Mit dem Eigenschaftsknoten können Sie verschiedene Eigenschaften einer Applikation oder eines VIs abrufen oder festlegen. Um die Eigenschaften des Knotens auszuwählen, klicken Sie ihn mit Hilfe des Bedienwerkzeugs an oder klicken Sie mit der rechten Maustaste auf den weißen Bereich des Knotens und wählen Sie im Kontextmenü die Option **Eigenschaften**.

Grundsätzlich können mit einem Knoten auch mehrere Eigenschaften abgerufen oder festgelegt werden. Auf einige Eigenschaften besteht jedoch nur Lesezugriff. Um neue Anschlüsse hinzuzufügen, erweitern Sie den Eigenschaftsknoten mit Hilfe des Positionierwerkzeugs. Ein kleiner Pfeil rechts neben der Eigenschaft zeigt an, dass diese gelesen wird. Wenn sich der Pfeil links befindet, wird die Eigenschaft geschrieben. Um die Einstellung der Eigenschaft zu ändern, klicken Sie diese mit der rechten Maustaste an und wählen Sie im Kontextmenü die Option **In Lesen ändern** bzw. **In Schreiben ändern**.

Die Abarbeitung der Eigenschaften des Knotens erfolgt von oben nach unten. Wenn vor der Ausführung ein Fehler auftritt, wird der Knoten nicht ausgeführt. Überprüfen Sie daher immer die Möglichkeit von Fehlern. Tritt in einer Eigenschaft ein Fehler auf, ignoriert LabVIEW die restlichen

Eigenschaften und gibt eine Fehlermeldung aus. Der Cluster **Fehlerausgang** enthält Informationen darüber, welche Eigenschaft zu dem Fehler geführt hat.

Wenn der Eigenschaftsknoten eine Referenz auf eine Anwendung oder ein VI öffnet oder zurückgibt, kann diese mit der Funktion “Referenz schließen” wieder geschlossen werden. Wird eine Bedienelement-Referenz nicht mehr benötigt, wird sie automatisch geschlossen.

Automatisch verknüpfte Eigenschaftsknoten

Wenn Sie einen Eigenschaftsknoten aus einem Frontpanel-Objekt erstellen, indem Sie mit der rechten Maustaste auf das Objekt klicken und aus dem Kontextmenü **Erstelle»Eigenschaftsknoten** auswählen, wird der neue Eigenschaftsknoten automatisch mit dem Objekt verknüpft. Aus diesem Grund besitzen sie keinen **RefNum**-Eingang, und der Eigenschaftsknoten muss nicht mit dem Anschluss des Frontpanel-Objekts oder der Bedienelement-Referenz verbunden werden. Weitere Informationen zu den Bedienelemente-RefNums finden Sie im Abschnitt *Steuern von Frontpanel-Objekten* dieses Kapitels.

Methodenknoten

Mit einem Methodenknoten können Sie auf eine Applikation oder ein VI Aktionen oder Methoden anwenden. Im Gegensatz zum Eigenschaftsknoten führt ein einzelner Methodenknoten nur eine einzige Methode einer Applikation oder eines VIs aus. Um eine Methode auszuwählen, klicken Sie mit Hilfe des Bedienwerkzeugs auf den Methodenanschluss oder klikken Sie mit der rechten Maustaste auf den weißen Bereich des Knotens und wählen Sie aus dem Kontextmenü die Option **Methoden**.

Die Bezeichnung der Methode ist im Methodenknoten stets der erste Anschluss in der Parameterliste. Wenn die Methode einen Wert zurückgibt, wird dieser im Methodenanschluss angezeigt. Ansonsten besitzt der Methodenanschluss keinen Wert.

Die Parameter werden von oben nach unten aufgelistet, wobei der Name der Methode zuerst und die optionalen Parameter am Ende (grau) angezeigt werden.

Bearbeiten von Eigenschaften und Methoden der Applikationsklasse

Es können Eigenschaften in einer lokalen oder Netzwerkinstanz von LabVIEW abgerufen oder festgelegt und/oder Methoden in LabVIEW ausgeführt werden. In Abbildung 17-1 ist zu sehen, wie alle VIs im Arbeitsspeicher eines lokalen Computers in einem String-Array auf dem Frontpanel angezeigt werden können.

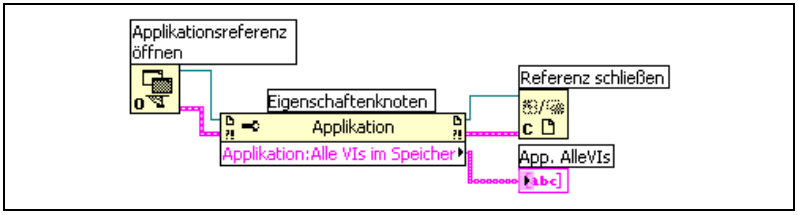


Abbildung 17-1. Anzeigen aller VIs im Speicher eines lokalen Computers

Wenn der **Referenz**-Eingang nicht verbunden ist, verwendet der Eigenschafts- oder Methodenknoten die aktuelle LabVIEW-Instanz als Referenz. Wenn die Eigenschaften oder Methoden einer anderen LabVIEW-Instanz verändert werden sollen, muss der **Referenz**-Eingang mit der entsprechenden RefNum verbunden werden.

Um die VIs im Speicher eines Netzwerkrechners zu finden, verbinden Sie ein String-Bedienelement mit dem Eingang **Rechnername** der Funktion “Applikationsreferenz öffnen” (siehe Abbildung 17-2) und geben die IP-Adresse oder den Domänennamen ein. Sie müssen auch die Eigenschaft **Exportierte VIs im Speicher** auswählen, da die Eigenschaft **Alle VIs im Speicher** der Abbildung 17-1 nur für lokale Instanzen von LabVIEW gilt.

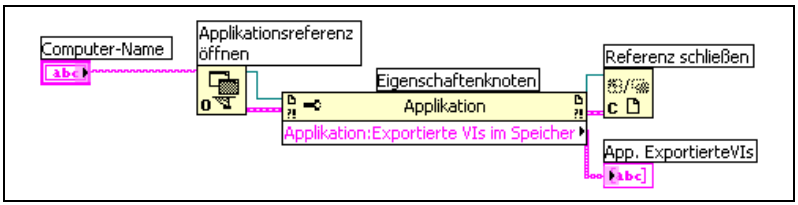


Abbildung 17-2. Anzeigen aller VIs im Speicher eines Netzwerkrechners

Bearbeiten von Eigenschaften und Methoden der VI-Klasse

Sie können Eigenschaften eines VIs abrufen oder festlegen und/oder Methoden eines VIs durchführen. In Abbildung 17-3 werden die Frontpanel-Objekte eines VIs mit Hilfe des Methodenknotens mit den Standardwerten reinitialisiert. Außerdem wird das Frontpanel wird geöffnet und die Werte werden angezeigt.

Wenn der **Referenz**-Eingang nicht verbunden ist, verwendet der Eigenschafts- oder Methodenknoten das VI als Referenz, in dem er sich befindet. Wenn die Eigenschaften oder Methoden eines anderen VIs verändert werden sollen, muss der **Referenz**-Eingang mit der entsprechenden RefNum verbunden werden.

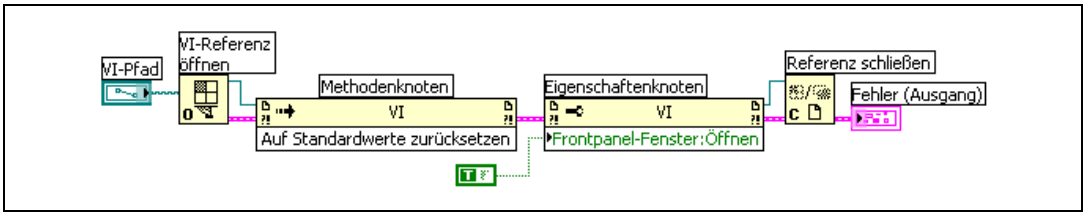


Abbildung 17-3. Verwenden von Eigenschafts- und Methodenknoten für VI-Klassen

Der Eigenschaftsknoten funktioniert ähnlich wie der Methodenknoten. Nachdem Sie eine VI-RefNum mit dem Eigenschaftsknoten verbunden haben, können Sie auf alle VI-Klasseneigenschaften zugreifen.

Bearbeiten von Eigenschaften und Methoden der Applikationsklasse und VI-Klasse

In manchen VIs müssen Sie auf Eigenschaften und Methoden der Applikationsklasse und der VI-Klasse zugreifen. Die RefNums zur Applikations- und VI-Klasse sind getrennt zu öffnen (siehe Abbildung 17-4).

In Abbildung 17-4 sehen Sie, wie die VIs im Arbeitsspeicher des lokalen Computers ermittelt werden und der Pfad jedes dieser VIs auf dem Frontpanel angezeigt wird. Um alle im Speicher befindlichen VIs zu finden, müssen Sie auf eine Applikationsklasseneigenschaft zugreifen. Für die Pfade verwenden Sie eine VI-Klasseneigenschaft. Die Anzahl der im Speicher befindlichen VIs bestimmt die Anzahl der Durchläufe der FOR-Schleife. Platzieren Sie die Funktion “VI-Referenz öffnen” und “Referenz schließen” in die FOR-Schleife, da für jedes im Arbeitsspeicher befindliche VI eine VI-RefNum benötigt wird. Die Applikations-RefNum darf erst geschlossen werden, nachdem die FOR-Schleife alle VI-Pfade abgerufen hat.

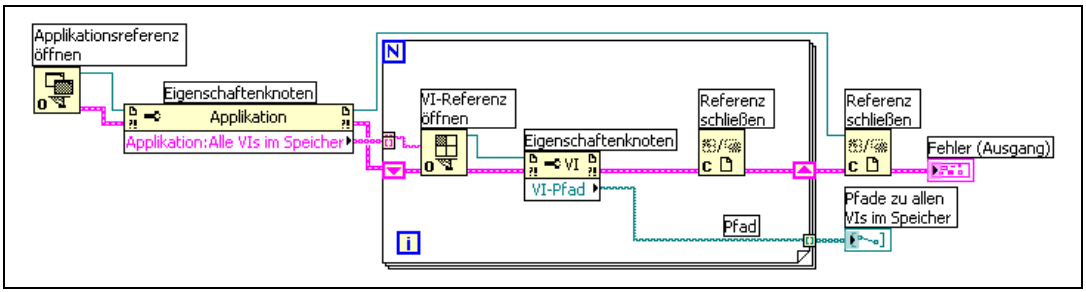


Abbildung 17-4. Verwenden von Eigenschaften und Methoden der Applikationsklasse und VI-Klasse

Dynamisches Laden und Aufrufen von VIs

Anstatt statisch verknüpfte SubVI-Aufrufe zu verwenden, können VIs auch dynamisch geladen werden. Bei einem statisch verknüpften SubVI handelt es sich um ein VI, das direkt in das Blockdiagramm eines aufrufenden VIs eingefügt wird. Es wird gleichzeitig mit dem aufrufenden VI geladen.

Im Gegensatz zu statisch verknüpften SubVIs werden dynamisch geladene SubVIs erst dann geladen, wenn sie vom übergeordneten VI aufgerufen werden. Bei einem größeren aufrufenden VI können Sie Ladezeit und Speicherplatz sparen, wenn das SubVI dynamisch geladen wird, da es immer nur bei Bedarf vom aufrufenden VI geladen wird. Nach Abschluss der Operation kann es aus dem Speicher entfernt werden.

Der Knoten “Aufruf über Referenz” und strikt typisierte VI-RefNums

Mit dem Knoten “Aufruf über Referenz” können VIs dynamisch aufgerufen werden.

Für den Knoten “Aufruf über Referenz” ist eine strikt typisierte VI-RefNum erforderlich. Diese identifiziert das Anschlussfeld des aufgerufenen VIs. Es wird keine permanente Zuordnung zu einem VI erstellt und es sind auch keine anderen Daten zu VIs enthalten, wie zum Beispiel Name oder Speicherort. Die Ein- und Ausgänge des Knotens “Aufruf über Referenz” können wie bei jedem anderen VI verbunden werden.

In Abbildung 17-5 sehen Sie, wie mit dem Knoten “Aufruf über Referenz” das VI “Frequenzantwort” dynamisch aufgerufen wird. Genau wie Eigenschafts- oder Methodenknoten muss der Knoten “Aufruf über Referenz” zusammen mit den Funktionen “VI-Referenz öffnen” und “Referenz schließen” verwendet werden.

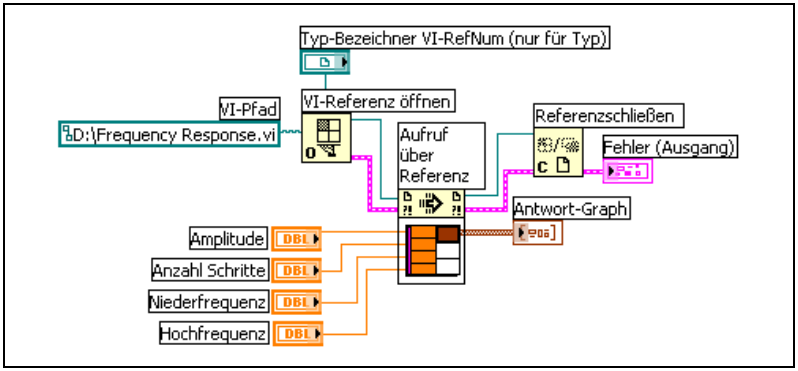


Abbildung 17-5. Verwendung des Knotens “Aufruf über Referenz”

Das für strikt typisierte RefNums angegebene VI stellt lediglich die Informationen zum Anschlussfeld bereit. Das bedeutet, dass keine ständige Verknüpfung zwischen der RefNum und dem VI hergestellt wird. Die Auswahl eines VI-Anschlussfeldes ist nicht mit dem Erhalt einer RefNum zum ausgewählten VI gleichzusetzen. Das VI wird über den Eingang **VI-Pfad** der Funktion “VI-Referenz öffnen” festgelegt.

Bearbeiten und Ausführen von VIs auf Netzwerkrechnern

Ein wichtiger Aspekt sowohl der Applikations- als auch der VI-RefNums ist ihre Netzwerktransparenz. Das heißt, das Erstellen von RefNums zu Objekten auf Netzwerkrechnern erfolgt auf dieselbe Art und Weise wie auf dem lokalen Computer.

Nachdem Sie eine RefNum zu einem Objekt im Netzwerk geöffnet haben, können Sie es mit wenigen Einschränkungen genau so behandeln wie ein lokales Objekt. Bei Operationen an einem Netzwerkobjekt sendet der VI-Server die entsprechenden Informationen über das Netzwerk und anschließend die Ergebnisse wieder zurück an den lokalen Rechner. Eine Netzwerkapplikation sieht nahezu genau so aus, als ob sie auf dem lokalen Rechner ausgeführt würde.

Steuern von Frontpanel-Objekten

Bedienelement-Referenzen in LabVIEW entsprechen den Objektreferenzen der Benutzeroberfläche in textbasierten Programmiersprachen. Sie sind nicht mit Zeigern zu gleichzusetzen.

Um Referenzen auf Frontpanel-Objekte an andere VIs zu übergeben, verwenden Sie die RefNum-Bedienelemente, die sich auf der Palette **RefNum** und **RefNum (klassisch)** befinden, oder klicken Sie mit der rechten Maustaste auf ein Frontpanel-Objekt und wählen Sie im Kontextmenü die Option **Erstelle»Referenz**. Sobald eine Objekt-RefNum an ein SubVI übergeben wurde, können mit den Eigenschafts- und Methodenknoten Eigenschaften abgefragt und geschrieben sowie Methoden des referenzierten Frontpanel-Objekts aufgerufen werden.

Informationen über das programmatische Steuern des Blockdiagramms über Frontpanel-Objeke mit Hilfe von Ereignissen finden Sie im Abschnitt *Case- und Sequenzstrukturen* des Kapitels 8, *Schleifen und Strukturen*.

Strikt und schwach typisierte Objekt-RefNums

Strikt typisierte Objekt-RefNums können nur mit Objekt-RefNums desselben Datentyps verwendet werden. Wenn zum Beispiel eine strikt typisierte Bedienelement-RefNum ein Schieberegler mit 32-Bit-Ganzzahlen ist, kann mit dem Anschluss der RefNum nur ein Schieberegler gleichen Typs verbunden werden. Es kann also weder ein Schieberegler mit 8-Bit-Ganzzahlen oder mit Skalaren doppelter Genauigkeit noch ein Cluster aus Schiebereglern mit 32-Bit-Ganzzahlen verwendet werden.

Objekt-Referenzen, die aus einem Bedienelement erstellt werden, sind standardmäßig strikt typisiert. Dies wird durch einen roten Stern in der linken unteren Ecke der Objekt-Referenz auf dem Frontpanel symbolisiert. Im Blockdiagramm wird im Eigenschafts- oder Methodenknoten, der an den Anschluss der Objekt-RefNum angeschlossen ist, der Hinweis `(strikt)` angezeigt.

Hinweis Da strikt typisierte Objekt-RefNums nicht mit booleschen Bedienelementen mit Latch kompatibel sind, werden bei solchen Elementen schwach typisierte Objekt-RefNums erzeugt.

Schwach typisierte Objekt-Referenzen sind hinsichtlich der akzeptierten Datentypen flexibler. Wenn der Typ einer schwach typisierten Objekt-Referenz ein Schieberegler ist, können Sie diese entweder mit einem Schieberegler mit 32-Bit-Ganzzahlen, mit einfacher Genauigkeit oder mit einem Cluster aus Schiebereglern mit 32-Bit-Ganzzahlen verbinden. Wenn der Typ einer schwach typisierten Objekt-Referenz “Bedienelement” ist, kann sie mit einer Objekt-RefNum eines beliebigen Bedienelementtyps verbunden werden.

Hinweis Wenn der Anschluss einer schwach typisierten Objekt-Referenz mit einem Eigenschaftsknoten verbunden wird, erzeugt die Eigenschaft “Wert” Variant-Daten, die eventuell konvertiert werden müssen, bevor sie verwendet werden können. Die Historiedaten-Eigenschaft ist nur dann verfügbar, wenn die Diagramm-RefNum strikt typisiert ist. Weitere Informationen zu Variant-Daten finden Sie in dem Abschnitt *Verarbeiten von Variant-Daten* des Kapitels 5, *Erstellen des Blockdiagramms*.

18

Arbeiten mit LabVIEW im Netzwerk

VIs können mit anderen Prozessen kommunizieren oder mit diesen vernetzt werden, auch mit solchen, die z. B. in anderen Applikationen oder auf Netzwerkrechnern ausgeführt werden. Mit den Netzwerkfunktionen in LabVIEW können folgende Aufgaben durchgeführt werden:

- Gemeinsame Nutzung von aktuellen Werten mit anderen VIs und Funktionen mit Hilfe der DataSocket-Technologie von National Instruments.
- Veröffentlichen von Frontpanel-Bildern und VI-Dokumentationen im Web.
- Versenden von VI-Daten per E-Mail.
- Erstellen von VIs, die mit anderen Applikationen und VIs über Low-Level-Protokolle wie TCP, UDP, Apple-Ereignisse oder PPC-Toolbox kommunizieren.

Weitere Informationen ...

Weitere Informationen zum Umgang mit LabVIEW im Netzwerk befinden sich in der *LabVIEW-Hilfe*.

Auswahl zwischen Datei-I/O, VI-Server, ActiveX und Arbeiten im Netzwerk

Das Arbeiten im Netzwerk ist unter Umständen nicht immer die beste Lösung für eine Applikation. Wenn eine Datei erstellt werden soll, welche Daten enthält, die andere VIs und Applikationen lesen können, verwenden Sie Datei-I/O-VIs und -Funktionen. Weitere Hinweise zur Verwendung von Datei-I/O-VIs und -Funktionen befinden sich in Kapitel 14, *Datei-I/O*.

Verwenden Sie zur Steuerung anderer VIs den VI-Server. Informationen zum Steuern von VIs und anderen LabVIEW-Applikationen auf lokalen Computern oder Netzwerkrechnern finden Sie in Kapitel 17, *Programmatische Steuerung von VIs*.

(Windows) Wenn Sie auf die Funktionen von Microsoft-Anwendungen zugreifen möchten, wie zum Beispiel das Einbetten eines Kurvengraphen in eine Excel-Tabelle, verwenden Sie ActiveX-VIs und -Funktionen. Hinweise zum Zugriff auf ActiveX-fähige Applikationen und Zulassen des Zugriffs von anderen ActiveX-Applikationen auf LabVIEW entnehmen Sie bitte Kapitel 19, *Windows-Konnektivität*.

LabVIEW als Netzwerk-Client und -Server

Sie können LabVIEW entweder als Client verwenden, um Daten zu empfangen und Funktionen anderer Applikationen zu verwenden, oder als Server, um bestimmte LabVIEW-Funktionen anderen Applikationen zur Verfügung zu stellen. Weitere Informationen zum Verwenden des VI-Servers, um VIs auf lokalen Computern oder Netzwerkrechnern zu steuern, finden Sie in Kapitel 17, *Programmatische Steuerung von VIs*. Sie steuern VIs, indem Sie mit dem Eigenschaftsknoten auf Eigenschaften zugreifen beziehungsweise mit dem Methodenknoten Methoden aufrufen.

Bevor auf die Eigenschaften und Methoden einer anderen Applikation zugegriffen werden kann, müssen Sie das verwendete Netzwerkprotokoll einrichten, mit dem der Zugriff auf die Eigenschaften und Methoden erfolgt. Sie können beispielsweise HTTP und TCP/IP als Protokolle verwenden. Das ausgewählte Protokoll hängt dabei von der jeweiligen Applikation ab. Das HTTP-Protokoll eignet sich beispielsweise ideal für die Veröffentlichung im Web. Es kann jedoch nicht verwendet werden, um ein VI zu erstellen, das von einem anderen VI erzeugte Daten empfängt. Verwenden Sie dazu das TCP-Protokoll.

Weitere Informationen zu den von LabVIEW unterstützten Kommunikationsprotokollen befinden sich in Abschnitt *Low-Level-Kommunikationsanwendungen* dieses Kapitels.

(Windows) Weitere Hinweise zum Verwenden der ActiveX-Technologie mit LabVIEW als ActiveX-Server bzw. -Client entnehmen Sie bitte Kapitel 19, *Windows-Konnektivität*.

Verwenden der DataSocket-Technologie

Mit der DataSocket-Technologie von National Instruments ist es möglich, anderen VIs im Netzwerk oder auf dem lokalen Computer Live-Daten zugänglich zu machen. Ähnlich wie bei einem Web-Browser, bei dem verschiedene Internetechnologien zur Anwendung kommen, werden bei DataSocket die gängigsten Kommunikationsprotokolle der Mess- und Automatisierungstechnik verwendet.

Die DataSocket-Technologie ermöglicht im Frontpanel über das Dialogfeld **DataSocket Verbindung** Zugriff auf mehrere Ein- und Ausgabemechanismen. Klicken Sie mit der rechten Maustaste auf ein Frontpanel-Objekt und wählen aus dem Kontextmenü **Datenoperationen»DataSocket-Verbindung**, um das Dialogfeld **DataSocket-Verbindung** anzuzeigen. Zum Senden und Empfangen der Daten ist ähnlich wie bei einem Web-Browser eine Adresse (URL) anzugeben.

Wenn Sie beispielsweise die Messwerte eines Thermometerelements auf dem Frontpanel für andere Computer im Internet freigeben möchten, geben Sie dazu in das Dialogfeld **DataSocket-Verbindung** eine Adresse ein. Um die Daten empfangen zu können, müssen Anwender an anderen Computern auf dem Frontpanel ein Thermometer ablegen und im Dialogfeld **DataSocket-Verbindung** die entsprechende Adresse auswählen. Weitere Hinweise zur Verwendung der DataSocket-Technologie im Frontpanel finden Sie im Abschnitt *Verwenden von DataSocket im Frontpanel* dieses Kapitels.

Für Einzelheiten zur DataSocket-Technologie lesen Sie die Ausarbeitung *Integrating the Internet into Your Measurement System*, die auch auf der Installations-CD im Verzeichnis `manuals` oder auf der Internetseite von National Instruments, `ni.com`, im PDF-Format verfügbar ist.

Angabe einer URL

Zur Datenübertragung von und zu URLs werden Kommunikationsprotokolle, wie zum Beispiel `dstp`, `ftp` oder `file`, verwendet. Das in einer URL verwendete Protokoll hängt von der Art der Daten ab, die anderen Anwendern zur Verfügung gestellt werden sollen, und von der Konfiguration des Netzwerks.

Beim Datenaustausch über DataSocket können folgende Protokolle verwendet werden:

- **DataSocket Transport Protocol** (`dstp`)—Systemspezifisches Protokoll für DataSocket-Verbindungen. Wenn Sie dieses Protokoll verwenden, kommuniziert das VI mit dem DataSocket-Server. Die Daten müssen dazu mit einem benannten Tag versehen werden, das an die URL angehängt wird. Dieses Tag dient zur Adressierung eines bestimmten Datenelements auf dem DataSocket-Server. Für dieses Protokoll ist ein aktiver DataSocket-Server erforderlich.
- **(Windows) OLE for Process Control** (`opc`)—Dient speziell zur gemeinsamen Nutzung von Echtzeit-Produktionsdaten, zum Beispiel von Daten industrieller Automatisierungsvorgänge. Um dieses Protokoll verwenden zu können, muss ein OPC-Server ausgeführt werden.
- **(Windows)** `logos`—Interne NI-Technologie zur Übertragung von Daten zwischen Netzwerk und lokalem Computer.
- **File Transfer Protocol** (`ftp`)—Bei diesem Protokoll kann eine Datei angegeben werden, aus der Daten ausgelesen werden sollen.

Hinweis Damit über DataSocket auf eine Textdatei auf einer FTP-Seite zugegriffen werden kann, ist an das Ende der DataSocket-URL `[text]` anzufügen.

- `file`—Mit diesem Protokoll kann eine Verbindung zu einer lokalen oder Netzwerkdatei mit Daten hergestellt werden.

Tabelle 18-1 zeigt Beispiele der einzelnen Protokoll-URLs.

Tabelle 18-1. Beispiele für DataSocket-URLs

URL	Beispiel
`dstp`	`dstp://servername.com/numeric`, wobei `numeric` für das benannte Tag steht.

`opc`	`opc:\National Instruments.OPCTest\objekt1`
	`opc:\\machine\National Instruments.OPCModbus\Modbus Demo Box.4:0`
	`opc:\\machine\National Instruments.OPCModbus\Modbus Demo Box.4:0?updaterate=100&deadband=0.7`

`logos`	`logos://computer_name/process/datenobjektname`

Tabelle 18-1. Beispiele für DataSocket-URLs (Fortsetzung)

URL	Beispiel
`ftp`	`ftp://ftp.ni.com/datasocket/ping.wav`

`file`	`file:ping.wav`
	`file:c:\meinedaten\ping.wav`
	`file:\\rechner\meinedaten\ping.wav`

Mit den `dstp`-, `opc`- und `logos`-URLs können aktuelle Daten anderen Anwendern verfügbar gemacht werden, da mit diesen Protokollen Bedien- und Anzeigeelemente (lokal und im Netzwerk) aktualisiert werden können. Mit `ftp`- und `file`-URLs sollten Daten aus Dateien gelesen werden, da über diese Protokolle keine Aktualisierung von Bedien- oder Anzeigeelementen möglich ist.

Beispiele zu DataSocket-Verbindungen befinden sich unter `examples\comm\datasktx.llb`.

Unterstützte DataSocket-Datenformate

Folgende Arten von Daten können über DataSocket anderen Anwendern zugänglich gemacht werden bzw. von anderen Anwendern genutzt werden:

- **Unformatierter Text**—Ist zu verwenden, um einen String an ein String-Anzeigeelement zu übermitteln.
- **Text mit Tabulatortrennzeichen**—Ist mit Text in einer Tabelle vergleichbar und sollte verwendet werden, um Daten in Arrays zu veröffentlichen. Text mit Tabulatortrennzeichen wird von LabVIEW als Daten-Array interpretiert.
- **Wave-Format**—Ist zu verwenden, um Sounds an andere VIs oder Funktionen zu übertragen.
- **Variant**—Dieser Datentyp sollte eingesetzt werden, wenn Daten aus einer anderen Applikation bezogen werden sollen, wie zum Beispiel einem ActiveX-Bedienelement von National Instruments Measurement Studio.

Verwenden von DataSocket im Frontpanel

Mit Frontpanel-DataSocket-Verbindungen können die aktuellen Werte eines Frontpanel-Objekts mit anderen Anwendern ausgetauscht werden. Wenn Daten eines Frontpanel-Objekts mit anderen Benutzern gemeinsam verwendet werden, bedeutet das eine Veröffentlichung dieser Daten.

Andere Benutzer können die veröffentlichten Daten abrufen und auf dem Frontpanel anzeigen lassen.

DataSocket-Verbindungen unterscheiden sich von Webserver- und ActiveX-Verbindungen dadurch, dass sie direkt vom Frontpanel aus hergestellt werden können, ohne dass dazu ein Eingriff in das Blockdiagramm erforderlich ist. Es kann zu jedem Bedien- oder Anzeigeelement eine eigene DataSocket-Verbindung erstellt werden. Beim Datenaustausch zwischen VIs über eine DataSocket-Verbindung werden keine Grafiken von Frontpanel-Bedienelementen übertragen, so dass die Daten in den VIs, die diese empfangen, individuell verarbeitet werden können.

Der Wert eines Bedienelements kann direkt im Frontpanel eingestellt und anschließend anderen Anwendern zugänglich gemacht werden. Es kann jedoch auch ein Blockdiagramm erstellt und der Ausgang eines VIs oder einer Funktion mit einem Anzeigeelement verbunden werden, von wo aus die Werte des Anzeigeelements veröffentlicht werden. Typische Anwendungsbereiche für DataSocket-Verbindungen zu Bedien- und Anzeigeelementen sind:

- Freigabe von Werten eines Frontpanel-Bedienelements für andere VIs, auf denen wiederum mit diesen Werten ein Bedienelement gesteuert werden kann oder die Werte in einem Anzeigeelement dargestellt werden können. Wenn sich beispielsweise auf einem Frontpanel ein Drehknopf zum Einstellen einer Temperatur befindet, können die entsprechenden Werte zur gemeinsamen Nutzung an andere VIs weitergegeben werden, wo sie in einem Anzeigeelement dargestellt oder über ein Bedienelement an ein SubVI oder eine Funktion übergeben werden können.
- Freigabe von Werten eines Anzeigeelements, so dass diese von anderen Benutzern in einem Bedien- oder Anzeigeelement auf deren Frontpanel dargestellt oder über ein Bedienelement an ein SubVI oder eine Funktion übergeben werden können. Bei einem VI, das die mittlere Temperatur berechnet und in einem Thermometer anzeigt, können zum Beispiel die Temperaturwerte an andere VIs übergeben werden.
- Um die Werte eines Bedien- oder Anzeigeelements in Ihr eigenes VI übernehmen zu können, müssen Sie sich dafür anmelden. Wenn ein Bedienelement verwendet wird, können Sie die Daten in einem VI mit dem Eingang eines SubVIs oder einer Funktion verbinden.
- Sobald Werte eines Frontpanel-Bedienelements freigegeben werden, kann dieses vom Frontpanel anderer Anwender aus verändert werden. Wenn Sie das VI ausführen, ruft das Frontpanel-Bedienelement in Ihrem VI den aktuellen Wert ab, der von einem anderen VI oder einer anderen Applikation über die DataSocket-Verbindung veröffentlicht

wurde. Ändert ein Benutzer den Wert des Bedienelements im eigenen Frontpanel, so wird der neue Wert über die DataSocket-Verbindung an das Frontpanel-Bedienelement Ihres VIs übertragen. Wenn Sie den Wert Ihres Frontpanel-Bedienelements ändern, werden die Frontpanel der anderen Benutzer entsprechend angepasst.

Objekte, die für den Datenempfang von anderen Objekten konfiguriert werden, müssen nicht mit diesen übereinstimmen. Die Frontpanel-Objekte müssen jedoch vom selben Datentyp sein, beziehungsweise bei numerischen Daten denselben Datentyp erzwingen. So können beispielsweise die Werte eines Thermometers, die aus einem anderen VI stammen, in Ihrem VI in einem digitalen Anzeigeelement dargestellt werden. Darüber hinaus kann das Thermometer Fließkommawerte und das digitale Anzeigeelement Ganzzahlen ausgeben.

Frontpanel-DataSocket-Verbindungen dienen hauptsächlich dazu, aktuelle Daten gemeinsam zu nutzen. Zum Auslesen von Dateien auf dem eigenen Rechner bzw. FTP- oder Webservern stehen Ihnen die Funktion “DataSocket: Lesen” sowie die VIs und Funktionen zur Datei-I/O und zur Applikationssteuerung zur Verfügung.

Lesen und Schreiben aktueller Daten über das Blockdiagramm

Mit Hilfe der DataSocket-Funktionen können Daten auch programmatisch vom Blockdiagramm aus in andere VIs übertragen oder aus anderen VIs übernommen werden.

Über die Funktion “DataSocket: Schreiben” können aktuelle Daten programmatisch über eine DataSocket-Verbindung an andere VIs übertragen werden. In Abbildung 18-1 sehen Sie, wie ein numerischer Wert geschrieben wird.

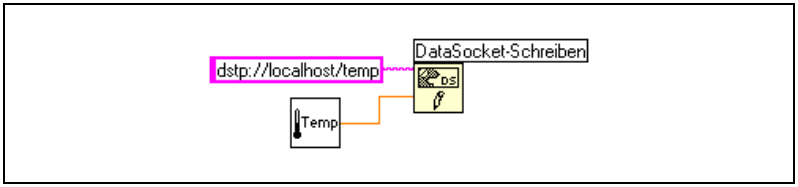


Abbildung 18-1. Veröffentlichen von Daten mit “DataSocket: Schreiben”

Die Funktion “DataSocket-Schreiben” ist polymorph, so dass der Eingang **Daten** mit den meisten Datentypen verbunden werden kann. Weitere Informationen zu polymorphen VIs und Funktionen finden Sie im Abschnitt *Polymorphe VIs und Funktionen* des Kapitels 5, *Erstellen des Blockdiagramms*.

Mit der Funktion “DataSocket: Lesen” können aktuelle Werte eines VIs programmatisch über eine DataSocket-Verbindung empfangen werden. In Abbildung 18-2 ist dargestellt, wie die Werte gelesen und Fließkommazahlen doppelter Genauigkeit konvertiert werden.

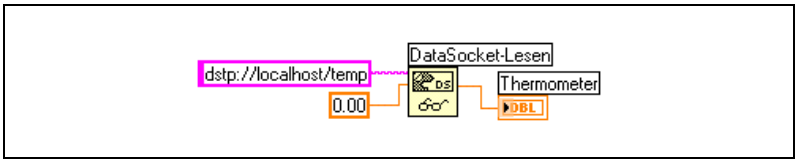


Abbildung 18-2. Lesen eines einzelnen Werts mit “DataSocket: Lesen”

Um aktuelle Daten in einen bestimmten Datentyp umzuwandeln, verbinden Sie ein Bedienelement oder eine Konstante mit dem Eingang **Typ** der Funktion “DataSocket: Lesen”. Ohne Angabe eines Typs werden vom **Daten**-Ausgang der Funktion Variant-Daten ausgegeben. Diese müssen mit der Funktion “Variant in Daten” in einen anderen Typ konvertiert werden.

Programmatisches Herstellen und Beenden von DataSocket-Verbindungen

Zum Herstellen und Beenden einer DataSocket-Verbindung dienen die Funktionen “DataSocket: Öffnen” und “DataSocket: Schließen”. Eine über die Funktion “DataSocket: Öffnen” hergestellte Verbindung bleibt so lange bestehen, bis sie entweder mit der Funktion “DataSocket: Schließen” beendet oder das VI geschlossen bzw. angehalten wird. Als Adresse am Eingang **URL** der Funktion “DataSocket: Öffnen” ist nur eine DataSocket-URL zulässig. Die Funktion gibt eine RefNum für die Verbindung aus, die an den **URL**-Eingang der Funktionen “DataSocket: Lesen” und “DataSocket: Schreiben” übergeben werden kann.

Puffern von DataSocket-Daten

Beim Einsatz des DataSocket Transport Protocol (`dstp`) werden standardmäßig immer nur die neuesten Werte an die Benutzer veröffentlicht, die die Daten beziehen. Wenn jedoch Daten schneller geschrieben werden, als sie vom anderen Client gelesen werden können, werden ältere, noch nicht ver-

arbeitete Daten durch die aktuelleren überschrieben, bevor sie ausgelesen werden können. Ein solcher Datenverlust kann sowohl beim Server als auch beim Client auftreten. Wenn der Datenempfänger immer nur die neuesten an den Server übertragenen Werte erhalten soll, stellt dieser Datenverlust normalerweise kein Problem dar. Wenn jedoch jeder Wert empfangen werden soll, der an den Server übertragen wurde, sollten die Daten beim Client gepuffert werden.

Hinweis Ein Client-seitiges Puffern ist auch bei anderen Protokollen wie `opc`, `logos` oder `file` möglich. Bei `dstp` muss über den DataSocket Server Manager auch der Server gepuffert werden. Weitere Hinweise zur Server-seitigen Pufferung befinden sich in der *DataSocket-Hilfe*.

Auch mit der `dstp`-Pufferung ist ein möglicher Datenverlust allerdings nicht auszuschließen. Wenn nämlich die in den Puffer des Servers oder Clients geschriebenen Daten die Kapazität des Puffers überschreiten, werden anstelle der aktuelleren die darin befindlichen alten Daten entfernt. Um zu erkennen, ob Daten in der Eingangsdatenfolge verloren gegangenen sind, verbinden Sie die veröffentlichten Daten mit der Funktion “Variant-Attribut setzen”. Damit kann im sendenden VI jeder Wert eindeutig identifiziert und anschließend beim empfangenden VI geprüft werden, ob IDs in der Eingangssequenz fehlen.

Setzen Sie den **Modus**-Eingang der Funktion “DataSocket: Öffnen” auf **Gepuffertes Empfangen** oder **Gepuffertes Empfangen/Schreiben** und legen Sie mit Hilfe eines Eigenschaftsknotens die Eigenschaften für die Größe des FIFO-Puffers fest. Dadurch wird sichergestellt, dass LabVIEW die Werte, die der Client empfängt, in einem Puffer speichert, so dass diese nicht bei jeder Wertänderung überschrieben werden.

Hinweis Wenn die Größe des FIFO-Puffers über die DataSocket-Eigenschaften festgelegt wird, ist der **Modus**-Eingang der Funktion “DataSocket: Öffnen” auf **Gepuffertes Empfangen** oder **Gepuffertes Empfangen/Schreiben** zu setzen. Ansonsten ist das Objekt auf dem Server nicht für die Verbindung gepuffert.

Abbildung 18-3 zeigt ein Beispiel zur DataSocket-Pufferung.

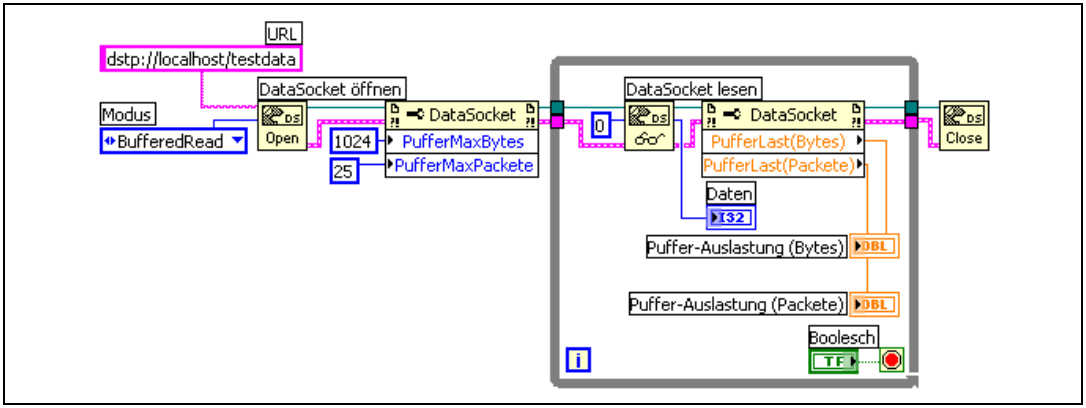


Abbildung 18-3. DataSocket-Pufferung

Hinweis Ein Puffern der Daten ist nur dann möglich, wenn mittels der Funktion “DataSocket: Lesen” der Client zum Empfang von vom Server zur Verfügung gestellten Daten registriert wird. Wenn die Registrierung zum Datenempfang über DataSocket-Verbindungen auf dem Frontpanel erfolgt, ist kein Puffern möglich.

Diagnoseinformationen

Über die Eigenschaft “Puffer-Auslastung (Bytes)” oder “Puffer-Auslastung (Pakete)” können Diagnoseinformationen zu den angegebenen Puffern abgefragt werden. So können Sie beispielsweise anzeigen lassen, wie viel Prozent der Pufferkapazität vom Client verwendet wird, um festzustellen, ob der Puffer ausreichend groß ist. Wenn einer der Werte zu nah an der maximalen Pufferkapazität liegt, sollte der Puffer vergrößert werden, so dass beim Empfang keine Daten verloren gehen.

Weitere Hinweise zum Festlegen der Puffergröße für einen DataSocket-Client finden Sie in der *LabVIEW-Hilfe*.

DataSocket und Variant-Daten

In einigen Fällen kann das VI oder die Applikation, mit dem bzw. der die Daten programmatisch empfangen werden, diese nicht in den ursprünglichen Datentyp zurückwandeln, zum Beispiel wenn die Daten aus einer anderen Applikation bezogen werden. Darüber hinaus müssen den Daten unter Umständen Attribute wie eine Zeitmarkierung oder Warnung hinzugefügt werden, was bei bestimmten Datentypen nicht möglich ist.

In diesen Fällen sollten die Daten, die über eine DataSocket-Verbindung gesendet werden, mit Hilfe der Funktion “Nach Variant” programmatisch in Variant-Daten umgewandelt werden. In Abbildung 18-4 ist ein Blockdiagramm dargestellt, mit dem kontinuierlich Temperaturmesswerte erfasst, die Daten in Variant-Daten konvertiert werden und als Attribut eine Zeitmarkierung hinzugefügt wird.

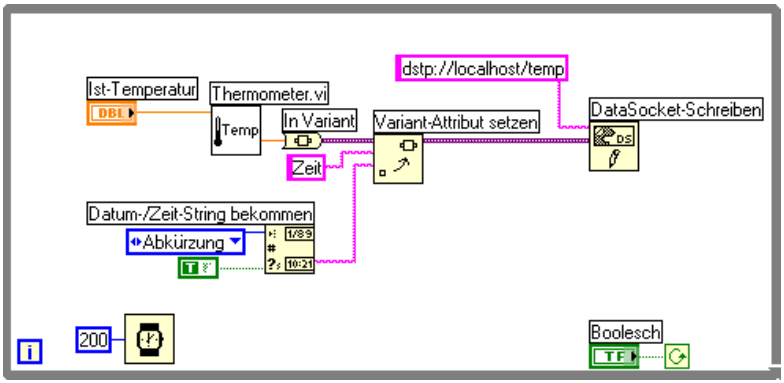


Abbildung 18-4. Umwandlung aktueller Temperaturwerte in Variant-Daten

Wenn die aktuellen Werte von einem anderen VI gelesen werden, muss das VI die Variant-Daten in einen Datentyp konvertieren, der von diesem VI verarbeitet werden kann. Über das Blockdiagramm in Abbildung 18-5 werden kontinuierlich Daten von einer DataSocket-Verbindung empfangen, diese Variant-Daten werden in Temperaturmesswerte konvertiert und das zu jedem Messwert gehörige Zeitmarkierungsattribut wird abgerufen. Temperatur und Zeitmarkierung werden auf dem Frontpanel angezeigt.

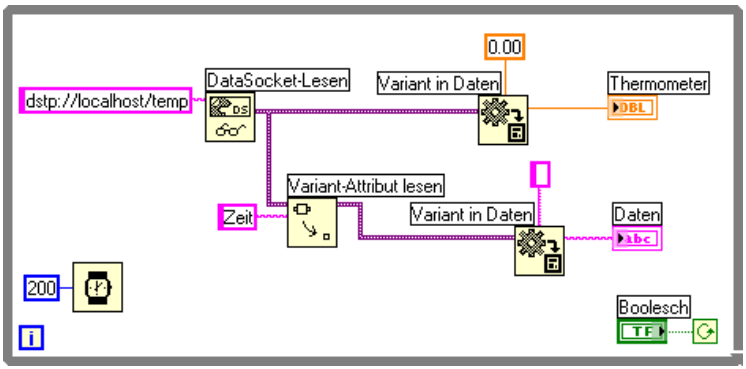


Abbildung 18-5. Umwandlung von Variant-Daten in Temperaturwerte

Veröffentlichen von VIs im Web

Mit dem LabVIEW-Webserver können HTML-Dokumente erstellt, Frontpanel im Internet abgebildet und VIs in Webseiten integriert werden. Außerdem besteht die Möglichkeit, den Browser-Zugriff auf die veröffentlichten Frontpanel zu steuern und zu konfigurieren, welche VIs im Web sichtbar sein sollen.

Hinweis Mit dem LabVIEW Enterprise Connectivity Toolset können VIs im Web gesteuert, und es können diesen VIs zusätzliche Sicherheitsmerkmale hinzugefügt werden. Nähere Informationen hierzu befinden sich auf der Internetseite von National Instruments, `ni.com`.

Webserver-Optionen

Wählen Sie **Werkzeuge»Optionen** und aus dem oberen Pulldown-Menü eines der **Webserver**-Elemente, um die folgenden Optionen einzustellen:

- Einrichten von Stammverzeichnis und Protokolldatei.
- Aktivieren des Webservers.
- Steuern des Browser-Zugriffs auf VI-Frontpanel.
- Konfigurieren, welche VI-Frontpanel im Web sichtbar sind.

Bevor VIs in das Web gestellt werden können, muss der Webserver unter **Webserver: Konfiguration**, im Menüpunkt **Optionen** aktiviert werden. Alternativ kann dazu auch das im Abschnitt *Erstellen von HTML-Dokumenten* dieses Kapitels beschriebene Web Publishing Tool verwendet werden. Die zu veröffentlichenden VIs müssen sich im Speicher befinden.

Erstellen von HTML-Dokumenten

Mit dem Web-Dokumentationswerkzeug unter **Werkzeuge»Web-Dokumentationswerkzeug** lassen sich folgende Arbeiten durchführen:

- Erstellen von HTML-Dokumenten.
- Einbetten von Frontpanelgrafiken oder -animationen in ein HTML-Dokument. Animationen können momentan jedoch nur in Netscape-Browsern dargestellt werden.
- Einbetten von VIs, die von Clients über das Netzwerk angezeigt und gesteuert werden können.
- Einfügen von Text über und unter der eingebetteten VI-Frontpanel-Grafik.
- Erstellen einer Umrandung für das eingefügte Bild oder VI.

- Verwendung der Seitenvorschau.
- Speichern von Dokumenten auf Festplatte.
- Aktivierung des Webservers zum Laden von HTML-Dokumenten und Frontpanel-Grafiken ins Netz.

Veröffentlichen von Frontpanel-Grafiken

Mit einer `.snap`-URL kann in einem Web-Browser oder einem HTML-Dokument ein statisches Abbild des Frontpanels eines VIs erstellt werden, das sich zu einem bestimmten Zeitpunkt im Speicher befindet. Die Abfrageparameter in der URL geben VI-Name und die Attribute des Bilds an. Verwenden Sie zum Beispiel `http://`*`web.server.adresse/`*`.snap?`*`VI-Name.vi`*, wobei *`VI-Name.vi`* für den Namen des anzuzeigenden VIs steht.

Mit einer `.monitor`-URL kann ein animiertes Abbild des Frontpanels eines im Speicher befindlichen VIs ausgegeben werden. In den Abfrageparametern in der URL sind VI-Name, Attribute der Animation und die Attribute des Bildes angegeben. Verwenden Sie zum Beispiel `http://`*`web.server.adresse/`*`.monitor?`*`VI-Name.vi`*, wobei *`VI-Name.vi`* für den Namen des anzuzeigenden VIs steht.

Formate der Frontpanel-Grafiken

Im Webserver können Frontpanel-Bilder im JPEG- und PNG-Format erzeugt werden.

Im JPEG-Format wird gewöhnlich eine gute Komprimierung von Grafiken erzielt. Jedoch können dadurch einige Grafikdetails verloren gehen. Dieses Format eignet sich am besten für Fotos. Bei Strichzeichnungen, Frontpanels und Blockdiagrammen können durch die Komprimierung unter Umständen verschwommene Bilder oder Farbunterschiede auftreten. Im PNG-Format werden Grafiken ebenfalls komprimiert, wenn auch nicht immer so gut wie in JPEG. Bei der Komprimierung in PNG gehen jedoch keine Details verloren.

Anzeigen und Steuern des Frontpanels über das Netzwerk

Um ein Frontpanel im Netz anzuzeigen und es von LabVIEW aus bzw. über einen Web-Browser zu steuern, muss eine Verbindung zum LabVIEW-eigenen Webserver hergestellt werden. Wenn Sie ein Frontpanel von einem Client aus im Netz öffnen, wird vom Webserver nur das

Frontpanel zum Client gesendet; das Blockdiagramm und die SubVIs verbleiben auf dem Server-Rechner. Das Frontpanel kann nun genau so bedient werden, als wenn das VI auf dem Client-Rechner laufen würde, nur mit dem Unterschied, dass das Blockdiagramm in diesem Fall auf dem Server-Rechner ausgeführt wird. Mit dieser Funktion lassen sich Frontpanels oder netzwerkgesteuerte Applikationen sicher, schnell und einfach veröffentlichen.

Hinweis Um ein ganzes VI über das Netzwerk zu steuern, sollte der Webserver verwendet werden. Für den Datenaustausch zwischen einzelnen Frontpanel-Bedienelementen empfiehlt sich dagegen der Einsatz des DataSocket-Servers. Weitere Informationen zum DataSocket-Server finden Sie im Abschnitt *Verwenden der DataSocket-Technologie* dieses Kapitels.

Konfiguration des Clients für den Server

Damit Frontpanel auf Client-Rechnern angezeigt und dort über LabVIEW oder einen Browser gesteuert werden können, muss auf dem Server-Rechner zunächst der Server konfiguriert werden. Klicken Sie dazu auf **Werkzeuge»Optionen** und wählen Sie aus dem Pulldown-Menü die Option **Webserver** aus. Auf diesen Seiten können die Zugriffsrechte auf den Server bestimmt werden, und es kann festgelegt werden, welche Frontpanel im Netzwerk sichtbar sein sollen. Außerdem lässt sich hier eine Zeitbegrenzung für die Nutzung des VIs einstellen, für den Fall, dass andere Client-Rechner darauf warten, auf das VI zuzugreifen.

Der Webserver gestattet, dass mehrere Clients gleichzeitig auf das Frontpanel zugreifen können, aber immer nur ein Client das Frontpanel steuern kann. Der Benutzer des Server-Rechners kann jederzeit die Benutzerrechte für alle VIs zurückverlangen. Wenn dann ein Frontpanel-Wert geändert wird, dann wird diese Änderung an alle Clients weitergegeben. Es sind jedoch nicht alle Änderungen auf den Client-Frontpanels zu sehen. So werden im Allgemeinen keine Änderungen des Server-Frontpanels angezeigt, sondern nur die aktuellen Werte. Wenn zum Beispiel auf dem Rechner, mit dem das VI gesteuert wird, der Abbildungsmodus eines Diagramms geändert oder eine Bildlaufleiste hinzufügt bzw. entfernt wird, so sind diese Änderungen nicht auf den Client-Rechnern sichtbar.

Lizenzen für netzwerkgesteuerte Frontpanel

Wenn mehrere Clients Zugriff auf den Server haben sollen, muss zunächst eine Lizenz mit der Anzahl der Clients erstellt werden.

Hinweis **(Windows 9*x*)** Eine Lizenz für den Zugriff eines Clients auf ein netzwerkgesteuertes Frontpanel ist im LabVIEW Full Development System und im Application Builder enthalten. Im LabVIEW Professional Development System wird eine entsprechende Lizenz für fünf Client-Rechner mitgeliefert. Unter Windows 9*x* ist keine Erweiterung der Lizenz möglich. **(Windows 2000/NT/XP, Mac OS und UNIX)** Im LabVIEW Full Development System und im Application Builder ist jeweils eine Lizenz für den Zugriff eines Clients auf ein netzwerkgesteuertes Frontpanel enthalten. Im LabVIEW Professional Development System wird eine entsprechende Lizenz für fünf Client-Rechner mitgeliefert. Diese Lizenzen sind auf mehrere Benutzer erweiterbar.

Darstellung und Steuerung von Frontpanels über LabVIEW oder einen Web-Browser

Ein Frontpanel kann auf einem Client-Rechner angezeigt und entweder über LabVIEW oder einen Web-Browser gesteuert werden.

Hinweis Dazu muss auf dem Rechner, auf dem sich das entsprechende VI befindet, der Webserver aktiviert werden.

Anzeige und Steuerung von Frontpanels über LabVIEW

Um ein netzwerkgesteuertes Frontpanel mit LabVIEW als Client anzuzeigen, öffnen Sie ein neues VI und wählen Sie **Ausführen»Mit Netzwerk-Panel verbinden**. Sie sehen dann das Dialogfeld **Verbinden mit Netzwerk-Panel**. Hier können die Internetadresse des Servers und das anzuzeigende VI festgelegt werden. Per Voreinstellung kann das Netzwerk-Panel zunächst nur betrachtet werden. Um das VI steuern zu können, setzen Sie in das Kontrollkästchen **Steuerung für VI beantragen** im Dialogfeld **Verbinden mit Netzwerk-Panel** ein Häkchen. Wenn das VI auf Ihrem Rechner erscheint, klicken Sie das Frontpanel mit der rechten Maustaste an und wählen aus dem Kontextmenü die Option **VI-Steuerung anfordern**. Zur Anzeige dieses Menüs kann auch die Statusleiste am unteren Rand des Frontpanels angeklickt werden. Wenn das VI von keinem anderen Client ausgeführt wird, erhalten Sie eine Meldung darüber, dass Sie die Steuerung des VIs übernommen haben. Wird das Frontpanel gerade von einem anderen VI gesteuert, wird Ihre Anfrage auf eine Warteliste gesetzt, bis das andere VI die Ressourcen wieder freigibt oder die Zugriffszeit des anderen Clients abgelaufen ist. Um auf dem Server-Rechner eine Liste der Clients anzuzeigen, ist **Werkzeuge»Remote-Panel-Verbindungsmanager** auszuwählen.

Wenn die Daten des auf dem Netzwerkcomputer ausgeführten VIs gespeichert werden sollen, empfiehlt es sich, anstelle von Netzwerk-Frontpanels DataSocket oder TCP zu verwenden. Weitere Informationen zur Verwendung von DataSocket befnden sich in Abschnitt *Verwenden der DataSocket-Technologie* dieses Kapitels. Für nähere Hinweise zu TCP lesen Sie den Abschnitt *TCP und UDP* dieses Kapitels.

Alle VIs, die auf Client-Rechnern darstellbar und steuerbar sein sollen, müssen sich im Speicher befinden. Nur dann kann der Server das Frontpanel an den Client senden. Wenn sich das VI nicht im Speicher befindet, wird unter **Verbindungsstatus** im Dialogfenster **Netzwerk-Panel öffnen** eine Fehlermeldung angezeigt.

Darstellung und Steuerung von Frontpanels in einem Web-Browser

Um VIs auch auf Client-Rechnern über einen Web-Browser darstellbar und steuerbar zu machen, auf denen die LabVIEW-Entwicklungsumgebung nicht installiert ist, muss auf diesen Rechnern die LabVIEW-Runtime-Engine installiert werden. In dieser Runtime-Engine ist ein LabVIEW-Browser-Plug-in-Paket enthalten, das in das Browser-Plug-in-Verzeichnis installiert wird. Ein entsprechendes Installationsprogramm ist auf der LabVIEW-CD enthalten.

Auf den Client-Rechnern ist die LabVIEW-Runtime-Engine zu installieren, und auf dem Server-Rechner ist eine HTML-Datei mit einem `<OBJECT>`- und einem `<EMBED>`-Tag zu erstellen, das auf das VI verweist, das über das Netzwerk gesteuert werden soll. Dieses Tag enthält eine URL-Referenz zu einem VI und Informationen, die den Web-Browser anweisen, das VI an das LabVIEW-Browser-Plug-In zu übergeben. Auf diese Weise können die Clients zum Webserver navigieren, indem sie in das Adressfeld die entsprechende URL eingeben. Über das Plug-In wird das Frontpanel im Web-Browser-Fenster angezeigt und die Kommunikation mit dem Webserver gesteuert, so dass ein interaktiver Zugriff auf das netzwerkgesteuerte Frontpanel möglich ist. Um ein VI bedienen zu können, muss am unteren Rand oder im Kontextmenü des netzwerkgesteuerten Frontpanels, das über einen Rechtsklick angezeigt wird, die Option **VI-Steuerung anfordern** ausgewählt werden.

Wenn weder ein anderer Client noch ein anderes Browser-Fenster die Steuerung des Frontpanels übernommen hat, erscheint eine Meldung, die darauf hinweist, dass Sie jetzt die Steuerung übernommen haben. Sollte das Frontpanel gerade auf einem anderen Client verwendet werden, wird Ihre Anfrage auf eine Warteliste gesetzt, bis die Ressourcen wieder freigegeben

werden oder die Zugriffszeit des anderen Clients abgelaufen ist. Die Clients, die sich in der Warteschlange befinden, können nur auf dem Server-Rechner angezeigt werden. Dazu ist **Werkzeuge»Remote-Panel-Verbindungsmanager** zu wählen.

Hinweis Zur Anzeige eines Frontpanels wird empfohlen, eine Netscape-Version ab 4.7 bzw. eine Internet-Explorer-Version ab 5.5 zu verwenden.

Einschränkungen in Bezug auf Darstellung und Steuerung von Frontpanels über das Netzwerk

Aufgrund der verschiedenen Eigenschaften von Web-Browsern können Höhe und Breite sowie die Position von Frontpanels im Netzwerk anders dargestellt werden als in LabVIEW. Obwohl beim Webserver und dem LabVIEW-Browser-Plug-In immer eine möglichst originalgetreue Darstellung komplexer Benutzeroberflächen angestrebt wird, kann es – speziell bei Applikationen mit Dialogfeldern und SubVI-Fenstern – vorkommen, dass diese im Browser nicht ordnungsgemäß arbeiten. Es wird daher empfohlen, auf die Darstellung solcher Applikationen in einem Web-Browser zu verzichten.

Es sollte darauf geachtet werden, keine VIs mit While-Schleifen zu exportieren, die keine Warte-Funktion enthalten, da solche VIs den im Hintergrund laufenden Tasks Rechenzeit entziehen, was dazu führen kann, dass das entsprechende Frontpanel im Netzwerk nicht mehr reagiert.

Manche VIs auf dem Netzwerk-Computer arbeiten auch nicht so wie auf einem lokalen Rechner. So werden zum Beispiel ActiveX-Elemente auf einem Frontpanel auf dem Netzwerk-Client nicht dargestellt, da sie fast völlig unabhängig von LabVIEW arbeiten. Wenn in einem netzwerkgesteuerten VI ein Standard-Dateifenster geöffnet wird, wird eine Fehlermeldung ausgegeben, da es nicht möglich ist, ein Dateisystem über das Netzwerk zu durchsuchen. Die Schaltfläche “Durchsuchen” des Pfad-Bedienelements ist bei Netzwerk-Frontpanels ebenfalls deaktiviert.

Frontpanels, die über das Netzwerk angezeigt werden, könnten sich unterschiedlich verhalten, je nachdem, ob sie von einer ausführbaren Anwendung stammen oder nicht. In diesem Fall werden programmatische Änderungen am Frontpanel, die stattfinden, bevor der Client damit verbunden wird, nicht an diesen weitergegeben. Wenn zum Beispiel der Untertitel eines Bedienelements über einen Eigenschaftsknoten verändert wird, bevor der Client-Rechner das Frontpanel abruft, wird auf diesem der ursprüngliche Untertitel angezeigt.

Nur ein Client mit Steuerungszugriff auf ein VI kann das Frontpanel eines dynamisch geöffneten und ausgeführten VIs anzeigen, indem dazu der VI-Server verwendet wird oder ein SubVI, das das Frontpanel anzeigt, wenn es aufgerufen wird. Bei anderen Clients, die keine Zugriffsrechte haben, wird das Frontpanel nicht angezeigt.

Blockdiagramme, bei denen die Benutzeroberfläche kontinuierlich nach Änderungen an Bedienelementen abgefragt wird, können sich bei der Ausführung über das Netzwerk verlangsamen. In diesem Fall empfiehlt es sich, zur Abfrage von Änderungen an Frontpanel-Elementen die VIs “Auf Frontpanelaktivität warten” zu verwenden.

Versenden von VI-Daten per E-Mail

Zum Versenden von E-Mails und Dateianhängen mittels SMTP (Simple Mail Transfer Protocol) sollten die E-Mail-VIs verwendet werden. Da in LabVIEW die Anmeldung über SMTP nicht unterstützt wird, ist es nicht möglich, Daten zu empfangen. Zur Codierung der Nachrichten wird bei den VIs das MIME-Format (Multipurpose Internet Mail Extensions) verwendet. Mit diesem Format lassen sich verschiedene Arten von Dokumenten (so zum Beispiel auch Binärdateien) per E-Mail versenden. Auch ist es möglich, die Eigenschaften der einzelnen Anhänge zu beschreiben, zum Beispiel welcher Zeichensatz in einem Textdokument verwendet wird. Zusätzliche Informationen zum Versand von VI-Daten per E-Mail befinden sich in der *LabVIEW-Hilfe*.

Hinweis Die E-Mail-VIs sind nur im Full und Professional Development System verfügbar.

Zur Verwendung der E-Mail-VIs ist neben den Adressen der Empfänger auch die Webadresse des SMTP-Servers erforderlich. Daneben ist außerdem über dem Eingang **Mail-Server** für jedes E-Mail-VI ein Mail-Server anzugeben. Hier sind Host-Name oder IP-Adresse eines externen Server-Rechners zu nennen, der Anfragen vom Computer bearbeiten kann, auf dem die E-Mail-VIs ausgeführt werden. Falls Sie nicht wissen, welchen Mail-Server Sie verwenden sollen, wenden Sie sich an Ihren Netzwerkadministrator. Nach Angabe eines gültigen Servers wird von den E-Mail-VIs eine Verbindung zum Mail-Server hergestellt und es werden die Serverbefehle für Empfänger sowie Inhalt der E-Mail übermittelt. Vom Server aus wird die Nachricht dann an die einzelnen Empfänger oder an andere SMTP-Server weitergeleitet. Beispiele zur Anwendung der E-Mail-VIs finden Sie unter `examples\comm\smtpex.llb`.

Hinweis Multibyte-Zeichensätze wie japanische Schriftzeichen werden von den SMTP-E-Mail-VIs nicht unterstützt.

Auswahl eines Zeichensatzes

Im Eingang **Zeichensatz** der E-Mail-VIs wird angegeben, welcher Zeichensatz in der E-Mail oder dem Anhang verwendet werden soll. Damit wird angegeben, welcher Code jedem Zeichen zugeordnet ist.

Ein Zeichen ist eine Grundeinheit der geschriebenen Sprache, also ein Buchstabe, eine Zahl, ein Satzzeichen oder in manchen Sprachen auch ein ganzes Wort. Buchstaben, die verschieden geschrieben werden (beispielsweise groß und klein oder einen Akzent tragen), werden dabei als unterschiedliche Zeichen interpretiert, wie `O`, `o`, Ö und Ô.

Ein Zeichencode ist eine Zahl, die für ein Zeichen steht. Eine Codierung der Zeichen ist notwendig, da in Computern nur Zahlen verarbeitet werden können.

In einem Zeichensatz wird die Zuordnung zwischen Zeichen und den numerischen Werten beschrieben, die im Computer verarbeitet werden. Im ASCII-Zeichensatz entspricht beispielsweise den Zeichen `A`, `B` und `C` der Code `65`, `66` und `67`.

Amerikanischer ASCII-Zeichensatz

Ein für E-Mails gängiger Zeichensatz ist ASCII. Bei vielen E-Mail-Programmen ist dieser Zeichensatz voreingestellt, und es werden keine anderen Zeichensätze verwendet. In ASCII sind alle Buchstaben und fast alle Satzzeichen der englischen Sprache enthalten, insgesamt 128 Zeichen. Bei den meisten anderen Zeichensätzen handelt es sich um Erweiterungen des ASCII-Codes.

Bei Sprachen, in denen Zeichen verwendet werden, die in ASCII nicht enthalten sind, ist dieser jedoch unzureichend. So könnte beispielsweise das Wort `Müller` nicht in ASCII geschrieben werden, da das Zeichen `ü` nicht darin enthalten ist.

Zeichensatz ISO Latin-1

Da in vielen Sprachen Zeichen enthalten sind, die nicht in ASCII vorkommen, sind entsprechende Zeichensätze entwickelt worden. Die meisten davon umfassen die ersten 128 Zeichen von ASCII sowie 128 weitere Zeichen, die nur in dieser Sprache vorkommen. Bei einigen dieser Zeichensätze werden dieselben Zeichen verschieden codiert. Dadurch

können bei der Darstellung von Zeichen aus anderen Zeichensätzen Probleme auftreten. Aus diesem Grund sollte immer ein Standardzeichensatz wie ISO Latin-1 (ISO-8859-1) verwendet werden. Der Zeichensatz, der die Zeichen der meisten westeuropäischen Sprachen umfasst, wird auch von vielen E-Mail-Programmen, in denen diese Sprache verwendet werden kann, eingesetzt.

Zeichensatz für Mac OS

Die Firma Apple hat bereits vor Entwicklung des ISO-Latin-1-Standards einen eigenen Zeichensatz entwickelt. Dieser basiert auf dem ASCII-Standard, wobei jedoch die 128 Zeichen, die darüber hinausgehen, anders als in ISO Latin-1 codiert sind. Aus diesem Grund werden bei Verwendung dieses Zeichensatzes Zeichen mit Akzent, die auf einem Apple-Computer eingegeben worden sind, nicht korrekt dargestellt. Vor dem Senden wird der Text daher in den E-Mail-Programmen unter Mac OS zunächst in den ISO-Latin-1-Standard konvertiert. Beim E-Mail-Empfang werden Texte, die in ISO Latin-1 codiert sind, in den Zeichensatz von Mac OS umgewandelt.

Zeichenkonvertierung

Bei der Zeichenkonvertierung werden Zeichen in einen anderen Zeichensatz umgewandelt. Eine Zeichenkonvertierung ist beispielsweise notwendig, wenn Sie eine E-Mail schreiben und einen anderen Zeichensatz verwenden möchten. Bei den E-Mail-VIs für SMTP ist es möglich, vor dem E-Mail-Versand für den Text einen bestimmten Zeichensatz auszuwählen. Wenn beispielsweise eine Nachricht im Standard-ASCII-Format erstellt wurde und auf MacRoman umgestellt werden soll, wird der Text von den VIs umgewandelt und im `iso-8859-1`-Format (ISO Latin-1) versendet. Welcher Zeichensatz verwendet werden soll, kann über den VI-Eingang **Zeichenkonvertierung** festgelegt werden. Zur Konvertierung ist immer ein Ausgangs- und ein Zielzeichensatz notwendig sowie eine Datei, in der beschrieben ist, welche Zeichen einander zugewiesen werden sollen.

Die Konvertierungsdatei ist eine Binärdatei mit 256 Einträgen und einer Größe von 256 Byte. Jeder Eintrag in der Datei besteht aus einem Zeichen eines vorliegenden Zeichensatzes und dem entsprechenden Zeichen im Zielzeichensatz. Damit beispielsweise das Zeichen `a`, dessen Code `61` ist, in das Zeichen `A` mit dem Code `41` umgewandelt werden kann, muss in der Datei beim Eintrag mit dem Index `61` der Wert `41` stehen. Wenn Index und Wert übereinstimmen, wird das entsprechende Zeichen im Ausgangszeichensatz nicht verändert. Angenommen, der Eintrag mit dem Index `61` in

der Datei enthielte den Wert `61`, würde das entsprechende Zeichen unverändert bleiben.

Wenn am Eingang **Zeichenkonvertierung** die entsprechende Datei als Zielzeichensatz angegeben wird, erfolgt die Zuweisung in der angegebenen Reihenfolge. Würde zum Beispiel der Eintrag für die Zeichenkonvertierung [MacRoman iso-8859-1 `macroman.trl`, MacRomanUp MacRoman `asciiup.trl`] lauten, so würde der Zeichensatz MacRoman mit `macroman.trl` in iso-8859-1 und MacRomanUp mit `asciiup.trl` MacRoman umgewandelt werden. Beispiele für `.trl`-Dateien befinden sich im Verzeichnis `vi.lib\Utility\SMTP`.

Low-Level-Kommunikationsanwendungen

In LabVIEW werden verschiedene Low-Level-Protokolle für die Kommunikation zwischen Computern unterstützt.

Alle Protokolle unterscheiden sich insbesondere in Hinsicht darauf, wie der Verweis auf die Netzwerkadresse einer Netzwerkapplikation erfolgt. Im Normalfall sind Protokolle nicht miteinander kompatibel. Wenn eine Verbindung zur Kommunikation zwischen Mac OS und Windows hergestellt werden soll, muss ein Protokoll verwendet werden, das auf beiden Plattformen funktioniert, wie zum Beispiel TCP.

TCP und UDP

Die Protokolle TCP (Transmission Control Protocol) und UDP (User Datagram Protocol) stehen auf allen von LabVIEW unterstützten Plattformen zur Verfügung. Bei TCP handelt es sich um ein zuverlässiges, verbindungsorientiertes Protokoll. Es bietet Fehlererkennung und gewährleistet, dass die Daten in der richtigen Reihenfolge und nicht dupliziert ankommen. TCP ist daher in der Regel für Netzwerkapplikationen am besten geeignet.

Mit UDP kann zwar eine höhere Leistung als mit TCP erreicht werden, aber es ist keine Verbindung erforderlich und es kann nicht garantiert werden, dass alle Daten übermittelt werden. Normalerweise wird UDP in Applikationen verwendet, bei denen Datenverluste kein Problem darstellen. Wenn Daten von einer Applikation beispielsweise oft genug an ein Ziel gesendet werden, ist der Verlust einiger Datensegmente weniger relevant. Weitere Informationen zum Verwenden von TCP und UDP in LabVIEW-Applikationen finden Sie in der Application Note *Using LabVIEW with TCP/IP and UDP*.

Zum Herstellen von Verbindungen mit Multicast-IP-Adressen zum Lesen, Schreiben bzw. Lesen und Schreiben von UDP-Daten sollte anstelle der Funktion “UDP: Öffnen” das VI “UDP: Multicast öffnen” verwendet werden. Über eine Multicast-IP-Adresse wird eine Multicast-Gruppe festgelegt. Die Multicast-IP-Adressen liegen im Bereich von `224.0.0.0` bis `239.255.255.255`. Um eine Multicast-Gruppe zu abonnieren, muss sich ein Client unter der Multicast-IP-Adresse einschreiben. Daraufhin erhält er alle Daten, die an die Multicast-IP-Adresse gesendet werden. Weitere Informationen zum *Verwenden von TCP/IP und UDP in LabVIEW*-Applikationen finden Sie in der entsprechenden Application Note.

Apple-Ereignisse und PPC-Toolbox (Mac OS)

Eine der gängigen Kommunikationsmethoden unter Mac OS sind Apple-Ereignisse. Mit Apple-Ereignissen können Nachrichten versendet werden, mit denen Aktionen angefordert oder Informationen von anderen Mac-Anwendungen zurückgegeben werden.

Bei der PPC-Toolbox (PPC = Program-to-Program Communication) werden Daten zwischen Mac-Anwendungen systemnah ausgetauscht. Die PPC-Toolbox bietet eine höhere Leistung als Apple-Ereignisse, da der zum Übertragen von Informationen erforderliche Overhead geringer ist. Da aber die Art der übertragbaren Informationen nicht angegeben ist, wird diese Toolbox von vielen Applikationen nicht unterstützt. Die PPC-Toolbox eignet sich jedoch optimal zum Senden umfangreicher Informationen zwischen Applikationen, die die Toolbox unterstützen. Weitere Informationen zum Verwenden von Apple-Ereignissen und der PPC-Toolbox in LabVIEW-Applikationen befinden sich in den Application Notes *Using Apple Events and the PPC Toolbox to Communicate with LabVIEW Applications on the Macintosh*.

Pipe-VIs (UNIX)

Mit den Pipe-VIs lassen sich UNIX-Named Pipes öffnen, schließen, lesen und schreiben. Named Pipes ermöglichen eine Kommunikation zwischen LabVIEW und nicht verwandten Prozessen.

Ausführen von Befehlen auf Systemebene (Windows und UNIX)

Das VI “System Exec” dient zur Ausführung von Windows-Anwendungen und UNIX-Befehlszeilenanwendungen. Die entsprechende Befehlszeile wird auf Systemebene ausgeführt und kann beliebige Parameter enthalten, die von der zu startenden Anwendung unterstützt werden.

19

Windows-Konnektivität

Mit Hilfe der ActiveX- oder .NET-Technologie ist es in LabVIEW möglich, auf andere Windows-Anwendungen zuzugreifen. In .NET kann LabVIEW als Client für den Zugriff auf Objekte, Eigenschaften und Methoden von .NET-Servern eingesetzt werden. LabVIEW ist kein .NET-Server. Das heißt, andere Anwendungen können mit LabVIEW über .NET nicht direkt kommunizieren. Mit einem VI, das für .NET aktiviert ist, können Verbindungen zu Windows Services und APIs hergestellt werden. Das .NET Framework umfasst die Komponentendienste COM+ sowie das ASP Web Development Framework und unterstützt Protokolle für Internetdienste wie SOAP, WSDL oder UDDI. Das .NET Framework ist das Programmiermodell der .NET-Plattform zum Erstellen, Bereitstellen und Ausführen von Anwendungen – speziell von Web- und Smart-Client-Anwendungen sowie von XML-Webdiensten.

Mit Hilfe der ActiveX-Technologie werden von einer Windows-Anwendung Objekte, Befehle und Funktionen zur Verfügung gestellt, auf die andere Windows-Applikationen zugreifen können. So kann LabVIEW beispielsweise als ActiveX-Client eingesetzt werden, wodurch das Programm auf Objekte, Eigenschaften, Methoden und Ereignisse anderer ActiveX-fähiger Applikationen zuzugreifen. Umgekehrt kann LabVIEW auch als ActiveX-Server fungieren und anderen ActiveX-fähigen Applikationen den Zugriff auf LabVIEW-Objekte, -Eigenschaften und -Methoden ermöglichen.

Die Abkürzung *.NET* steht für die .NET-Technologie von Microsoft. Um .NET zu nutzen, müssen Sie das .NET Framework installieren. Nähere Details zu .NET und zur Installation des .NET Framework erhalten Sie auf der Webseite von MSDN (Microsoft Developer's Network). Die Bezeichnung *ActiveX* steht im vorliegenden Handbuch für die ActiveX- und OLE-Technologie von Microsoft. Weitere Informationen zu ActiveX finden Sie in den MSDN-Büchern *Inside OLE* von Kraig Brockschmidt, zweite Ausgabe, und *Essential COM* von Don Box.

Weitere Informationen …

Informationen zur Anwendung der .NET- und ActiveX-Technologie entnehmen Sie bitte der *LabVIEW-Hilfe* oder der Webseite von National Instruments, `ni.com`.

.NET-Umgebung

Nachfolgend sind die verschiedenen Elemente der .NET-Umgebung aufgeführt. Diese Informationen sollen Ihnen .NET verständlicher machen, sind jedoch für die Verwendung der .NET-Komponenten in LabVIEW nicht zwingend erforderlich.

- **Common Language Runtime (CLR)**—Die Common Language Runtime besteht aus mehreren Bibliotheken, die für Laufzeitfunktionen wie die Sprachintegration, den Arbeitsspeicher, die Überwachung von Schutzverletzungen oder das Ressourcen-, Prozess- bzw. Thread-Management verantwortlich sind. Die CLR bildet die Grundlage von .NET und arbeitet mit der von allen Programmiersprachen erzeugten so genannten “Intermediate Language” (IL), über die .NET relativ einfach mit anderen Programmen kommunizieren kann.

  Durch die vielfältigen in der CLR verwendeten Datentypen stellt .NET eine Schnittstelle zwischen den verschiedensten Programmiersprachen und Plattformen dar. Während im Allgemeinen ein System als Zusammensetzung von Speicherinhalten und Threads betrachtet wird, können mit Hilfe der CLR die Datentypen und Objekte eines Systems angezeigt werden. Daten werden in der CLR zunächst in die Zwischensprache IL übersetzt, wozu Compiler und Linker erforderlich sind. In einem Win32-System wird von allen Maschinensprachen-Compilern anstatt von Assembly-Code der CLR-IL-Code erzeugt.

- **Klassenbibliotheken**—Stellen Grundfunktionen wie Ein- und Ausgabe, String-Manipulation, Sicherheitsmanagement, Netzwerkkommunikation, Thread- und Textverwaltung oder Funktionalität der Benutzeroberfläche bereit. Die Funktionen dieser Klassen sind dieselben wie im Win32/COM-System. Im .NET Framework können Klassen, die in einer .NET-Sprache erstellt wurden, auch in anderen .NET-Sprachen verwendet werden.

- **Assemblies**—Einsatzfähige Komponenten wie .DLL-, .OCX- oder ausführbare COM-Komponenten. Assemblies sind in .NET CLR erstellte DLLs und lauffähige Anwendungen. Sie können aus einer Datei oder aus mehreren Dateien bestehen. Jede Assembly enthält ein Manifest mit dem Namen, der Version, dem Ländercode, Sicherheitsinformationen, einer Liste der Dateien, aus denen die Assembly besteht, den referenzierten Assemblies, Ressourcen und exportierten Datentypen. Wenn Assemblies nur aus einer Datei bestehen, sind darin alle Daten einschließlich des Manifests und der benötigten Ressourcen enthalten. Assemblies mit mehreren Dateien können externe Ressourcen wie Bitmaps, Symbole oder Sounddateien umfassen. Es kann aber

auch jeweils eine Datei für den Core Code und die Hilfsbibliotheken verwendet werden.

Assemblies können entweder von allen Anwendungen (global) oder nur von einer Anwendung nutzbar (privat) sein. Private Assemblies müssen immer im Programmverzeichnis einer Applikation und öffentliche Assemblies an einem zentralen Speicherort, dem so genannten Global Assembly Cache (GAC), gespeichert werden. Private Assemblies werden meist nur für die Verwendung mit der Applikation geschrieben, in deren Verzeichnis sie gespeichert sind. Dem Entwickler einer Assembly obliegt auch die Versionskontrolle. Der Name der Assembly stimmt, mit Ausnahme der Erweiterung, mit dem der Datei überein, in der das Manifest enthalten ist.

- **Global Assembly Cache (GAC)**—Enthält alle öffentlich zugänglichen Assemblies im System. Der GAC ähnelt der Registry in COM.

.NET-Funktionen und -Knoten

Mit den nachfolgenden LabVIEW-Funktionen und -Knoten kann auf Objekte, Eigenschaften und Methoden von .NET-Servern zugegriffen werden.

- Mit Hilfe des Konstruktor-Knotens, der sich auf der **.NET**-Palette befindet, kann ein Konstruktor einer .NET-Klasse aus einer Assembly ausgewählt und bei der Ausführung eine Instanz davon erstellt werden. Wenn ein solcher Knoten in das Blockdiagramm eingefügt wird, öffnet sich das Dialogfeld **.NET-Konstruktor auswählen**.
- Mit den Eigenschaftsknoten in der **.NET**-Palette können die Eigenschaften einer .NET-Klasse abgefragt (gelesen) oder festgelegt (gesetzt) werden.
- Der Methodenknoten auf der **.NET**-Palette ermöglicht es, die Methoden einer .NET-Klasse abzurufen.
- Mit der Funktion “Referenz schließen” auf der **.NET**-Palette können alle Referenzen zu .NET-Objekten beendet werden, wenn diese Zuordnung nicht mehr benötigt wird.
- Die Funktion “Nach allgemeinerer Klasse” auf der **.NET**-Palette dient dazu, eine Referenz in ihre Basisklasse zu konvertieren.
- Über die Funktion “Nach spezifischerer Klasse” auf der **.NET**-Palette kann eine Referenz in ihre abgeleitete Klasse konvertiert werden.

LabVIEW als .NET-Client

Wenn LabVIEW auf die Objekte einer .NET-Assembly zugreift, fungiert das Programm wie ein .NET-Client. Um das Programm als .NET-Client zu verwenden, ist nach folgenden drei wesentlichen Schritten vorzugehen:

1. Erzeugen Sie mit Hilfe des Konstruktors ein .NET-Objekt und erstellen Sie eine Referenz auf das Objekt.
2. Verbinden Sie diese mit einem Eigenschafts- oder Methodenknoten und wählen Sie eine Eigenschaft bzw. Methode aus.
3. Wenn die Zuweisung der Referenz zum Objekt aufgehoben werden soll, schließen Sie sie.

Zum Zugriff auf ein .NET-Objekt ist zunächst im Blockdiagramm das entsprechende Objekt zu erstellen. Dazu verwenden Sie den Konstruktor-Knoten. Mit Hilfe dieses Knotens kann eine Objekt-Klasse aus einer Assembly ausgewählt werden. Wenn Sie einen Konstruktor-Knoten in das Blockdiagramm einfügen, öffnet sich das Dialogfeld **.NET-Konstruktor auswählen**, in dem alle öffentlich zugänglichen Assemblies im GAC aufgeführt sind. Wenn statt dessen eine private Assembly verwendet werden soll, klicken Sie im Dialogfeld auf die Schaltfläche **Durchsuchen** und wählen Sie die Assembly aus dem Verzeichnissystem aus. Gültige Dateitypen für .NET-Assemblies sind `.dll` und `.exe`. Wenn das Dialogfeld **.NET-Konstruktor auswählen** das nächste Mal geöffnet wird, erscheint die ausgewählte private Assembly ebenfalls im Pulldown-Menü **Assembly**.

Wenn Sie eine Assembly und eine Klasse ausgewählt haben, werden unter **Konstruktoren** die Konstruktoren zur entsprechenden Klasse aufgeführt. Wählen Sie einen Konstruktor aus und klicken Sie anschließend auf **OK**. Im Konstruktor-Knoten wird nun der Name der ausgewählten Klasse angezeigt.

Der Konstruktor-Knoten ist mit der Funktion “ActiveX-Objekt öffnen” vergleichbar, bietet jedoch den Vorteil, dass auch Initialisierungsparameter zur Erstellung des Objekts festgelegt werden können. Die .NET-Server-Referenz des Knotens kann mit einem Eigenschafts- oder Methodenknoten verbunden werden, wo aus dem Kontextmenü eine Eigenschaft bzw. Methode ausgewählt werden kann. Mit Hilfe der Funktion “Referenz schließen” kann die Referenz auf das .NET-Objekt geschlossen werden.

Weitere Hinweise zum Erstellen von .NET-Objekten finden Sie auch in der *LabVIEW-Hilfe*.

Konvertierung von Datentypen

Damit die entsprechenden Daten von LabVIEW gelesen werden können, müssen die Datentypen der .NET-Eigenschaft, -Methode und der Parameter des Konstruktors in LabVIEW-Typen umgewandelt werden. Datentypen, die nicht umgewandelt werden konnten, werden als .NET-RefNums angezeigt. In der Tabelle sehen Sie, wie die Zuordnung der Datentypen erfolgt.

Tabelle 19-1. .NET- und LabVIEW-Datentypen

.NET-Typen	LabVIEW-Typen
System.Int32, System.UInt32, System.Int16, System.UInt16	I32, U32, I16, U16
System.String	abc
System.Boolean	TF
System.Byte, System.Char, System.UByte	I8, U8, U8
System.Single, System.Double, System.Decimal	SGL, DBL, DBL
System.Array	Wird als Array des entsprechenden Typs dargestellt.
Enum	(selten)
DateTime	
Sonstiges .NET-Objekt	

Bereitstellen von .NET-Applikationen

Nachdem Sie ein VI erstellt haben, in dem .NET-Komponenten enthalten sind, können Sie dieses in eine ausführbare Anwendung, eine LLB oder eine DLL umwandeln.

Bereitstellen von ausführbaren Anwendungen

Wenn Sie eine ausführbare Anwendung erstellen, in der ein .NET-Objekt enthalten ist, müssen Sie privaten Assemblies, die von diesem VI genutzt werden, in das dazugehörige Programmverzeichnis kopiert werden. Außerdem muss auf dem Zielrechner das .NET Framework installiert sein.

Bereitstellen von VIs

Wenn Sie statt dessen die VIs für externen Zugriff zur Verfügung stellen möchten, müssen Sie die von den VIs verwendeten privaten Assemblies in das Verzeichnis des Haupt-VIs kopieren. Es ist wichtig, dass sich alle verwendeten Assemblies in derselben Verzeichnisstruktur wie das Haupt-VI befinden und dass auf dem Zielcomputer das .NET Framework installiert ist.

Bereitstellen von DLLs

Wenn Sie eine in LabVIEW erstellte DLL bereitstellen möchten, speichern Sie alle privaten Assemblies, die von den darin befindlichen VIs genutzt werden, in das Verzeichnis der Anwendung, in der die DLL verwendet wird. Außerdem muss auf dem Zielrechner das .NET Framework installiert sein.

Erweiterte Konfiguration einer .NET-Client-Anwendung

Über Konfigurationsdateien können .NET-Applikationen Verwaltungsaufgaben übernehmen. Konfigurationsdateien sind in XML geschrieben und meist durch die Erweiterung `.config` gekennzeichnet. Bei der Konfiguration einer .NET-Client-Applikation, die in LabVIEW erstellt wurde, kann dem entsprechenden Haupt-VI oder der ausführbaren Anwendung ebenfalls eine Konfigurationsdatei hinzugefügt werden. Eine solche Datei sollte immer mit dem Namen des VIs bzw. der Anwendung und der Erweiterung `.config` versehen werden, zum Beispiel `My App.vi.config` oder `MyApp.exe.config`.

ActiveX-Objekte, -Eigenschaften, -Methoden und -Ereignisse

ActiveX-fähige Applikationen enthalten Objekte mit bestimmten Eigenschaften und Methoden, auf die andere Applikationen zugreifen können. Diese Objekte können für den Benutzer sichtbar sein, wie zum Beispiel Schaltflächen, Fenster, Bilder, Dokumente und Dialogfelder, oder unsichtbar, wie zum Beispiel Registrierungsobjekte von Applikationen. Der Zugriff auf eine Applikation erfolgt also über ein dazugehöriges Objekt, indem für das Objekt eine Eigenschaft festgelegt oder eine Methode aufgerufen wird.

Weitere Informationen zu Objekten, Eigenschaften und Methoden in der LabVIEW-Umgebung finden Sie im Abschnitt *Bearbeiten von Applikations- und VI-Einstellungen* des Kapitels 17, *Programmatische Steuerung von VIs*.

Ereignisse werden durch Aktionen an einem Objekt ausgelöst, wie zum Beispiel durch einen Mausklick, Tastendruck oder Speicherüberlauf. Immer, wenn an dem Objekt eine dieser Aktionen vorgenommen wird, sendet das Objekt ein Ereignis und die dazugehörigen Informationen an den ActiveX-Container.

Nähere Hinweise dazu, wie ActiveX-Ereignisse mit dem Knoten “Ereignis-Callback registrieren” verarbeitet werden können, finden Sie im Abschnitt *ActiveX-Ereignisse* dieses Kapitels.

ActiveX-VIs, -Funktionen, -Bedienelemente und -Anzeigeelemente

Verwenden Sie die folgenden VIs, Funktionen, Bedien- und Anzeigeelemente, um auf die Objekte, Eigenschaften, Methoden und Ereignisse zuzugreifen, die von anderen ActiveX-fähigen Applikationen stammen:

- Mit Hilfe des ActiveX-RefNum-Bedienelements kann eine Referenz auf ein ActiveX-Objekt erstellt werden. Klicken Sie im Frontpanel mit der rechten Maustaste auf dieses Bedienelement, um ein Objekt aus der ActiveX-Klasse auszuwählen, auf die Sie zugreifen möchten.
- Über eine ActiveX-Objekt-RefNum kann ein ActiveX-Objekt ausgewählt werden.
- Über den ActiveX-Container ist es möglich, auf dem Frontpanel ein ActiveX-Objekt anzeigen zu lassen. Klicken Sie diesen dazu mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü über die Option **ActiveX-Objekt einfügen** das anzuzeigende Objekt aus.
- Über einen Eigenschaftsknoten können die Eigenschaften eines ActiveX-Objekts festgelegt und abgefragt werden.
- Mit Hilfe eines Methodenknotens können die zum Objekt gehörenden Methoden aufgerufen werden.
- Mit Hilfe des Knotens “Ereignis-Callback registrieren” können Ereignisse an ActiveX-Objekten verarbeitet werden.
- Verwenden Sie zum Datenaustausch mit ActiveX-Bedienelementen das Variant-Bedien- und Anzeigeelement. Weitere Informationen zu Variant-Daten finden Sie im Abschnitt *Verarbeiten von Variant-Daten* des Kapitels 5, *Erstellen des Blockdiagramms*.

LabVIEW als ActiveX-Client

Wenn LabVIEW auf die Objekte einer anderen ActiveX-fähigen Applikation zugreift, fungiert das Programm als ActiveX-Client. Sie können LabVIEW folgendermaßen als ActiveX-Client verwenden:

- Öffnen Sie eine Applikation, wie zum Beispiel Microsoft Excel, erstellen Sie ein Dokument und fügen Sie Daten in das Dokument ein.
- Betten Sie mittels eines Containers ein Dokument wie beispielsweise ein Microsoft-Word-Dokument oder eine Excel-Tabelle in das Frontpanel ein.
- Fügen Sie nun eine Schaltfläche oder ein Objekt aus einer anderen Applikation hinzu, zum Beispiel eine **Hilfe**-Schaltfläche, über die die Hilfedatei der anderen Applikation aufgerufen werden kann.
- Erstellen Sie eine Verknüpfung zu einem ActiveX-Objekt, das mit einer anderen Applikation erstellt wurde.

Zum Zugriff auf ein ActiveX-Objekt über das Frontpanel wird in LabVIEW entweder die ActiveX-Objekt-RefNum oder der ActiveX-Container verwendet. Mit beiden Elementen können ActiveX-Objekte ausgewählt werden. Für anzeigbare ActiveX-Objekte wie Schaltflächen oder Dokumente sollte der Container verwendet werden. Auf dem Blockdiagramm werden beide als ActiveX-Objekt-RefNums angezeigt.

Zugriff auf ActiveX-fähige Applikationen

Um eine Referenz auf eine ActiveX-fähige Applikation zu erstellen, damit auf diese zugegriffen werden kann, verwenden Sie die ActiveX-Objekt-RefNum im Blockdiagramm. Verbinden Sie das Objekt mit der Funktion “ActiveX-Objekt öffnen”, mit der die aufrufende Applikation geöffnet wird. Verwenden Sie den Eigenschaftsknoten, um die zu dem Objekt gehörenden Eigenschaften, und den Methodenknoten, um die Objekt-Methoden auszuwählen und darauf zuzugreifen. Beenden Sie die Referenz auf das Objekt mit Hilfe der Funktion “Referenz schließen”. Das Objekt wird damit aus dem Speicher entfernt.

So können Sie beispielsweise ein VI erstellen, das das Programm Microsoft Excel auf dem Bildschirm öffnet, ein Arbeitsblatt und Spreadsheet erstellt, eine LabVIEW-Tabelle erzeugt und diese in das Excel-Spreadsheet schreibt.

Ein Beispiel zur Verwendung von LabVIEW als Excel-Client finden Sie im VI “Write Table To XL” unter `examples\comm\ExcelExamples.llb`.

Hinweis Anwendungen mit benutzerdefinierten ActiveX-Schnittstellen sind mit dem Symbol gekennzeichnet. Klicken Sie auf dieses Symbol, um ein Objekt für die benutzerdefinierte Schnittstelle auszuwählen. Weitere Informationen zu benutzerdefinierten Schnittstellen finden Sie im Abschnitt *Unterstützung benutzerdefinierter ActiveX-Schnittstellen* dieses Kapitels.

Einbinden eines ActiveX-Objekts in das Frontpanel

Um ein ActiveX-Objekt in das Frontpanel einzufügen, klicken Sie den ActiveX-Container mit der rechten Maustaste an und wählen Sie dann über die Option **ActiveX-Objekt einfügen** das gewünschte Bedienelement oder Dokument aus. Um Eigenschaften zu einem ActiveX-Objekt festzulegen, verwenden Sie die Eigenschaftsseiten, den Eigenschafts-Browser oder einen Eigenschaftsknoten im Blockdiagramm. Für weitere Informationen zu ActiveX-Eigenschaften lesen Sie bitte den Abschnitt *Festlegen von ActiveX-Eigenschaften* dieses Kapitels.

Um die Methoden eines Objekts aufzurufen, verwenden Sie einen Methodenknoten.

Wenn Sie beispielsweise auf dem Frontpanel mit Hilfe eines ActiveX-Containers eine Internetseite anzeigen lassen möchten, greifen Sie auf das Objekt “Microsoft Web Browser” zu, wählen Sie die Methodenklasse “Navigate” und die Methode “URL” aus und geben Sie die entsprechende URL an.

Bei Verwendung eines ActiveX-Containers muss die RefNum des ActiveX-Objekts weder mit der Funktion “ActiveX-Objekt öffnen” erstellt noch mit Hilfe der Funktion “ActiveX-Objekt schließen” geschlossen werden. Da der ActiveX-Container die aufrufende Applikation in LabVIEW einbettet, kann eine direkte Verbindung zu einem Methoden- oder Eigenschaftsknoten erstellt werden. Enthält dieser jedoch Eigenschaften oder Methoden, die andere ActiveX-Objekt-RefNums zurückgeben, müssen diese beendet werden.

Design-Modus für ActiveX-Objekte

Um bei der Bearbeitung eines VIs einen ActiveX-Container im Design-Modus anzuzeigen, klicken Sie den Container mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü **Fortgeschritten» Design-Modus** aus. In diesem Modus werden weder Ereignisse erzeugt noch Vorgänge zur Bearbeitung eines Ereignisses gestartet. Der voreingestellte Modus ist “Ausführung”, das heißt, Sie können mit dem Objekt ebenso umgehen wie der Anwender.

Festlegen von ActiveX-Eigenschaften

Nachdem Sie ein ActiveX-Objekt oder Dokument aufgerufen bzw. eingefügt haben, können Sie zu diesem Eigenschaften festlegen. Verwenden Sie dazu den ActiveX-Eigenschafts-Browser, -Knoten oder die Eigenschaftsseiten.

Browser für ActiveX-Eigenschaften

Über den ActiveX-Eigenschaften-Browser können Sie die Eigenschaften eines Bedienelements oder -Dokuments in einem ActiveX-Container anzeigen lassen bzw. festlegen. Klicken Sie dazu das Objekt im Container mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Eigenschaften-Browser** aus. Statt dessen kann auch **Werkzeuge»Fortgeschritten»ActiveX-Eigenschaften-Browser** ausgewählt werden. Mit Hilfe des ActiveX-Eigenschaften-Browsers können die Eigenschaften eines ActiveX-Objekts auf einfache Weise interaktiv festgelegt werden, was hiermit allerdings nicht programmatisch möglich ist und nur auf Container-Objekte beschränkt ist. Im Ausführungsmodus der Objekte oder während das VI läuft, ist diese Option nicht verfügbar.

ActiveX-Eigenschaftsseiten

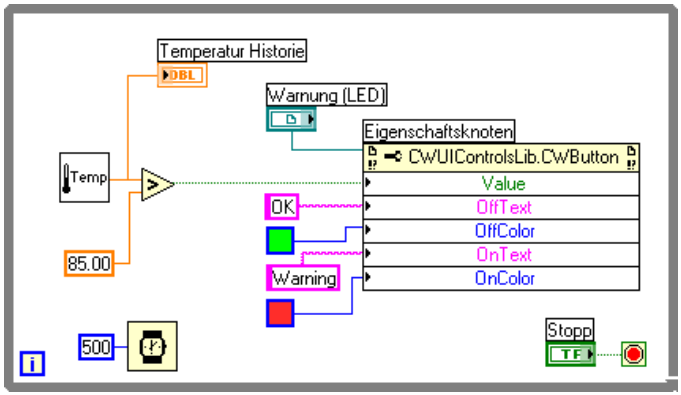
Zu vielen ActiveX-Objekten gibt es Eigenschaftsseiten. Hier sind die Eigenschaften des Objekts in verschiedene Registerkarten unterteilt. Um auf diese Seiten zuzugreifen, klicken Sie das entsprechende Container-Objekt oder -Dokument mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü den Namen des Objekts aus.

Wie mit dem ActiveX-Eigenschaften-Browser können auch hier die Eigenschaften ActiveX-Elements interaktiv festgelegt werden. Jedoch beschränkt sich die Option ebenfalls nur auf Container-Objekte und dient nicht dem programmatischen Setzen von Eigenschaften. Auch stehen nicht zu allen ActiveX-Objekten Eigenschaftsseiten zur Verfügung. Im Ausführungsmodus der Objekte und während der Ausführung des VIs ist die Option deaktiviert.

Eigenschaftsknoten

Um Eigenschaften von ActiveX-Objekten programmatisch festzulegen, verwenden Sie Eigenschaftsknoten. Wenn Sie zum Beispiel ein ActiveX-Element verwenden, um einen Benutzer zu warnen, wenn ein Temperaturlimit überschritten wird, kann der Temperaturhöchstwert über einen Eigenschaftsknoten festgelegt werden.

Im folgenden Beispiel wird die Eigenschaft “Wert” des CWButton-ActiveX-Objekts (Bestandteil der Component Works Control Library) verändert, wenn die Temperatur 85 °F (29,4 °C) erreicht oder übersteigt.



In diesem Fall nimmt der CWButton wie eine LED den Status “Ein” an, ändert seine Farbe und zeigt eine `Warnung` an.

Hinweis Die Eigenschaften OffText, OffColour und OnColour des CWButton sind nicht programmatisch veränderbar und müssten daher über den ActiveX-Eigenschaften-Browser oder die Eigenschaftsseiten festgelegt werden.Weitere Informationen hierzu finden Sie unter *Browser für ActiveX-Eigenschaften* und *ActiveX-Eigenschaftsseiten* in diesem Kapitel.

LabVIEW als ActiveX-Server

Über ActiveX-Aufrufe sind VIs, die LabVIEW-Umgebung sowie Eigenschaften und Methoden von Objekten auch anderen Applikationen zugänglich. Andere ActiveX-fähige Applikationen, wie zum Beispiel Microsoft Excel, können Eigenschaften, Methoden und einzelne VIs von LabVIEW anfordern, wobei LabVIEW als ActiveX-Server fungiert.

So kann beispielsweise ein VI-Graph in eine Excel-Tabelle eingebettet werden, wobei es von Excel aus möglich ist, Werte in die VI-Eingänge einzugeben und das VI zu starten. Beim Ausführen des VIs werden die Werte dann in Form eines Graphen dargestellt.

Ein Beispiel für die Verwendung der LabVIEW-Eigenschaften und -Methoden in einer Excel-Tabelle finden Sie unter `examples\comm\freqresp.xls`.

Unterstützung benutzerdefinierter ActiveX-Schnittstellen

Beim Erstellen eines ActiveX-Clients, der auf Eigenschaften und Methoden eines LabVIEW-ActiveX-Servers zurückgreift, ist es möglich, vom Server erstellte benutzerdefinierte Schnittstellen anzusprechen. Dazu ist kein IDispatch erforderlich. Bei der Erstellung des Servers muss jedoch sichergestellt werden, dass es sich bei den Parametern der Eigenschaften und Methoden der benutzerdefinierten Schnittstelle um Automation-(IDispatch)-Datentypen handelt. Dadurch können mehrere Schnittstellen von einem Objekt anstatt von mehreren Objekten aus freigegeben werden. In der LabVIEW-Umgebung können die Schnittstellen weiterhin verwendet werden. Nähere Hinweise zum Zugriff auf benutzerdefinierte Schnittstellen finden Sie in der Dokumentation zur Programmierumgebung für die Serverentwicklung.

Festlegen von Parametern in ActiveX-VIs mit Hilfe von Konstanten

An einige Parameter in ActiveX-Knoten können separate Listen zulässiger Werte übergeben werden. Um diese Werte festzulegen, wählen Sie in der Ring-Konstante den entsprechenden Namen aus. Um beim Erstellen eines ActiveX-VIs auf die Ring-Konstante zuzugreifen, klicken Sie mit der rechten Maustaste auf den Parameter des Knotens, an den die Datenwerte übergeben werden, und wählen aus dem Kontextmenü die Option **Konstante erzeugen**. Die in der Ring-Konstante verfügbaren Auswahlmöglichkeiten hängen von der an den Knoten übergebenen RefNum ab. In den Abbildungen 19-1 und 19-2 sehen Sie Beispiele für die Verwendung der Ring- und numerischen Konstanten zur Einstellung von Parameterwerten.

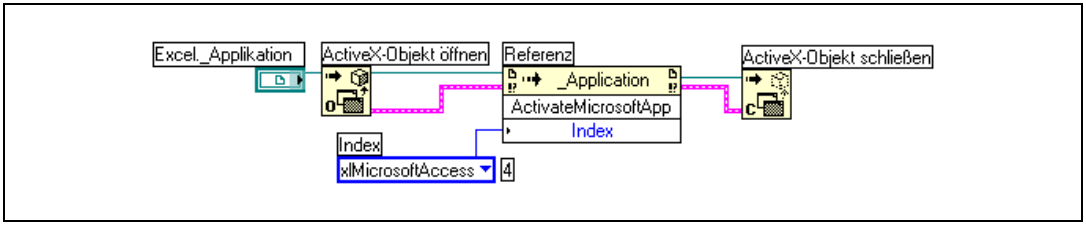


Abbildung 19-1. Festlegen eines Datenwerts mittels einer Ring-Konstante

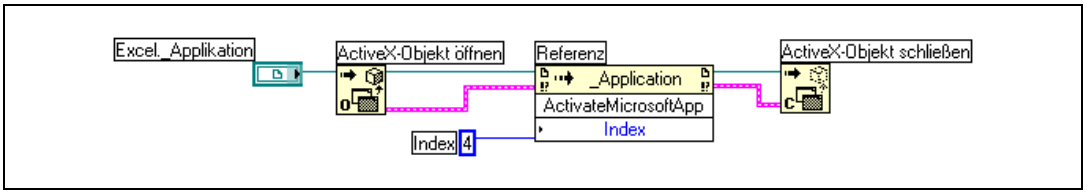


Abbildung 19-2. Festlegen eines Datenwerts mittels einer numerischen Konstante

Bei Parametern, an die Datenwerte übergeben werden, wird links neben dem Namen ein kleiner Pfeil angezeigt. Klicken Sie zur Anzeige des entsprechenden numerischen Datenwerts mit der rechten Maustaste auf die Ring-Konstante und wählen Sie aus dem Kontextmenü **Sichtbare Objekte»Zahlenwertanzeige**.

Mit den VIs in den Abbildungen 19-1 und 19-2 wird jeweils auf Microsoft Excel zugegriffen und eine Methode ausgeführt. Zum Parameter **Index** der Methode **ActivateMicrosoftApp** sind folgende Optionen verfügbar: **MicrosoftWord**, **MicrosoftPowerPoint**, **MicrosoftMail**, **MicrosoftAccess**, **MicrosoftFoxPro**, **MicrosoftProject** und **MicrosoftSchedulePlus**.

Um den numerischen Wert des Parameters **Index** zu ermitteln, welcher der Option **MicrosoftAccess** in Abbildung 19-1 entspricht, wählen Sie aus der Liste in der Ring-Konstante die Option **MicrosoftAccess** aus. Der numerische Wert der aktuell ausgewählten Option wird in einem Feld neben der Ring-Konstante angezeigt. Anstatt eine Ring-Konstante zu verwenden, können Sie den numerischen Wert einer Option in eine numerische Konstante eingeben (siehe Abbildung 19-2).

ActiveX-Ereignisse

Der Einsatz von ActiveX-Ereignissen in Anwendungen erfordert eine Registrierung zu dem Ereignis sowie eine entsprechende Verarbeitung des autretenden Ereignisses. Die Registrierung für ActiveX-Ereignisse ähnelt der dynamischer Ereignisse, welche im Abschnitt *Dynamische Ereignisre-*

gistrierung des Kapitels 9, *Ereignisgesteuerte Programmierung*, erörtert sind. Die VIs zur Verarbeitung von ActiveX-Ereignissen sind jedoch anders aufgebaut als die VIs, die auch in Kapitel 9, *Ereignisgesteuerte Programmierung*, beschrieben sind. Ein VI zur Verarbeitung von ActiveX-Ereignissen ist in der Regel aus folgenden Komponenten aufgebaut:

- Das ActiveX-Objekt, zu dem ein Ereignis erzeugt werden soll.
- Der Knoten “Ereignis-Callback registrieren”, um den Typ des zu erzeugenden Ereignisses anzugeben und zu registrieren.
- Ein Callback-VI, mit dem das Ereignis wie angegeben verarbeitet wird.

Ereignisse sind sowohl für ActiveX-Objekte in Containern als auch für Objekte verfügbar, die über eine ActiveX-RefNum angegeben wurden. So kann zum Beispiel von einem ActiveX-Container aus ein Baumstrukturelement von Windows aufgerufen werden und angegeben werden, dass bei einem Doppelklick auf eines der darin angezeigten Elemente ein Ereignis ausgelöst werden soll.

Der Knoten “Ereignis-Callback registrieren” ist erweiterbar und kann, ähnlich wie der Knoten zur Ereignisregistrierung, mehrere Ereignisse verarbeiten.

Bei ActiveX erfolgt die Definition von Ereignissen, indem eine Referenz auf ein Objekt mit dem Knoten “Ereignis-Callback registrieren” verbunden und das zu erzeugende Ereignis angegeben wird. Anschließend wird ein Callback-VI zur Verarbeitung des Ereignisses programmiert.

Verarbeitung von ActiveX-Ereignissen

Wenn an einem ActiveX-Bedienelement das erwünschte Ereignis auftritt, wird dieses durch ein Callback-VI verarbeitet. Ein solches VI müssen Sie jedoch selbst erstellen. Das VI wird immer ausgeführt, wenn das Ereignis stattfindet. Um ein Callback-VI zu erstellen, klicken Sie mit der rechten Maustaste auf den Eingang **VI Ref** des Knotens “Ereignis-Callback registrieren” und wählen aus dem Kontextmenü die Option **Callback-VI erstellen** aus. Daraufhin wird ein ablaufinvariantes VI erstellt, das folgende Elemente umfasst:

- **Allgemeine Angaben zum Ereignis** mit den Elementen:
  - **Quelle**—Numerisches Bedienelement zur Angabe der Ereignisquelle, wie beispielsweise LabVIEW oder ActiveX. Der Wert `1` steht für ein ActiveX-Ereignis.

  - **Typ**—Gibt an, was für ein Ereignis aufgetreten ist. Bei Ereignissen der Benutzeroberfläche handelt es sich hierbei um eine Enum, und wenn das Ereignis über ActiveX oder eine andere Quelle ausgelöst wurde, um einen vorzeichenlosen 32-Bit-Integer. Der Ereignistyp wird in diesem Fall als Code der Methode (oder ID) des registrierten Ereignisses angegeben.
  - **Zeit** ist der Zeitstempel zur Angabe des Zeitpunkts, zu dem das Ereignis stattgefunden hat. Die Angabe erfolgt in Millisekunden.
- **CtlRef** (eine Referenz auf eine ActiveX-RefNum zum Objekt, an dem das Ereignis stattgefunden hat).
- **Ereignisdaten**—Ist ein Cluster mit den ereignisspezifischen Parametern. Bei Auswahl eines Ereignisses im Knoten “Ereignis-Callback registrieren” werden die entsprechenden **Ereignisdaten** automatisch konfiguriert. Für Hinweise zu Melder- und Filterereignissen lesen Sie bitte Abschnitt *Melder- und Filterereignisse* des Kapitels 9, *Ereignisgesteuerte Programmierung*.
- **Ereignisdaten (Ausgang)**—Ist ein Cluster mit den spezifischen, veränderlichen Parametern des Ereignisses, das vom Callback-VI verarbeitet werden soll. Dieses Element ist jedoch nur bei Filterereignissen verfügbar.
- (Optional) **Benutzer-Parameter**—Sind die Daten, die an das Callback-VI übergeben werden, wenn am ActiveX-Objekt das entsprechende Ereignis auftritt.

Hinweis Als Callback-VI kann auch ein vorhandenes VI verwendet werden, solange das Anschlussfeld des VIs mit dem der Ereignisdaten übereinstimmt. Es wird jedoch empfohlen, ein ablaufinvariantes VI zu verwenden, denn andernfalls kann es nicht mehrmals gleichzeitig aufgerufen werden, wenn Ereignisse mehrfach auftreten.

20

Aufrufen von Code aus textbasierten Programmiersprachen

Die meisten Shared Libraries können über die Funktion “Aufruf externer Bibliotheken” in LabVIEW eingebunden werden. Für den Zugriff auf C-Code sollte ein Code-Interface-Knoten (CIN) verwendet werden.

Das Aufrufen externen Codes auf verschiedenen Plattformen ist in den *LabVIEW Development Guidelines* in Kapitel 6, *LabVIEW Style Guide*, im Abschnitt *Call Library Function Nodes and Code Interface Nodes* beschrieben. Weitere Hinweise zum Aufrufen von Programmcode textbasierter Programmiersprachen erhalten Sie im Handbuch *Using External Code in LabVIEW*.

Weitere Informationen ...

Lesen Sie zu diesem Thema auch die *LabVIEW-Hilfe*.

Knoten zum Aufruf externer Bibliotheken

Die Funktion “Aufruf externer Bibliotheken” ist für die meisten Shared Libraries (DLLs) geeignet. Sie können damit in LabVIEW eine Schnittstelle erstellen, um vorhandene oder neue Bibliotheken aufzurufen, die speziell für die Verwendung mit LabVIEW geschrieben wurden. Es wird empfohlen, zum Erstellen einer Schnittstelle zu externem Code die Funktion zum Aufruf externer Bibliotheken einzusetzen.

Code-Interface-Knoten

Der CIN bietet eine alternative Methode, um in C geschriebenen Quellcode aufzurufen. Die Funktion zum Aufruf externer Bibliotheken ist jedoch in der Regel einfacher zu verwenden als der CIN.

21

Formeln und Gleichungen

Wenn Sie in LabVIEW eine komplexe Formel verwenden möchten, müssen verschiedene arithmetische Funktionen im Blockdiagramm nicht verbunden werden. Statt dessen können Sie die Gleichungen in einer vertrauten mathematischen Umgebung entwickeln und sie anschließend in eine Applikation integrieren.

Mit Formel- und Ausdrucksknoten können Sie in der LabVIEW-Umgebung mathematische Operationen ausführen. Um eine noch größere Anzahl an Funktionen zu erhalten, können Sie auch eine Verknüpfung zur Mathematik-Anwendung MATLAB herstellen.

Weitere Informationen ...

Weitere Informationen zu Gleichungen und der zu verwendenden Syntax, den verfügbaren Funktionen und Operatoren sowie Beschreibungen möglicher Fehler finden Sie in der *LabVIEW-Hilfe*.

Methoden zur Nutzung von Gleichungen in LabVIEW

Für mathematische Operationen im Blockdiagramm stehen Ihnen der Formelknoten, der HiQ-Skript-Knoten und der MATLAB-Skript-Knoten zur Verfügung.

Hinweis MATLAB muss auf Ihrem Computer installiert sein, um den entsprechenden Skript-Knoten verwenden zu können, da für die Ausführung der Skripts der MATLAB-Skript-Server aufgerufen wird. Da LabVIEW zur Implementierung der Skript-Knoten ActiveX-Technologie verwendet, steht diese Funktion nur unter Windows zur Verfügung. Der Skript-Knoten ist mit dem Formelknoten vergleichbar. Allerdings kann damit auch ein vorhandenes HiQ- oder MATLAB-Skript in ASCII-Form importiert und in LabVIEW ausgeführt werden. Wie beim Formelknoten können Daten an den Knoten übergeben und aus dem Knoten entnommen werden.

Formelknoten

Beim Formelknoten handelt es sich um einen praktischen textbasierten Knoten, mit dem Sie im Blockdiagramm mathematische Operationen durchführen können. Sie benötigen keinen Zugriff auf externen Code oder Applikationen, und Sie brauchen keine rudimentären arithmetischen Funktionen zu verbinden, um Gleichungen zu erstellen. Zusätzlich zu textbasierten Gleichungsausdrücken kann der Formelknoten textbasierte Versionen von If-Anweisungen, While-Schleifen, For-Schleifen und Do-Schleifen akzeptieren, die C Programmierern vertraut sind. Diese Programmierelemente sind denen der C-Programmierung ähnlich, jedoch nicht identisch.

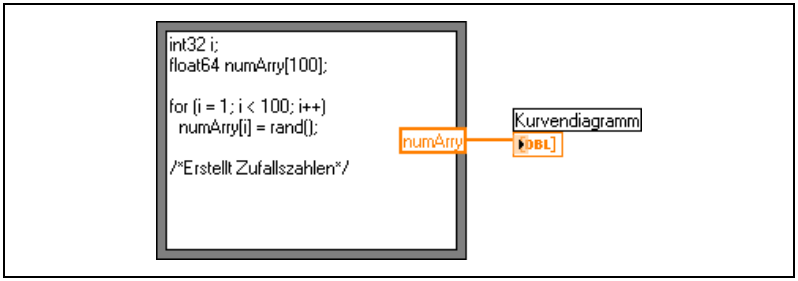
Formelknoten sind nützlich für Gleichungen, die viele Variablen besitzen oder sehr kompliziert sind, sowie für die Nutzung vorhandener textbasierter Codes. Sie können den vorhandenen textbasierten Code kopieren und in einem Formelknoten einfügen, anstatt den Code erneut grafisch zu erstellen.

In Formelknoten ist eine Typprüfung implementiert. Hierdurch wird gewährleistet, dass es sich bei den Array-Indizes um numerische Daten und bei den Operanden für die Bit-Operationen um ganzzahlige Daten handelt. Formelknoten überprüfen auch, ob Array-Indizes ihren Bereich nicht überschreiten. Bei Arrays wird ein Wert außerhalb des Bereichs standardmäßig auf Null gesetzt und eine Zuweisung außerhalb des Bereichs wird standardmäßig auf `nop` gesetzt, um anzuzeigen, dass keine Operation erfolgt ist.

In Formelknoten kann auch eine automatische Typumwandlung durchgeführt werden.

Verwendung des Formelknotens

Ein Formelknoten ist ein in der Größe veränderliches Feld ähnlich einer FOR- oder While-Schleife oder Case- oder Sequenzstruktur. Der Formelknoten enthält jedoch anstelle eines Subdiagramms C-ähnliche Anweisungen, die, wie im folgenden Beispiel gezeigt, durch Semikola voneinander getrennt sind. Wie bei C können Sie Kommentare zwischen einem Schrägstrich/Sternchen-Paar (`/*Kommentar*/`) hinzufügen.



Ein Beispiel für die Verwendung eines Formelknotens ist das VI “Equations” unter `examples\general\structs.llb`.

Variablen im Formelknoten

Bei der Arbeit mit Variablen sind folgende Hinweise zu beachten:

- Die Anzahl der Variablen oder Gleichungen in einem Formelknoten ist unbegrenzt.
- Die Namen von Eingängen beziehungsweise Ausgängen müssen eindeutig sein, ein Ausgang kann jedoch denselben Namen haben wie ein Eingang.
- Um eine Eingangsvariable zu deklarieren, klicken Sie auf den Rand des Formelknotens und wählen Sie aus dem Kontextmenü die Option **Eingang hinzufügen** aus. Innerhalb des Formelknotens können keine Eingangsvariablen deklariert werden.
- Um eine Ausgangsvariable zu deklarieren, klicken Sie auf den Rand des Formelknotens und wählen Sie aus dem Kontextmenü die Option **Ausgang hinzufügen** aus. Die Bezeichnung der Ausgangsvariable muss entweder mit dem einer Eingangsvariable oder einer im Inneren des Formelknotens definierten Variable übereinstimmen.
- Sie können ändern, ob es sich bei einer Variablen um einen Eingang oder einen Ausgang handelt, indem Sie mit der rechten Maustaste auf die Variable klicken und aus dem Kontextmenü **In Eingang umwandeln** beziehungsweise **In Ausgang umwandeln** auswählen.
- Variablen können auch innerhalb des Formelknotens deklariert und verwendet werden, ohne sie einem Eingang oder Ausgang zuzuordnen.
- Sie müssen alle Eingangsanschlüsse verbinden.
- Variablen können skalare Fließkommazahlen sein, deren Genauigkeit von der Konfiguration des jeweiligen Computers abhängt. Sie können auch ganze Zahlen und Arrays aus Zahlen als Variablen verwenden.
- Variablen dürfen keine Einheiten besitzen.

Ausdrucksknoten

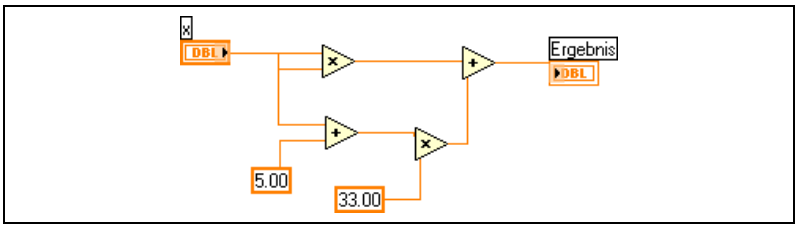
Mit Hilfe von Ausdrucksknoten können Sie Ausdrücke oder Gleichungen berechnen, die eine einzelne Variable enthalten. Ausdrucksknoten sind insbesondere für Gleichungen geeignet, die nur eine Variable enthalten, jedoch anderweitig kompliziert sind.

Ausdrucksknoten verwenden den Wert, den Sie an den Eingangsanschluss übergeben, als Variablenwert. Der Ausgangsanschluss gibt den Wert der Berechnung zurück.

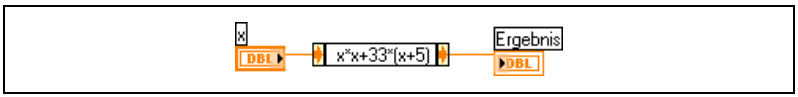
Betrachten Sie zum Beispiel die folgende einfache Gleichung:

$$x \times x + 33 \times (x + 5)$$

Das Blockdiagramm in der folgenden Abbildung verwendet zur Darstellung dieser Gleichung numerische Funktionen.



Mit Hilfe eines Ausdrucksknotens können Sie, wie in der folgenden Abbildung gezeigt, ein wesentlich einfacheres Blockdiagramm erstellen.



Polymorphie in Ausdrucksknoten

Der Eingangsanschluss eines Ausdrucksknotens hat denselben Datentyp wie das Bedienelement oder die Konstante, die Sie mit dem Ausdrucksknoten verbinden. Der Ausgangsanschluss hat den selben Datentyp wie der Eingangsanschluss. Der Datentyp des Eingangs kann eine beliebige nichtkomplexe skalare Zahl, ein beliebiges Array mit nichtkomplexen skalaren Zahlen oder ein beliebiger Cluster mit nichtkomplexen skalaren Zahlen sein. Bei Arrays und Clustern wendet der Ausdrucksknoten die Gleichung auf jedes Element eines Eingangs-Arrays oder -Clusters an.

MATLAB-Skript-Knoten

Mit Hilfe von MATLAB-Skript-Knoten ist es möglich, im Blockdiagramm MATLAB-Skripts zu erstellen, zu laden oder zu bearbeiten.

Dazu muss jedoch MATLAB installiert sein. Wenn Sie bereits über ein in HiQ oder MATLAB geschriebenes Skript verfügen, können Sie es in den Skript-Knoten importieren.

Damit zwischen HiQ bzw. MATLAB und LabVIEW Werte ausgetauscht werden können, müssen im Skript als Ein- oder Ausgänge für die Variablen Anschlüsse für Skript-Knoten zugewiesen werden. Die Funktion eines Anschlusses kann anhand der Schreibweise der Gleichung festgelegt werden. Wenn das Skript beispielsweise die Zuordnungsanweisung $X = i + 3$ enthält, können Sie i einem Eingangsanschluss zuweisen, der steuert, wie der Skript-Knoten X berechnet, und Sie können X einem Ausgangsanschluss zuweisen, um das Endergebnis der Skript-Berechnung abzurufen.

Wenn Sie noch kein Skript geschrieben haben, können Sie in das Blockdiagramm einen Skript-Knoten einfügen und unter Berücksichtigung MATLAB-Syntax ein Skript erstellen. LabVIEW kommuniziert mit dem Skript-Server-Modul. Dabei handelt es sich um das Programm, welches das Skript ausführt. LabVIEW kommuniziert und steuert das Skript-Server-Modul über ein industrieanerkanntes Protokoll. Mit MATLAB wird auch ein Skript-Server installiert.

Aufgrund der Eigenheiten der MATLAB-Skript-Sprache kann der Skript-Knoten nicht den Datentyp der erstellten Anschlüsse ermitteln. Jedem Anschluss des Skript-Knotens muss ein LabVIEW-Datentyp zugewiesen werden. In der Tabelle 21-1 finden Sie eine Übersicht über die MATLAB- und die entsprechenden LabVIEW-Datentypen.

Tabelle 21-1. LabVIEW- und MATLAB-Datentypen

LabVIEW-Datentyp	MATLAB-Datentyp
DBL	Reell
abc	String
	String
[DBL]	Reeller Vektor
[DBL]	Reelle Matrix
CDB	Komplex

Tabelle 21-1. LabVIEW- und MATLAB-Datentypen (Fortsetzung)

LabVIEW-Datentyp	MATLAB-Datentyp
[CDB]	Komplexer Vektor
[CDB]	Komplexe Matrix

Verwenden Sie zur Umwandlung von LabVIEW- in MATLAB-Datentypen die Konvertierungsfunktionen für Zahlen, Strings, Arrays und Pfade.

Programmiervorschläge für MATLAB-Skripts

Die folgenden Programmiertechniken erleichtern die Fehlersuche in einem Skript:

- Schreiben Sie das Skript und führen es in MATLAB für Test- und Fehlersuchzwecke aus, bevor Sie es in LabVIEW importieren.
- Überprüfen Sie die Datentypen. Wenn Sie einen neuen Eingang oder Ausgang erstellen, müssen Sie sicherstellen, dass der Datentyp des Anschlusses richtig ist. Dazu können auch die Funktionen “Fehler (Eingang)” und “Fehler (Ausgang)” verwendet werden.
- Um die Werte, die zwischen LabVIEW und MATLAB ausgetauscht werden, auf Richtigkeit zu überprüfen, erstellen Sie an den Ein- und Ausgängen Bedien- und Anzeigeelemente. Auf diese Weise können Sie gegebenenfalls feststellen, wo ein Skript-Knoten einen Wert falsch berechnet.
- Nutzen Sie die Fehlerparameter zur Anzeige von Informationen für die Fehlersuche. Erstellen Sie ein Anzeigeelemente für den Anschluss **Fehlerausgang** eines Skript-Knotens, damit die Fehlerinformationen zur Laufzeit angezeigt werden. Bei Formelknoten werden Fehler auch beim Kompilieren angezeigt.

A

Die Struktur von LabVIEW

In diesem Anhang wird erörtert, wie das LabVIEW-Dateisystem aufgebaut ist und welche Verzeichnisse zum Speichern bestimmter Dateien vorgesehen sind.

Aufbau der LabVIEW-Verzeichnisstruktur

In diesem Abschnitt ist der Aufbau des LabVIEW-Dateisystems unter Windows, Mac OS und UNIX beschrieben. Die Verzeichnisse für die Treiber-Software für GPIB, DAQ, VISA, IVI, Motion Control und IMAQ-Hardware werden nur angelegt, wenn die entsprechende Komponente für die jeweilige Plattform verfügbar ist. Informationen zur Konfiguration Ihrer Hardware finden Sie auch in Kapitel 3, *Configuring Measurement Hardware*, des *LabVIEW Measurements Manual*.

Bei der Installation von LabVIEW werden im LabVIEW-Programmverzeichnis folgende Unterverzeichnisse angelegt:

Bibliotheken

- `user.lib`—Verzeichnis, in dem benutzerdefinierte Bedienelemente und VIs gespeichert werden können. Selbst erstellte Bedienelemente und VIs sind in LabVIEW über die Paletten **Eigene Elemente** und **Eigene Bibliotheken** abrufbar. Eine Aktualisierung oder Deinstallation von LabVIEW hat keinen Einfluss auf dieses Verzeichnis. Für weitere Hinweise zum Speichern von VIs im Verzeichnis `user.lib` lesen Sie bitte den Abschnitt *Hinzufügen von VIs und Bedienelementen zur Benutzerbibliothek und zur Instrumentenbibliothek* des Kapitels 3, *LabVIEW-Umgebung*.
- `vi.lib`—Enthält Bibliotheken der integrierten VIs, die in LabVIEW auf den Unterpaletten der Palette **Funktionen** angezeigt werden. In dieses Verzeichnis sollten keine selbst erstellten Elemente gespeichert werden, da diese Dateien bei einer Neuinstallation oder Aktualisierung von LabVIEW überschrieben werden.
- `instr.lib`—Enthält Instrumententreiber, mit denen PXI-, VXI-, GPIB-, serielle und computerbasierte Instrumente gesteuert werden. Wenn Sie Instrumententreiber von National Instruments in dieses Ver-

zeichnis installieren, werden diese in LabVIEW zur Unterpalette **Instrumententreiber** hinzugefügt.

Dateien zur Strukturierung und Hilfsdateien

- `menus`—Enthält Dateien, mit denen LabVIEW die Struktur der Paletten **Elemente** und **Funktionen** konfiguriert.
- `resource`—Enthält zusätzliche Hilfsdateien für die LabVIEW-Anwendung. In dieses Verzeichnis sollten keine Dateien gespeichert werden, da LabVIEW diese bei der Installation neuer Versionen überschreibt.
- `project`—Enthält die Dateien für die Objekte im LabVIEW-Menü **Werkzeuge**.
- `templates`—Enthält Vorlagen für allgemein gebräuchliche VIs.
- `www`—Verzeichnis für HTML-Dateien, auf die Sie über den Web-Server zugreifen können.

Beispiele

- `Beispiele`—Enthält Beispiel-VIs, die über **Hilfe»Beispiele suchen** ausgewählt werden können.

Dokumentation

- `manuals`—Enthält die Dokumentation im PDF-Format. Beachten Sie bitte, dass sich die Hilfedateien nicht in diesem Ordner befinden. Um die PDF-Dateien zu öffnen, wählen Sie in LabVIEW die Option **Hilfe»Suchen in der LabVIEW-Bibliothek**.
- `help`—Enthält die Dateien der LabVIEW-Hilfe. Zum Öffnen der *LabVIEW-Hilfe* wählen Sie **Hilfe»VI-, Funktionen- und Anwendungshilfe**.

Mac OS

Zusätzlich zu den oben genannten Dateien ist unter Mac OS ein Ordner mit Shared Libraries verfügbar, in dem Hilfsdateien für die LabVIEW-Anwendung enthalten sind.

Empfehlungen zum Speichern von Dateien

Die Verzeichnisse `vi.lib` und `resource` werden ausschließlich für Zwecke des LabVIEW-Systems installiert. In diesen Ordnern sollten keine Dateien gespeichert werden.

Zum Speichern von Dateien stehen folgende Verzeichnisse zur Verfügung:

- `user.lib`—Für alle Bedienelemente oder VIs, die in den Paletten **Eigene Elemente** und **Eigene Bibliotheken** angezeigt werden sollen. Für weitere Hinweise zum Speichern von VIs im Verzeichnis `user.lib` lesen Sie bitte den Abschnitt *Hinzufügen von VIs und Bedienelementen zur Benutzerbibliothek und zur Instrumentenbibliothek* des Kapitels 3, *LabVIEW-Umgebung*.

Hinweis SubVIs sollten nur dann im Verzeichnis `user.lib` gespeichert werden, wenn sie ohne Änderungen projektübergreifend portiert werden können. Die VI-Pfade in der `user.lib` werden relativ zum Verzeichnis `labview` angegeben. Die Pfadangabe von SubVIs, die in anderen Verzeichnissen gespeichert werden, erfolgt relativ zum übergeordneten VI. Wenn Sie ein VI aus `user.lib` kopieren, um es zu modifizieren, wird daher der Pfad zu den dazugehörigen SubVIs in `user.lib` nicht geändert.

- `instr.lib`—Für Instrumententreiber-VIs, die auf der Palette **Instrumententreiber** angezeigt werden sollen.
- `project`—Für VIs zur Erweiterung von LabVIEW. Die in diesem Verzeichnis gespeicherten VIs werden im Menü **Werkzeuge** angezeigt.
- `www`—Verzeichnis für HTML-Dateien, auf die Sie über den Web-Server zugreifen können.
- `help`—Alle VIs, PDF-Dateien und `.hlp`-Dateien, die im Menü **Hilfe** zur Verfügung gestellt werden sollen.
- `LabVIEW Data`—Von LabVIEW erzeugte Dateien mit Daten wie `.lvm` oder `.txt`-Dateien.

Sie können aber auch an jeder anderen Stelle Verzeichnisse erstellen, um von Ihnen erstellte LabVIEW-Dateien zu speichern. Weitere Informationen zu Clustern finden Sie in Abschnitt *Speichern von VIs* des Kapitels 7, *Erstellen von VIs und SubVIs*.

B

Polymorphe Funktionen

Funktionen sind unterschiedlich polymorph: Es können entweder keine, einige oder alle Eingänge polymorph sein. Einige Funktionseingänge akzeptieren Zahlen oder boolesche Werte. Für einige sind Zahlen oder Strings zulässig. Einige lassen nicht nur skalare Zahlen, sondern auch Zahlen-Arrays, Zahlen-Cluster, Arrays von Zahlen-Clustern und so weiter zu. Einige akzeptieren nur eindimensionale Arrays, wobei die Array-Elemente von jedem Datentyp sein können. Einige Funktionen lassen alle Datentypen zu, komplexe Zahlen eingeschlossen. In den Application Notes *Polymorphic Units in LabVIEW* finden Sie weitere Informationen zu polymorphen Einheiten.

Weitere Informationen ...

Weitere Informationen zu polymorphen Funktionen finden Sie in der *LabVIEW-Hilfe*.

Numerische Konvertierungen

Jede numerische Darstellung kann in jede andere numerische Darstellung konvertiert werden. Wenn zwei oder mehrere numerische Eingänge verschiedener Darstellungen mit einer Funktion verbunden werden, gibt die Funktion normalerweise die Ausgabe in dem größeren oder breiteren Format zurück. Die Funktionen wandeln das Format der kleineren Darstellungen vor der Ausführung in die breiteste Darstellung um, und am entsprechenden Anschluss wird ein Formatumwandlungspunkt angezeigt.

Einige Funktionen, wie zum Beispiel Dividieren, Sinus und Kosinus, erzeugen immer Ausgaben mit Fließkommazahlen. Wenn Sie Ganzzahlen an den Eingängen eingeben, wandeln diese Funktionen die Ganzzahlen vor der Ausführung der Berechnung in Double-Fließkommazahlen um.

Für skalare Mengen mit Fließkommazahlen werden normalerweise am besten Double-Fließkommazahlen verwendet. Single-Fließkommazahlen sparen wenig oder keine Ausführungszeit und laufen sehr viel schneller über. Die Analysebibliotheken verwenden beispielsweise Double-Fließkommazahlen. Extended-Fließkommazahlen sollten nur wenn notwendig

verwendet werden. Die Leistung und Genauigkeit von arithmetischen Operationen mit Extended-Fließkommazahlen hängt von der jeweiligen Plattform ab. Weitere Informationen zum Überlauf von Fließkommazahlen finden Sie in dem Abschnitt *Undefinierte oder unvorhergesehene Daten* des Kapitels 6, *Ausführen von VIs und Fehlersuche*.

Für ganze Zahlen wird normalerweise am besten ein vorzeichenbehafteter 32-Bit-Integer verwendet.

Wenn ein Ausgang mit einem Ziel verbunden wird, das mit einer anderen numerischen Darstellung arbeitet, konvertiert LabVIEW die Daten entsprechend den folgenden Regeln:

- **Ganzzahl mit oder ohne Vorzeichen in Fließkommazahl**—Die Konvertierung erfolgt exakt, es sei denn, Long-Integer werden in Single-Fließkommazahlen umgewandelt. In diesem Fall reduziert LabVIEW die Genauigkeit von 32 Bit auf 24 Bit.
- **Fließkommazahl in Ganzzahl mit oder ohne Vorzeichen**—LabVIEW wandelt Werte außerhalb des Bereichs in den Maximal- oder Minimalwert der Ganzzahl. Die meisten der ganzzahligen Objekte, wie zum Beispiel der Iterationsanschluss einer For-Schleife, runden Fließkommazahlen auf beziehungsweise ab. Bruchteile von 0,5 werden auf die nächste gerade Ganzzahl gerundet. So wird beispielsweise 6,5 auf 6 und nicht auf 7 gerundet.
- **Ganzzahl in Ganzzahl**—LabVIEW wandelt Werte außerhalb des Bereichs nicht in den Maximal- oder Minimalwert der Ganzzahl. Wenn die Quelle kleiner als das Ziel ist, erweitert LabVIEW das Vorzeichen einer vorzeichenbehafteten Quelle und fügt in die überzähligen Bits einer vorzeichenlosen Quelle Nullen ein. Wenn der Wertebereich der Quelle größer ist als der des Ziels, werden nur die niederwertigen Bits des Wertes kopiert.

Polymorphie von numerischen Funktionen

Die Eingangswerte für arithmetische Funktionen müssen numerisch sein. Mit einigen Ausnahmen, auf die bei den Funktionsbeschreibungen hingewiesen wird, besitzt der Ausgang dieselbe numerische Darstellung wie der Eingang. Bei verschiedenen Darstellungen der Eingänge ist die Ausgabe breiter als die Eingabe.

Die arithmetischen Funktionen arbeiten mit Zahlen, Arrays mit Zahlen, Clustern mit Zahlen, Arrays mit Clustern von Zahlen, komplexen Zahlen, und so weiter. Die formale und rekursive Definition des zulässigen Eingabetyps lautet:

> *Numerischer Typ* = numerischer Skalar ODER Array [*numerischer Typ*] ODER Cluster [*numerische Typen*]

Die numerischen Skalare können Fließkommazahlen, Ganzzahlen oder komplexe Fließkommazahlen sein. LabVIEW lässt keine Verwendung von Arrays aus anderen Arrays zu.

Arrays können eine beliebige Anzahl Dimensionen jeder Größe besitzen. Cluster können eine beliebige Anzahl von Elementen besitzen. Der Ausgangstyp von Funktionen besitzt dieselbe numerische Darstellung wie der Eingangstyp. Bei Funktionen mit nur einem Eingang verarbeiten die Funktionen jedes Element des Arrays oder Clusters.

Bei Funktionen mit zwei Eingängen können Sie die folgenden Eingangskombinationen verwenden:

- **Ähnlich**—Beide Eingänge besitzen dieselbe Struktur und der Ausgang hat dieselbe Struktur wie die Eingänge.
- **Ein Skalar**—Ein Eingang ist ein numerischer Skalar, der andere ein Array oder Cluster, und der Ausgang ist ein Array oder Cluster.
- **Array**—Ein Eingang ist ein numerisches Array, der andere der numerische Typ selbst, und der Ausgang ist ein Array.

Bei ähnlichen Eingängen führt LabVIEW die Funktion für die jeweiligen Elemente der Strukturen durch. Beispielsweise werden zwei Arrays Element für Element addiert. Beide Arrays müssen dieselbe Dimension besitzen. Sie können Arrays mit einer unterschiedlichen Anzahl von Elementen addieren. Die Ausgabe einer solchen Addition besitzt die gleiche Anzahl von Elementen wie die kleinste Eingabe. Cluster müssen dieselbe Anzahl von Elementen besitzen, und die jeweiligen Elemente müssen vom selben Typ sein.

Mit der Multiplikationsfunktion kann keine Matrixmultiplikation durchgeführt werden. Wenn Sie die Multiplikationsfunktion für zwei Matrizen verwenden, multipliziert LabVIEW die erste Zahl in der ersten Zeile der ersten Matrix mit der ersten Zahl in der ersten Zeile der zweiten Matrix, und so weiter.

Bei Operationen mit einem Skalar und einem Array oder Cluster führt LabVIEW die Funktion mit dem Skalar und den entsprechenden Elementen der Struktur durch. LabVIEW kann beispielsweise eine Zahl von allen Elementen eines Arrays subtrahieren, unabhängig von der Dimension des Arrays.

Bei Operationen mit einem numerischen Typ und einem Array des betreffenden Typs führt LabVIEW die Funktion für jedes Array-Element durch. Bei einem Graphen handelt es sich beispielsweise um ein Array von Punkten, wobei ein Punkt ein Cluster zweier numerischer Typen, *x* und *y*, bildet. Um einen Graphen 5 Einheiten in *x*-Richtung und 8 Einheiten in *y*-Richtung zu verschieben, können Sie den Punkt (5, 8) zum Graphen addieren.

In Abbildung B-1 sind einige mögliche polymorphe Kombinationen der Additionsfunktion dargestellt.

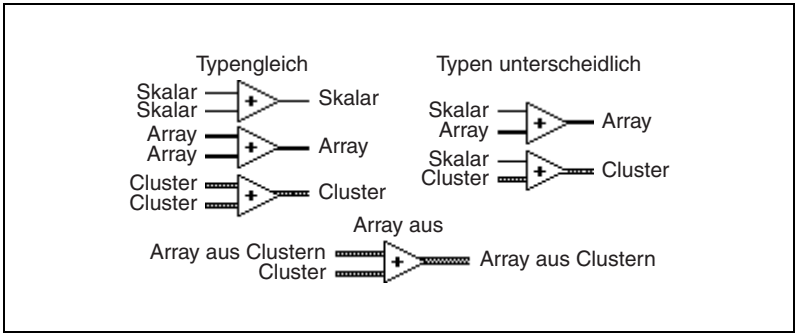


Abbildung B-1. Polymorphe Kombinationen der Additionsfunktion

Polymorphie von booleschen Funktionen

Die logischen Funktionen arbeiten mit booleschen und numerischen Eingangsdaten. Bei einem numerischen Eingang führt LabVIEW die Operation bitweise durch. Wenn der Eingang eine Ganzzahl ist, besitzt der Ausgang dieselbe Darstellung. Ist der Eingang eine Fließkommazahl, rundet LabVIEW die Zahl in eine Long-Integer-Zahl, und der Ausgang ist vom Typ Long Integer.

Die logischen Funktionen arbeiten mit Zahlen oder booleschen Werten, Clustern mit Zahlen oder booleschen Werten, Arrays mit Clustern von Zahlen oder booleschen Werten, und so weiter.

Die formale und rekursive Definition des zulässigen Eingabetyps lautet:

> *Logischer Typ* = boolescher Skalar ODER numerischer Skalar ODER Array [*logischer Typ*] ODER Cluster [*logische Typen*],

mit der Ausnahme, dass komplexe Zahlen und Arrays von Arrays nicht zugelassen sind.

Logische Funktionen mit zwei Eingängen können dieselben Eingangskombinationen wie die arithmetischen Funktionen haben. Die logischen Funktionen unterliegen jedoch der weiteren Beschränkung, dass die Grundoperationen nur zwischen zwei booleschen Werten oder zwei Zahlen erfolgen können. Eine UND-Operation zwischen einem booleschen Wert und einer Zahl ist beispielsweise nicht möglich. In Abbildung B-2 sehen Sie einige Kombinationen boolescher Werte für die UND-Funktion.

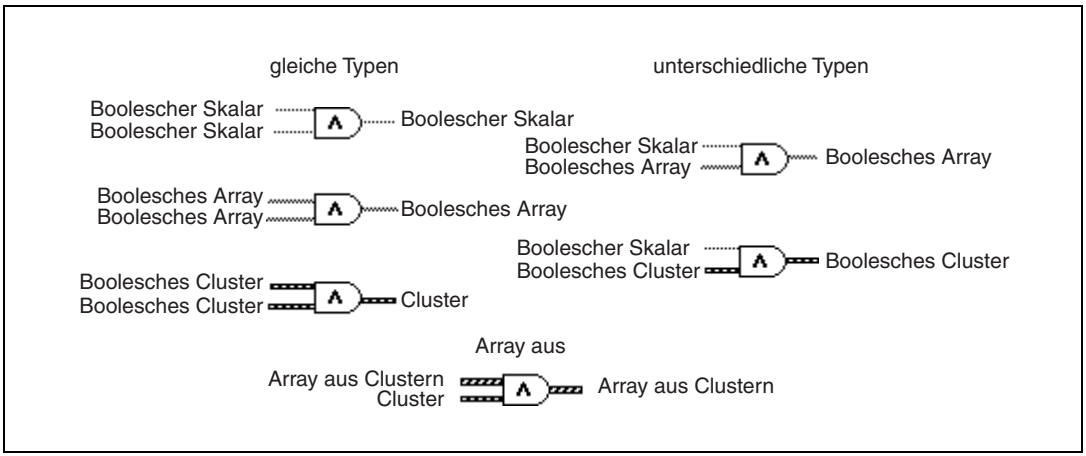


Abbildung B-2. Boolesche Kombinationen für die UND-Funktion

Polymorphie von Array-Funktionen

Die meisten Array-Funktionen konnen *n*-dimensionale Arrays beliebigen Typs verarbeiten. Die Verbindungsdiagramme in den Funktionsbeschreibungen zeigen als Standarddatentyp numerische Arrays.

Polymorphie von String-Funktionen

Die Funktionen “String-Länge”, “In Großbuchstaben”, “In Kleinbuchstaben”, “String umkehren” und “String rotieren” akzeptieren Strings, Cluster und Arrays mit Strings sowie Arrays mit Clustern. “In Großbuchstaben” und “In Kleinbuchstaben” akzeptieren auch Zahlen, Cluster mit Zahlen sowie Arrays mit Zahlen, wobei die Zahlen als ASCII-Zeichen interpretiert werden. Die Eingänge für Breite und Genauigkeit müssen skalar sein.

Polymorphie von String-Konvertierungsfunktionen

Die Funktionen “Pfad in String” und “String in Pfad” sind polymorph. Das heißt, sie arbeiten mit skalaren Werten, Arrays von Skalaren, Clustern von Skalaren, Arrays von Clustern mit Skalaren, und so weiter. Der Ausgang besitzt dieselbe Struktur wie der Eingang, jedoch mit dem neuen Typ.

Polymorphie von zusätzlichen Funktionen zur Konvertierung von Strings in Zahlen

Die Funktionen “Zahl in Dezimal-String”, “Zahl in Hexadezimal-String”, “Zahl in Oktal-String”, “Zahl in String in techn. Notation”, “Zahl in String in gebrochener Notation” und “Zahl in String in Exponentialnotation” können Cluster bzw. Arrays aus numerischen Werten verarbeiten und geben Cluster bzw. Arrays mit Strings aus. Die Funktionen “Dezimal-String in Zahl”, “Hexadezimal-String in Zahl”, “Oktal-String in Zahl”, “Bruch-/Exponentional-String in Zahl” akzeptieren Cluster und Arrays mit Strings und erzeugen Cluster und Arrays mit Zahlen. Die Eingänge für Breite und Genauigkeit müssen skalar sein.

Polymorphie von Cluster-Funktionen

Die Funktionen “Elemente bündeln” und “Aufschlüsseln” zeigen erst dann die Datentypen für die einzelnen Eingangs- oder Ausgangsanschlüsse, wenn Objekte mit den betreffenden Anschlüssen verbunden werden. Wenn Sie diese Anschlüsse verbinden, sehen sie ähnlich wie die Datentypen der entsprechenden Frontpanel-Anschlüsse der Bedien- oder Anzeigeelemente aus.

Polymorphie von Vergleichsfunktionen

Die Funktionen “Gleich?”, “Nicht gleich?” und “Wählen” arbeiten mit Eingaben jeden Typs, solange die Eingaben vom selben Typ sind.

Für die Funktionen “Größer oder gleich?”, “Kleiner oder gleich?”, “Kleiner?”, “Größer?”, “Max & Min” sowie “Wertebereich prüfen und erzwingen” sind alle Datentypen außer komplexen Zahlen, Pfaden oder RefNums zulässig, solange die Eingaben vom selben Typ sind. So können zum Beispiel Zahlen, Strings, boolesche Werte, Arrays aus Strings, Cluster aus Zahlen oder Cluster aus Strings verglichen werden. Dagegen ist zum Beispiel ein Vergleich einer Zahl mit einem String oder eines Strings mit einem booleschen Wert nicht möglich.

Die Funktionen, die Werte mit Null vergleichen, akzeptieren numerische Skalare, Cluster und Arrays aus Zahlen. Diese Funktionen geben boolesche Werte in derselben Datenstruktur wie der Eingang aus.

Die Funktion “Keine Zahl/Pfad/RefNum” akzeptiert dieselben Eingabetypen wie die Funktionen, die Werte mit Null vergleichen. Diese Funktion kann auch Pfade und RefNums verarbeiten. Die Funktion “Keine Zahl/Pfad/RefNum” gibt boolesche Werte in derselben Datenstruktur wie der Eingang aus.

Die Funktionen “Dezimalziffer?”, “Hexadezimalziffer?”, “Oktalziffer?”, “Druckbar?” und “Nicht darstellbare Zeichen?” akzeptieren die Eingabe eines skalaren Strings oder einer Zahl, von Clustern aus Strings oder nichtkomplexen Zahlen, von Arrays aus Strings oder nichtkomplexen Zahlen und so weiter. Die Ausgabe besteht aus booleschen Werten in derselben Datenstruktur wie die Eingabe.

Die Funktion “Leerer String/Pfad?” akzeptiert einen Pfad, einen skalaren String, Cluster aus Strings, Arrays aus Strings und so weiter. Die Ausgabe besteht aus booleschen Werten in derselben Datenstruktur wie die Eingabe.

Die Funktionen “Gleich?”, “Ungleich?”, “Keine Zahl/Pfad/RefNum?”, “Leerer String/Pfad” und “Wählen” sind die einzigen Vergleichsfunktionen, die Pfade und RefNums verarbeiten können.

Bei Vergleichsfunktionen, die Arrays und Cluster verwenden, werden normalerweise boolesche Arrays und Cluster mit derselben Struktur erzeugt. Wenn die Funktion einen einzelnen booleschen Wert ausgeben soll, klicken Sie diese mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Vergleichsmodus»Elementsätze vergleichen**. Weitere Informationen darüber, wie Elementsätze verglichen werden können, finden Sie im Abschnitt *Vergleichen von Arrays und Clustern* des Anhangs C, *Vergleichsfunktionen*.

Polymorphie von logarithmischen Funktionen

Die logarithmischen Funktionen akzeptieren numerische Eingangsdaten. Wenn die Eingabe eine Ganzzahl ist, ist die Ausgabe eine Fließkommazahl mit doppelter Genauigkeit. Andernfalls besitzt der Ausgang dieselbe numerische Darstellung wie der Eingang.

Diese Funktionen arbeiten mit Zahlen, Arrays mit Zahlen, Clustern mit Zahlen, Arrays mit Clustern von Zahlen, komplexen Zahlen, und so weiter. Die formale und rekursive Definition des zulässigen Eingabetyps lautet:

> *Numerischer Typ* = numerischer Skalar ODER Array [*numerischer Typ*] ODER Cluster [*numerische Typen*],

mit der Ausnahme, dass Arrays aus Arrays nicht zugelassen sind.

Arrays können eine beliebige Größe und eine beliebige Anzahl Dimensionen besitzen. Cluster können eine beliebige Anzahl von Elementen besitzen. Der Ausgangstyp besitzt dieselbe numerische Darstellung wie der Eingangstyp, und die Funktionen verarbeiten jedes Element des Clusters oder Arrays. Weitere Informationen zu polymorphen Funktionen mit zwei Eingängen finden Sie im Abschnitt *Polymorphie von numerischen Funktionen* dieses Anhangs. Für logarithmische Funktionen mit zwei Eingängen sind die folgenden Kombinationen von Eingangstypen zulässig:

- **Ähnlich**—Beide Eingänge besitzen dieselbe Struktur und der Ausgang hat dieselbe Struktur wie die Eingänge.
- **Ein Skalar**—Der eine Eingang ist ein numerischer Skalar, der andere ein numerisches Array oder ein numerischer Cluster, und der Ausgang ist ein Array oder Cluster.

C

Vergleichsfunktionen

Die Vergleichsfunktionen eignen sich sowohl zum Vergleichen von booleschen Werten als auch von Strings, numerischen Werten, Arrays und Clustern. Die Mehrzahl der Vergleichsfunktionen überprüft einen Eingang oder vergleicht zwei Eingänge und gibt einen booleschen Wert aus.

Weitere Informationen ...

Weitere Informationen zu Vergleichsfunktionen finden Sie in der *LabVIEW-Hilfe*.

Vergleichen von booleschen Werten

Für die Vergleichsfunktionen ist der boolesche Wert TRUE größer als der boolesche Wert FALSE.

Vergleichen von Strings

LabVIEW vergleicht Strings basierend auf dem numerischen Äquivalent der ASCII-Zeichen. Demnach ist `a` (mit einem Dezimalwert von 97) größer als `A` (65), was wiederum größer als `0` (48) ist, was wiederum größer als das Leerzeichen (32) ist. LabVIEW vergleicht die einzelnen Zeichen am Anfang des String beginnend Zeichen für Zeichen, bis eine Ungleichheit auftritt. An diesem Punkt endet dann der Vergleich. LabVIEW vergleicht beispielsweise die Strings `abcd` und `abef` bis `c` gefunden wird, dessen Wert kleiner ist als `e`. Das Vorhandensein eines Zeichens wird als größer als das Nichtvorhandensein eines Zeichens interpretiert. Der String `abcd` ist also größer als `abc`, weil der erste String länger ist.

Die Funktionen, welche die Kategorie eines Stringzeichens überprüfen (beispielsweise die Funktionen “Dezimalstellen?” und “Druckbar?”), werten nur das erste Zeichen eines Strings aus.

Vergleichen von Zahlen

Die Vergleichsfunktionen konvertieren numerische Werte vor dem Vergleichen in dieselbe Darstellung. Bei Vergleichen mit einem oder zwei Eingängen mit dem Wert `NaN`) wird ein Wert ausgegeben, der Ungleichheit anzeigt. Weitere Informationen zu dem Wert `NaN` finden Sie in dem Abschnitt *Undefinierte oder unvorhergesehene Daten* des Kapitels 6, *Ausführen von VIs und Fehlersuche*.

Vergleichen von Arrays und Clustern

Einige Vergleichsfunktionen verfügen über zwei Modi, um Arrays oder Cluster mit Daten zu vergleichen. Beim Vergleich zweier Arrays oder Cluster im Modus "Elementsätze vergleichen" gibt die Funktion einen einzelnen booleschen Wert aus. Im Modus "Elemente vergleichen" werden die Elemente einzeln miteinander verglichen und es wird ein Array oder ein Cluster mit booleschen Werten ausgegeben.

Im Modus "Elementsätze vergleichen" folgen die String- und Array-Vergleichsoperationen demselben Verfahren – der String wird als Array von ASCII-Zeichen betrachtet.

Um die Betriebsart der Funktion zu verändern, klicken Sie diese mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü die Option **Vergleichsmodus»Elemente vergleichen** oder **Vergleichsmodus» Elementsätze vergleichen**. Einige Vergleichsfunktionen arbeiten nur im Modus "Elementsätze vergleichen", so dass im Kontextmenü keine entsprechenden Optionen angezeigt werden.

Arrays

Beim Vergleichen von mehrdimensionalen Arrays müssen alle in die Funktion eingegebenen Arrays dieselbe Dimension besitzen. Bei Vergleichsfunktionen, die nur über den Modus "Elementsätze vergleichen" verfügen, werden Arrays in derselben Weise wie Strings verglichen: Mit dem ersten Element beginnend wird so lange ein Element nach dem anderen verglichen, bis eine Ungleichheit auftritt.

Modus "Elemente vergleichen"

Im Modus "Elemente vergleichen" wird von den Vergleichsfunktionen ein Array mit booleschen Werten ausgegeben. Dieses hat dieselbe Dimensionenanzahl wie das Eingabe-Array. Jede Dimension des Ausgabe-Arrays

stellt die Größe des kleineren der beiden Eingabe-Arrays in der jeweiligen Dimension dar. Bei jeder Dimension (Zeile, Spalte oder Seite) vergleichen die Funktionen die entsprechenden Elementwerte in allen Eingabe-Arrays, um den entsprechenden booleschen Wert im Ausgabe-Array zu erzeugen.

Modus “Elementsätze vergleichen”

Im Modus “Elementsätze vergleichen” geben die Funktionen nach dem Vergleich der Elemente eines Arrays immer einen einzelnen booleschen Wert aus. Die Reihenfolge der Elemente im Array spielt dabei eine entscheidende Rolle. So werden Elemente, die anderen vorangehen, diesen Elementen gegenüber als vorrangig betrachtet. Die Funktionen gehen daher beim Vergleich folgendermaßen vor:

- Die Funktion vergleicht zugehörige Elemente in jedem Eingabe-Array, beginnend am Anfang des Arrays.
- Wenn die entsprechenden Elemente *nicht* übereinstimmen, hält die Funktion an und gibt das Ergebnis des Vergleichs aus.
- Stimmen die entsprechenden Elemente überein, wird das nächste Wertepaar verarbeitet, bis eine Ungleichheit festgestellt oder das Ende eines Eingabe-Arrays erreicht wird.
- Stimmen alle Werte in den Eingabe-Arrays überein, besitzt ein Array am Ende jedoch weitere Elemente, wird das längere Array als größer angesehen als das kürzere Array. So wird das Array `[1,2,3,2]` beispielsweise als größer erachtet als das Array `[1,2,3]`.

Cluster

Cluster, die Sie verglichen, müssen dieselbe Anzahl Elemente enthalten, die Typen der einzelnen Elemente in den Clustern müssen kompatibel sein und die Elemente müssen in derselben Cluster-Reihenfolge vorkommen. Ein Cluster mit Daten vom Typ DBL und einem String kann beispielsweise mit einem Cluster mit Daten vom Typ I32 und einem String verglichen werden.

Modus “Elemente vergleichen”

Im Modus “Elemente vergleichen” wird von den Vergleichsfunktionen ein Cluster mit booleschen Werten ausgegeben, wobei jedes Cluster-Element dem dazugehörigen Element im Eingabe-Cluster entspricht.

Modus “Elementsätze vergleichen”

Im Modus “Elementsätze vergleichen” wird nur ein einziger boolescher Wert ausgegeben, nachdem die Elemente auf eine Ungleichheit überprüft wurden. Zwei Cluster werden nur dann als gleich betrachtet, wenn alle darin enthaltenen Elemente übereinstimmen.

Mit dem Modus “Elementsätze vergleichen” können Datensätze mit sortierten Daten verglichen werden. Später im Cluster auftretende Elemente werden dabei als untergeordnete Elemente der früher im Cluster vorkommenden Elemente betrachtet. In einem Cluster mit zwei Strings, wie beispielsweise Nachname gefolgt von Vorname, werden die Felder für die Vornamen beispielsweise nur dann verglichen, wenn die Nachnamen übereinstimmen.

D

Technische Unterstützung und professioneller Service

Für professionelle Serviceleistungen und technische Unterstützung lesen Sie bitte auf der Internetseite ni.com von National Instruments unter folgenden Abschnitten nach:

- **Support**—Die Online-Ressourcen zur technischen Unterstützung umfassen:
  - **Ressourcen zur Selbsthilfe**—Für Soforthilfe bei Fragen und Problemen empfiehlt es sich, unsere umfangreiche Sammlung von Ressourcen zum technischen Support unter ni.com/support zu Rate zu ziehen (auf Englisch, Spanisch und Japanisch verfügbar). Diese sind für alle registrierten Benutzer kostenlos und zu den meisten Produkten erhältlich. Hier finden Sie unter anderem Software-Treiber und -Updates, eine Informationsdatenbank (KnowledgeBase), Produkthandbücher, Schritt-für-Schritt-Assistenten zur Problemlösung, Schaltpläne von Geräten, Dokumente über die Einhaltung von Standards, Programmierbeispiele und Lernhilfen, Anwendungshinweise (Application Notes), Instrumententreiber, Diskussionsforen und ein Glossar zur Messtechnik.
  - **Support mit persönlicher Betreuung**—Treten Sie unter ni.com/support mit NI-Ingenieuren und anderen Fachleuten aus dem Bereich Messtechnik und Automatisierung in Verbindung. Über unser System können Sie Fragen stellen und erhalten die Möglichkeit, sich per Telefon, Diskussionsforum oder E-Mail mit den entsprechenden Fachleuten in Verbindung zu setzen.
- **Training**—Unter ni.com/training finden Sie Lernhandbücher, Videos und interaktive CDs. Hier können Sie sich auch für eine der weltweit angebotenen Software-Schulungen anmelden.
- **System-Integration**—Wenn Sie aus Zeit-, Personalmangel oder anderen Gründen bei der Fertigstellung eines Projekts in Verzug geraten, können Ihnen die Mitglieder des NI-Alliance-Programms weiterhelfen. Für Informationen zu diesem Programm setzen Sie sich

entweder telefonisch mit einer National-Instruments-Niederlassung in Ihrer Nähe in Verbindung oder besuchen Sie die Seite ni.com/alliance.

Sollten Sie nach dem Besuch unserer Internetseite ni.com immer noch offene Fragen haben, wenden Sie sich bitte an eine National-Instruments-Niederlassung in Ihrer Nähe. Die Telefonnummern aller Niederlassungen finden Sie vorn in diesem Handbuch. Weitere Kontaktinformationen sowie Telefonnummern für technischen Support, E-Mail-Adressen und Informationen über Ereignisse und Veranstaltungen sind auf der Seite ni.com/niglobal unter "Worldwide Offices" erhältlich.

Glossar

Kurzzeichen	SI-Vorsatz	Faktor
m	Milli	10^{-3}
k	Kilo	10^{3}
M	Mega	10^{6}

Zahlen/Symbole

Δ	Delta; Differenz. Δx bezeichnet den Wert, um den x von einem Index zum nächsten geändert wird.
π	Pi.
∞	Unendlich.
1D	Eindimensional.
2D	Zweidimensional.
3D	Dreidimensional.
3D-Kurve	Spezielle parametrische Kurve ($x(t)$, $y(t)$, $z(t)$), bei der der Parameter t ein gegebenes Intervall durchläuft.

A

A	Ampere.
absolute Koordinaten	Bildkoordinaten relativ zum Ursprung (0, 0) des Koordinatensystems.
absoluter Pfad	Datei- oder Verzeichnispfad, mit dem der Speicherort einer Datei relativ zur höchsten Ebene des Dateisystems beschrieben wird.
Abtastwert	Einzelner Datenwert für Analog-/Digitaleingabe oder -ausgabe.
Abzweigung	Punkt, an dem zwei Verbindungssegmente aufeinander treffen.

AC	Wechselstrom.
aktives Fenster	Das Fenster, auf das sich aktuelle Tastatureingaben beziehen. Dabei handelt es sich in der Regel um das vorderste Fenster. Die Titelleiste des aktiven Fensters ist hervorgehoben. Um ein Fenster in den Vordergrund zu bringen, klicken Sie es an oder wählen Sie es im **Windows**-Menü aus.
aktuelles VI	Das VI, dessen Frontpanel, Blockdiagramm oder Symbol-Editor das aktive Fenster darstellt.
Anschluss	1. Teil des VIs oder Funktionsknotens, der Ein- und Ausgangsanschlüsse enthält. Über die Anschlüsse werden Daten an einen Knoten übergeben und aus dem Knoten übernommen. 2. Objekt oder Bereich eines Knotens zur Ein- oder Ausgabe von Daten.
Anschlussfeld	Bereich oben rechts auf dem Frontpanel oder Blockdiagramm, in dem das Anschlussmuster des VIs angezeigt wird. Hier werden die Ein- und Ausgänge festgelegt, die mit einem VI verbunden werden können.
Anzeigeelement	Frontpanel-Objekt, mit dem Ausgabewerte angezeigt werden, wie zum Beispiel ein Graph oder eine LED.
Applikations-Software	Applikation, die mit Hilfe des LabVIEW-Entwicklungssystems erstellt und im LabVIEW-Laufzeitystem ausgeführt wird.
Array	Geordnete, indizierte Liste aus Datenelementen des gleichen Typs.
ASCII	American Standard Code for Information Interchange.
Auto-Indizierung	Fähigkeit von Schleifenstrukturen, an ihren Begrenzungen Arrays zusammenzuführen und zu trennen. Wenn ein Array mit aktivierter Auto-Indizierung in eine Schleife eintritt, wird es automatisch von der Schleife zerlegt, indem Skalare aus 1D-Arrays, 1D-Arrays aus 2D-Arrays und so weiter extrahiert werden. Schleifen fassen Daten in Arrays zusammen, wenn die Daten eine Schleife wieder verlassen.
automatische Skalierung	Die Fähigkeit von Skalen, sich an den Bereich der dargestellten Werte anzupassen. In Diagrammskalen bestimmt die automatische Skalierung die maximalen und minimalen Skalierungswerte.

B

Bedingungsanschluss	Anschluss einer While-Schleife mit einen booleschen Wert. Der Wert an diesem Anschluss bestimmt, ob die Schleife erneut durchläuft.
Bedienelement	Frontpanel-Objekt, über das Daten interaktiv in ein VI oder programmatisch in ein SubVI eingegeben werden können, wie zum Beispiel ein Drehknopf, eine Taste oder ein Drehregler.
Bedienwerkzeug	Werkzeug zur Eingabe von Daten in Bedienelemente.
Befehlsfluss	Programmierungssystem, in dem die Ausführungsreihenfolge durch die sequentielle Reihenfolge von Anweisungen festgelegt wird. Die meisten textbasierten Programmiersprachen sind befehlsorientierte Sprachen.
benutzerdefinierte Konstante	Blockdiagrammobjekt, das einen vom Benutzer festgelegten Wert enthält.
Bereich	Wird über zwei Grenzwerte definiert, die eine gemessene, empfangene oder gesendete Größe nicht überschreiten darf. Ein Bereich wird anhand eines unteren und eines oberen Grenzwertes angegeben.
Beschriftung	Textobjekt zur Bezeichnung bzw. Beschreibung von Objekten oder Bereichen auf dem Frontpanel oder im Blockdiagramm.
Beschriftungswerkzeug	Werkzeug, mit dem Beschriftungen erstellt werden und Text in Fenster eingegeben wird.
Bibliothek	*Siehe* VI-Bibliothek.
Bildanzeigeelement	Allgemeines Anzeigeelement zur Anzeige von Grafiken, die Linien, Kreise, Text und andere Arten grafischer Formen enthalten können.
Bildlaufwerkzeug	Werkzeug, mit dem ein Bildlauf durch Fenster durchgeführt werden kann.

Blockdiagramm	Grafische Beschreibung oder Darstellung eines Programms oder Algorithmus. Das Blockdiagramm setzt sich aus ausführbaren Knoten, die durch Symbole dargestellt sind, und Verbindungen zusammen, über die Daten zwischen den Knoten übertragen werden. Das Blockdiagramm ist der Quellcode des VIs. Es wird im Blockdiagrammfenster des VIs angezeigt.
boolesche Bedien- und Anzeigeelemente	Frontpanel-Objekte, in denen boolesche Werte (TRUE bzw. FALSE) angezeigt oder verändert werden können.
Bytestream-Datei	Eine Datei, in der Daten als eine Sequenz von ASCII-Zeichen oder Bytes gespeichert werden.

C

Case	Ein Unterdiagramm einer Case-Struktur.
Case-Struktur	Konditionale Struktur zur Abschnittssteuerung, bei der ein Unterdiagramm nur dann ausgeführt wird, wenn am Eingang der Struktur ein bestimmter Wert anliegt. Es handelt sich hier um eine Kombination aus den Befehlen IF, THEN, ELSE und CASE in befehlsorientierten Programmiersprachen.
CIN	*Siehe* Code-Interface-Knoten (CIN).
Cluster	Eine Reihe unindizierter, geordneter Datenelemente beliebiger Datentypen wie numerisch, boolesch, String, Array oder Cluster. Die Elemente müssen entweder alle Bedien- oder alle Anzeigeelemente sein.
Code-Interface-Knoten (CIN)	CIN. Ein spezieller Blockdiagramm-Knoten, über den Programmcode aus textbasierten Programmiersprachen in ein VI eingebunden werden kann.

D

D	Delta; Differenz. Δx bezeichnet den Wert, um den sich x von einem Index zum nächsten ändert.
DAQ	*Siehe* Datenerfassung (DAQ).

DAQ-Kanalassistent	Programm, das Ihnen Hilfestellung bei der Bezeichnung und Konfiguration von DAQ-Kanälen bietet. Steht in der Datenumgebung von Measurement & Automation Explorer **(Windows)** oder im DAQ Channel Wizard **(Mac OS)** zur Verfügung.
Darstellung	Schreibweise eines numerischen Datentyps. Ein numerischer Wert kann als 8-, 16- und 32-Bit-Integer mit oder ohne Vorzeichen, Word-, Long-Integer oder als Single-, Double- oder Extended-Fließkommazahl dargestellt werden.
Datei-RefNum	*Siehe* RefNum.
Datenabhängigkeit	Bedingung in einer datenflussorientierten Programmiersprache, bei der ein Knoten erst dann ausgeführt werden kann, wenn er Daten von einem anderen Knoten empfängt. *Siehe auch* künstliche Datenabhängigkeit.
Datenerfassung (DAQ)	1. Aufnahme und Messung analoger oder digitaler elektrischer Signale über Sensoren, Signalwandler, Messköpfe oder -vorrichtungen. 2. Erzeugung analoger oder digitaler elektrischer Signale.
Datenfluss	Programmiersystem, das aus ausführbaren Knoten besteht, die nur ausgeführt werden, wenn alle erforderlichen Eingangsdaten empfangen wurden. Die Ausgangsdaten werden automatisch bei der Ausführung erzeugt. LabVIEW ist eine datenflussorientierte Programmiersprache.
Datenprotokolldatei	Datei, in der Daten als Sequenz von Datensätzen eines einzigen, beliebigen Datentyps gespeichert werden, den Sie beim Erstellen der Datei angeben müssen. Obwohl alle Datensätze in einer Datenprotokolldatei vom gleichen Typ sein müssen, kann dieser Typ komplex sein. So kann beispielsweise jeder Datensatz ein Cluster sein, der einen String, einen numerischen Wert und ein Array enthält.
Datenprotokollierung	Erfassen und gleichzeitiges Speichern von Daten in eine Datei. Eine Datenprotokollierung ist in LabVIEW beispielsweise mit den VIs und Funktionen zur Datei-I/O möglich.
Datentyp	Format für Informationen. In LabVIEW sind für die meisten VIs und Funktionen die Datentypen “numerisch”, “Array”, “String”, “boolesch”, “Pfad”, “Refnum”, “Enum”, “Signalverlauf” und “Cluster” zulässig.

Diagramm — Zweidimensionale Anzeige einer oder mehrerer Kurven, wobei eine bestimmte Anzahl vorhergehender Werte zwischengespeichert wird. Die Höchstanzahl der zu speichernden Werte ist individuell einstellbar. Die Daten werden in das Diagramm übertragen, und die Anzeige wird Wert für Wert oder Array für Array aktualisiert. Eine bestimmte Anzahl vorhergehender Werte wird für Anzeigezwecke in einem Puffer gespeichert. *Siehe auch* Oszilloskopdiagramm, Streifendiagramm und Laufdiagramm.

diskret — Nicht kontinuierliche Werte einer unabhängigen Variable, in der Regel die Zeit.

Dithering — Hinzufügen eines gaussverteilten Rauschens zu einem analogen Signal. Durch das Anlegen eines solchen Hilfssignals und anschließende Mittelwertbildung der Eingangsdaten kann die Auflösung effektiv um ein halbes Bit gesteigert werden.

DLL — Dynamic Link Library.

E

Eigenschaftsknoten — Setzt oder liest die Eigenschaften eines VIs oder einer Applikation.

eindimensional — Das Vorhandensein einer Dimension, wie bei einem Array mit einer Reihe von Elementen.

Ereignis — Bedingung oder Status eines analogen oder digitalen Signals.

Ereignisdaten-Knoten — Knoten, die sich an der rechten und linken Seite einer Ereignisstruktur befinden. Sie zeigen die verfügbaren Daten für das Ereignis an, das vom entsprechenden Case verarbeitet werden soll. Wenn Sie einen einzelnen Case auf mehrere Ereignisse konfigurieren, stehen nur die Daten zur Verfügung, die für alle Ereignisse gleich sind.

externer Trigger — Spannungsimpuls von einer externen Quelle, der ein Ereignis auslöst, wie zum Beispiel eine A/D-Wandlung.

F

Farbkopierwerkzeug	Kopiert Farben, um diese mit dem Farbwerkzeug einzufügen.
Farbwerkzeug	Werkzeug, mit dem Vorder- und Hintergrundfarben eingestellt werden können.
Fehler (Eingang)	Fehlerstruktur, die an ein VI übergeben wird.
Fehlermeldung	Zeigt Software- oder Hardware-Fehler oder eine unzulässige Dateneingabe an.
Fehler (Ausgang)	Fehlerstruktur, die ein VI verlässt.
Fehlerstruktur	Besteht aus einem booleschen Anzeigeelement “Status”, einem numerischen Anzeigeelement “Code” und einem String-Anzeigeelement “Quelle”.
FIFO	Durchlaufspeicherpuffer (First-In-First-Out). Es werden immer die zuerst gespeicherten Daten an den Empfänger gesendet.
Filterereignisse	Mit Filterereignissen kann das Verhalten der Benutzeroberfläche gesteuert werden.
Filterung	Art der Signalkonditionierung, bei der unerwünschte Signale aus dem zu messenden Signal entfernt werden.
flache Sequenzstruktur	Struktur zur Programmsteuerung, deren Rahmen in numerischer Reihenfolge ausgeführt werden. Mit dieser Struktur werden Knoten, die keiner natürlichen Datenabhängigkeit unterliegen, dazu gezwungen, Vorgänge einer festgelegten Reihenfolge auszuführen. Bei einer flachen Sequenzstruktur sind alle Rahmen sichtbar. Die Ausführung erfolgt von links nach rechts, bis der letzte Rahmen abgearbeitet ist.
FOR-Schleife	Iterative Schleifenstruktur, deren Unterdiagramme so oft wie festgelegt ausführt werden. Entspricht in textbasierten Programmiersprachen dem Befehl: `For i = 0 to n - 1, do...`
Formatumwandlungspunkt	Erscheint an einem Blockdiagrammknoten, um darauf hinzuweisen, dass zwei unterschiedliche Datentypen oder ein Variant und ein beliebiger Datentyp miteinander verbunden wurden.

Formelknoten	Knoten, der Gleichungen ausführt, die als Text eingegeben werden. Formelknoten sind besonders für lange Gleichungen geeignet, deren Umsetzung im Blockdiagramm zu aufwendig wäre.
freie Beschriftung	Beschriftung auf dem Frontpanel oder Blockdiagramm, die keinem Objekt zugeordnet ist.
Frontpanel	Interaktive Benutzeroberfläche eines VIs. Mit Hilfe des Frontpanels werden physikalische Geräte nachgebildet, wie zum Beispiel Oszilloskope oder Multimeter.
Funktion	Integriertes Ausführungselement; vergleichbar mit einem Operator, einer Funktion oder einer Anweisung in einer textbasierten Programmiersprache.

G

gebrochene Schaltfläche **Ausführen**	Die Schaltfläche, die anstelle der Schaltfläche **Ausführen** angezeigt wird, wenn ein VI aufgrund von Fehlern nicht ausgeführt werden kann.
General Purpose Interface Bus	GPIB—Synonym für HPIB. Standardbus zur Steuerung elektronischer Instrumente mit einem Computer. Wird auch IEEE-488-Bus genannt, da er in den ANSI/IEEE-Standards 488-1978, 488.1-1987 und 488.2-1992 definiert ist.
Gerät	Instrument oder Controller, der als einzelne Einheit adressierbar ist und reale I/O-Werte steuert oder überwacht. Ein Gerät wird oftmals über ein Datenübertragungsnetzwerk an einen Host-Computer angeschlossen.
gestapelte Sequenzstruktur	Struktur zur Programmsteuerung, deren Rahmen in numerischer Reihenfolge ausgeführt werden. Mit dieser Struktur werden Knoten, die keiner natürlichen Datenabhängigkeit unterliegen, dazu gezwungen, Vorgänge einer festgelegten Reihenfolge auszuführen. Bei gestapelten Sequenzstrukturen wird immer nur einen Rahmen angezeigt. Die Ausführung erfolgt der Reihe nach bis zum letzten Rahmen.
globale Variable	Greift auf Daten verschiedener VIs in einem Blockdiagramm zu und leitet sie weiter.
GPIB	*Siehe* General Purpose Interface Bus.

Grafik — Bild, das aufgrund bestimmter Anweisungen von einem Bildanzeigeelement erzeugt wird.

Graph — Zweidimensionale Anzeige einer oder mehrerer Kurven. Ein Graph empfängt Daten und stellt diese als Block dar.

Graph-Bedienelement — Frontpanel-Objekt, das Daten in einer kartesischen Ebene anzeigt.

Gruppe — Vom Anwender festgelegte, zusammengefaßte Eingangs- und Ausgangskanäle. Gruppen können analoge und digitale Ein- und Ausgangskanäle oder Counter/Timer-Kanäle enthalten. Eine Gruppe kann jedoch immer nur einen Kanaltyp enthalten. Um nach dem Erstellen auf eine Gruppe Bezug zu nehmen, verwenden Sie eine Task-ID. Es können gleichzeitig bis zu 16 Gruppen definiert werden.

Um eine Gruppe zu löschen, leiten Sie die entsprechende Gruppennummer und ein leeres Kanal-Array an das VI “Gruppenkonfiguration” weiter. Sie müssen eine Gruppe nicht löschen, um deren Zuweisung zu ändern. Wenn Sie eine Gruppe im aktiven Task neu konfigurieren, wird der Task gelöscht und eine Warnung ausgegeben. Nach einer Neukonfiguration der Gruppe erfolgt kein Neustart des Tasks.

H

Haltepunkt — Hält die Ausführung für die Fehlersuche an.

Haltepunkt-Werkzeug — Ein Werkzeug zum Festlegen eines Haltepunkts in einem VI, Knoten oder an einem Verbindungsstück.

Handle — Zeiger auf einen Speicherblock-Zeiger, mit dem Referenz-Arrays und -Strings verwaltet werden. Ein String-Array ist ein Handle für einen Speicherblock, der Handles für Strings beinhaltet.

hex — Hexadezimal. Zahlensystem mit der Basis 16.

Hierarchie-Fenster — Fenster, in dem die Hierarchie von VIs und SubVIs grafisch dargestellt ist.

Highlight-Funktion	Technik zur Fehlersuche, bei der während der VI-Ausführung der Datenfluss durch das VI grafisch dargestellt wird.
Hinweisstreifen	Kleines gelbes Textbanner mit einer Anschlussbezeichnung, um einen Anschluss für den Verbindungsvorgang leichter kenntlich zu machen.

I

I/O	Input/Output (Eingabe/Ausgabe). Die Übertragung von Daten zu oder von einem Computer-System einschließlich Kommunikationskanälen, Eingangsgeräten und/oder Datenerfassungs- und Steuerschnittstellen.
IEEE	Institute for Electrical and Electronic Engineers.
in Stringdaten konvertierte Daten	Daten beliebigen Typs, die in einen String aus Binärwerten konvertiert wurden. Diese Umwandlung wird meist vorgenommen, wenn Daten in eine Datei geschrieben werden sollen.
`Inf`	Numerischer Anzeigewert für die Fließkomma-Darstellung des Wertes Unendlich.
Instrumententreiber	VI zur Steuerung programmierbarer Instrumente.
Integer (ganze Zahl)	Eine beliebige ganze Zahl, deren Wert negativ, null oder positiv sein kann.
Intensitätsgraph	Zweidimensionale Darstellung dreidimensionaler Daten mit Hilfe von Farben.
IP	Internet-Protokoll.
Iterationsanschluss	Anschluss einer FOR- oder While-Schleife, der die aktuelle Anzahl der ausgeführten Schleifendurchläufe enthält.

K

Kanal

1. Physikalischer Kanal—Anschluss oder Pin, an dem ein analoges oder digitales Signal erzeugt oder gemessen wird. Es ist aber auch möglich, dass ein einzelner physikalischer Kanal mehr als einen Anschluss beinhaltet, wie zum Beispiel bei einem digitalen Port mit acht Leitungen oder bei differentiell geschalteten Analogeingabekanälen. Auch ein Zähler kann ein physikalischer Kanal sein, selbst wenn dieser nicht dieselbe Bezeichnung wie der Anschluss hat, an dem mit dem Zähler ein digitales Signal erzeugt oder gemessen wird.
2. Virtueller Kanal—Sammlung von Einstellungen, die eine Bezeichnung, einen physikalischen Kanal, eine Anschlussverbindung (Eingang), die Art der Messung oder Signalerzeugung sowie Skalierungsinformationen enthalten können. Virtuelle NI-DAQmx-Kanäle können außerhalb eines Tasks (global) oder innerhalb eines Tasks (lokal) definiert werden. Während die Konfiguration virtueller Kanäle beim herkömmlichen NI-DAQ-Treiber und in Vorgängerversionen optional ist, muss bei DAQmx zu jeder Messung ein virtueller Kanal konfiguriert werden. Bei NI-DAQmx ist dies nicht mehr nur im Measurement & Automation Explorer, sondern auch in Ihrem Programm möglich, und Kanäle können entweder als Teil eines Tasks oder einzeln konfiguriert werden.
3. Switch—Ein Switch-Kanal steht für einen beliebigen Verbindungspunkt auf einem Switch. Ein solcher Kanal kann je nach Aufbau des Switches aus einem oder mehreren Signaldrähten bestehen (meist zwei oder vier). Switch-Kanäle sind nicht zur Erstellung eines virtuellen Kanals geeignet. Sie dürfen nur mit den Switch-VIs und -Funktionen für NI-DAQmx verwendet werden.

Kanalname

Eindeutiger Name, der im DAQ-Kanalassistenten einer Kanalkonfiguration zugewiesen wird.

Klasse

Kategorie, die Eigenschaften, Methoden und Ereignisse enthält. Klassen sind in einer Hierarchie angeordnet, wobei jede Klasse die Eigenschaften und Methoden der Klasse der übergeordneten Ebene erbt.

Klonen	Kopieren eines Bedienelements oder eines anderen Objektes, durch Klicken auf das Objekt bei gedrückter <Strg>-Taste und Ablegen der Kopie an einer neuen Position. **(Mac OS)** Drücken Sie <Wahltaste>. **(Sun)** Drücken Sie die <Meta>-Taste. **(Linux)** Drücken Sie die <Alt>-Taste. **(UNIX)** Sie können ein Objekt auch mit der mittleren Maustaste anklicken und anschließend die Kopie an eine neue Position ziehen.
Knoten	Programmausführungselement. Knoten entsprechen Anweisungen, Operatoren, Funktionen und Unterprogrammen in textbasierten Programmiersprachen. Dabei kann es sich um Funktionen, Strukturen und SubVIs handeln.
Kompilieren	Verfahren, bei dem der Programmcode einer höheren Programmiersprache in Maschinencode umgewandelt wird. In LabVIEW werden alle VIs automatisch kompiliert, bevor sie nach Erstellung oder einer Änderung ausgeführt werden.
Konfigurationsprogramm	Bezeichnet unter Windows den Measurement & Automation Explorer und unter Mac OS das NI-DAQ-Konfigurationsprogramm.
Konstante	*Siehe auch* Universalkonstante *oder* benutzerdefinierte Konstante.
Kontexthilfe-Fenster	Fenster, in dem die wichtigsten Informationen zu dem Objekt angezeigt werden, über das Sie den Cursor bewegen. Die Kontexthilfe ist zu VIs, Funktionen, Konstanten, Strukturen, Paletten, Eigenschaften, Methoden, Ereignissen und Komponenten von Dialogfeldern verfügbar.
Kontextmenü	Menü, auf das über einen Rechtsklick auf ein Objekt zugegriffen wird. Die angezeigten Menüelemente sind meist objektspezifisch.
Kontrollkästchen	Kleines rechteckiges Kästchen in einem Dialogfeld, das Sie aktivieren oder deaktivieren können. Kontrollkästchen sind in der Regel mit mehreren Optionen verbunden. Es können auch mehrere Kontrollkästchen aktiviert werden.
Konvertierung	Ändern des Typs eines Datenelements.
künstliche Datenabhängigkeit	Bedingung in einer datenflussorientierten Programmiersprache, bei der die Ausführung eines Knotens dann ausgelöst wird, wenn Daten an den Knoten übergeben werden. Die übergebenen Werte sind in diesem Fall von geringerer Bedeutung.

Kurve	Grafische Darstellung eines Daten-Arrays. Die Anzeige kann entweder in Form eines Graphen oder eines Diagramms erfolgen.
Kurvendiagramm	Anzeigeelement zur Darstellung von Daten in einer bestimmten Rate.

L

LabVIEW	Laboratory Virtual Instrument Engineering Workbench. LabVIEW ist eine grafische Programmiersprache, in der zum Erstellen von Programmen Symbole anstelle von Textzeilen verwendet werden.
LabVIEW-Systemzeit	Datum und Zeit, die von LabVIEW als Referenz für die absolute Zeit verwendet werden. Unter der LabVIEW-Systemzeit versteht man den 1. Januar 1904, 00.00 Uhr GMT.
Laufdiagramm	Numerisches Anzeigeelement, dessen Betriebsweise der eines Oszilloskops nachempfunden ist. Im Gegensatz zum Oszilloskopdiagramm werden beim Laufdiagramm die vorher angezeigten Daten nicht gelöscht, sondern es gleitet eine Linie über die Anzeige, um die vorherigen und die neuen Daten voneinander zu trennen.
Laufwerk	Buchstabe zwischen `a` und `z` gefolgt von einem Doppelpunkt (`:`) zur Angabe eines logischen Disketten- oder Festplattenlaufwerks.
LED	Leuchtdiode (Light Emitting Diode).
leeres Array	Array, das null Elemente beinhaltet, aber einen definierten Datentyp aufweist. Beispielsweise ist ein Array mit einem numerischen Bedienelement, dessen Elemente keine definierten Werte enthalten, ein leeres numerisches Array.
Legende	Zu einem Graphen oder Diagramms gehöriges Objekt, mit dem die Namen und Darstellungsarten von Kurven angezeigt werden.
Listenfeld	Ein Feld in einem Dialogfeld, in dem alle verfügbaren Möglichkeiten für eine Aktion aufgelistet werden, beispielsweise eine Liste mit Dateinamen auf der Festplatte.
LLB	VI-Bibliothek.

lokale Sequenz-Variablen — Anschluss zur Weiterleitung von Daten zwischen den Rahmen einer gestapelten Sequenzstruktur.

lokale Variable — Variable, mit der ein Bedien- oder Anzeigeelement auf dem Frontpanel eines VIs ausgelesen oder beschrieben werden kann.

M

Matrix — Zweidimensionales Array mit Zahlen oder mathematischen Elementen, die die Koeffizienten eines linearen Gleichungssystems darstellen.

Measurement & Automation Explorer — Die National-Instruments-Standardumgebung unter Windows für die Konfiguration und Diagnose von Hardware.

Melder-Ereignisse — Mit Hilfe von Melder-Ereignissen wird LabVIEW darüber informiert, dass eine bestimmte Aktion des Benutzers stattgefunden hat, zum Beispiel eine Wertänderung an einem Bedienelement.

Menüleiste — Horizontale Leiste, in der sich die Hauptmenüs einer Applikation befinden. Die Menüzeile befindet sich unter der Titelleiste eines Fensters. Jede Applikation enthält eine spezifische Menüleiste, auch wenn bestimmte Menüs und Befehle in mehreren Applikationen vorkommen.

Methode — Eine Aktion, die ausgeführt wird, wenn ein Objekt eine Meldung empfängt. Eine Methode ist immer einer Klasse zugeordnet.

Multithread-Applikation — Applikation, bei der mehrere verschiedene Threads unabhängig voneinander ausgeführt werden. Auf einem Computer mit mehreren Prozessoren können diese verschiedenen Threads simultan auf verschiedenen Prozessoren ausgeführt werden.

N

`NaN` — Digitaler Anzeigewert für die Fließkommadarstellung des Wertes *Keine Zahl* (*not a number*). Dieser Wert ist in der Regel das Ergebnis einer nicht definierten Operation, wie zum Beispiel log(–1).

nicht ausführbares VI — Ein VI, das aufgrund von Fehlern nicht ausgeführt werden kann. Gekennzeichnet ist ein solches VI durch einen gebrochenen Pfeil auf der Schaltfläche **Ausführen**.

NI-DAQ — Treibersoftware, die zum Lieferumfang aller Datenerfassungsgeräte von National Instruments gehört. Der NI-DAQ-Treiber ist eine umfassende Bibliothek von VIs und Funktionen, die aus einer Entwicklungsumgebung für Applikationen wie LabVIEW aufgerufen werden kann. Der Treiber dient zur Programmierung aller Komponenten eines Datenerfassungsgerätes von National Instruments und ermöglicht die Konfiguration des Geräts, die Datenerfassung sowie Erzeugung und Empfang von Daten.

NI-DAQmx — Neuer NI-DAQ-Treiber, der mit neuen VIs, Funktionen und Entwicklungswerkzeugen zur Steuerung von Datenerfassungsgeräten arbeitet. Im Vergleich zu früheren Versionen von NI-DAQ-Treibern bietet NI-DAQmx einen integrierten DAQ-Assistenten zur Konfiguration von Tasks, Kanälen und Skalierungen, eine höhere Leistungsfähigkeit, eine größere Anzahl von Funktionen sowie eine einfachere Schnittstelle zur Anwendungsentwicklung.

Numerische Bedien- und Anzeigeelemente — Frontpanel-Objekte zur Bearbeitung und Anzeige numerischer Daten.

O

Objekt — Generische Bezeichnung für die Elemente auf dem Frontpanel oder im Blockdiagramm. Zu den Objekten zählen zum Beispiel Bedien- und Anzeigeelemente, Knoten, Verbindungsstücke oder importierte Bilder.

OLE — Object Linking and Embedding (Software-Standard von Microsoft, mit dem Daten anderer Programme in ein Dokument integriert werden können).

Oszilloskopdiagramm — Numerisches Anzeigeelement, das einem Oszilloskop nachempfunden ist.

P

Palette	Anzeige von Symbolen, die mögliche Optionen darstellen.
Palette **Elemente**	Palette, die Bedien-, Anzeige- und Gestaltungselemente für das Frontpanel enthält.
Palette **Funktionen**	Palette, die VIs, Funktionen, Blockdiagrammstrukturen und Konstanten enthält.
Palette **Werkzeuge**	Enthält Werkzeuge zum Bearbeiten von Frontpanel- und Blockdiagrammobjekten und für die Fehlersuche.
Panel-Fenster	VI-Fenster, in dem das Frontpanel, die Symbolleiste sowie das Symbol- und Anschlussfeld enthalten sind.
Pixel	Kleinste Einheit eines digitalisierten Bildes.
Pixmap	Standardformat zur Speicherung einer Grafik, bei jedes Pixel durch einen Farbwert darstellt wird. Die Schwarz/Weiß-Version einer Pixmap bezeichnet man als Bitmap.
Polymorphie	Fähigkeit eines Knoten, sich automatisch auf Daten unterschiedlicher Darstellungsformen, Typen und Strukturen einzustellen.
Positionierwerkzeug	Werkzeug zum Verschieben und zur Größenänderung von Objekten.
PPC	Kommunikation zwischen Programmen (Program-to-Program Communication).
Puffer	Temporärer Speicher für erfasste oder erzeugte Daten.
Pulldown-Menüs	Menüs, auf die in der Menüleiste zugegriffen werden kann. Pulldown-Menüs enthalten in der Regel allgemeine Menüoptionen.
Punkt	Cluster mit zwei 16-Bit-Integern, die eine horizontale und eine vertikale Koordinate darstellen.
PXI	PCI eXtensions for Instrumentation. Eine modulare, computerbasierte Instrumentenplattform.

R

Race Condition	Tritt auf, wenn zwei oder mehrere Teile eines Programmcodes, die parallel ausgeführt werden, den Wert einer gemeinsam genutzten Ressource ändern; in der Regel eine globale oder lokale Variable.
Rahmen	Unterdiagramm einer flachen oder gestapelten Sequenzstruktur.
Rechteck	Cluster, der vier 16-Bit-Integer enthält. Die ersten beiden Werte beschreiben die vertikale und die horizontale Koordinate der oberen linken Ecke. Die letzten beiden Werte beschreiben die vertikale und horizontale Koordinate der unteren rechten Ecke.
RefNum	Referenz. Ein Bezeichner, den LabVIEW einer geöffneten Datei zuordnet. Verwenden Sie Refnums, um anzugeben, dass eine Funktion oder ein VI eine Operation in der offenen Datei ausführen soll.
relative Koordinaten	Bildkoordinaten, deren Angabe relativ zur aktuellen Cursor-Position erfolgt.
Ring-Bedienelement	Spezielles numerisches Bedienelement, bei dem Texte oder Grafiken 32-Bit-Integern zuordnet werden, wobei mit 0 begonnen und der Wert fortlaufend inkrementiert wird.

S

Skala	Teil eines Diagramms, Graphen und einiger numerischer Bedien- und Anzeigeelemente. Die Skala enthält in bestimmten Abständen Markierungen zur Darstellung von Messeinheiten.
Skalar	Zahl, die durch einen Punkt auf einer Skala dargestellt werden kann. Im Gegensatz zu einem Array ist ein Skalar immer ein einzelner Wert. Boolesche Skalarwerte und Cluster sind explizit einmalige Instanzen ihrer jeweiligen Datentypen.
SCXI	Signal Conditioning eXtensions for Instrumentation. Die National-Instruments-Produktreihe zur Aufbereitung von Signalen mit geringem Pegel in einem externen Gehäuse in Sensorennähe, so dass in einer Umgebung mit Störsignalen nur Signale mit hohen Pegeln an die DAQ-Geräte gesendet werden.
Sequenzstruktur	*Siehe auch* flache Sequenzstruktur *oder* gestapelte Sequenzstruktur.

Shared Library (gemeinsame Bibliothek)	Datei mit ausführbaren Programm-Modulen, deren Funktionen von anderen Computerprogrammen genutzt werden können. Die Verwendung von Shared Libraries bietet sich an, wenn Sie bestimmte Funktionen anderen Anwendern zugänglich machen möchten.
Schieber	Der bewegliche Teil von Schieberegler-Bedien- und Anzeigeelementen.
Schieberegister	Mechanismus in Schleifenstrukturen, mit dem Variablenwerte von einem Schleifendurchlauf an den nächsten übergeben werden können. Schieberegister verhalten sich ähnlich wie statische Variablen in textbasierten Programmiersprachen.
Signalverlauf	Mehrere Spannungswerte, die mit einer spezifischen Abtastrate erfasst wurden.
Sonde	Fehlersuchfunktion, mit der Hilfe in einem VI Zwischenwerte angezeigt werden können.
Sonden-Werkzeug	Werkzeug zur Anzeige von Zwischenwerten an Verbindungen.
Speicherpuffer	*Siehe* Puffer.
Streifendiagramm	Anzeigeelement zur grafischen Darstellung numerischer Daten ähnlich einem XT-Schreiber, bei dem sich während der Anzeige der Daten die Zeitachse verschiebt.
String	Darstellung eines Wertes als Text.
String-Bedien- und -Anzeigeelemente	Frontpanel-Objekte zur Bearbeitung und Anzeige von Text.
Struktur	Programmsteuerelement, wie zum Beispiel eine Sequenz-Struktur, Case-Struktur, FOR- oder While-Schleife.
SubVI	VI, das im Blockdiagramm eines anderen VIs verwendet wird. Ein SubVI ist mit einem Unterprogramm vergleichbar.
Symbol	Grafische Darstellung eines Knotens in einem Blockdiagramm.

Symbolleiste — Leiste, die Befehlsschaltflächen zum Ausführen von VIs und zur Fehlersuche enthält.

Syntax — Ein Regelsatz, dem die Anweisungen einer Programmiersprache entsprechen müssen.

T

TCP/IP — Transmission Control Protokoll/Internet Protokoll. Standardformat zur Übertragung von Datenpaketen von einem Rechner zum anderen. Dabei wird TCP bei der Erstellung der Datenpakete verwendet und IP bei der Übertragung.

Treiber — Software, die nur für ein bestimmtes Gerät oder einen bestimmten Gerätetyp gilt. Ein Treiber enthält alle Befehle, die für das entsprechende Gerät zulässig sind.

Tunnel — Dateneingangs- oder -ausgangsanschluss in einer Struktur.

Typdefinition — Master-Kopie einese benutzerdefinierten Objektes, das von vielen VIs verwendet werden kann.

Typumwandlung — Automatische Umwandlung der Darstellungsweise numerischer Daten in LabVIEW.

U

UDP — User Datagram Protocol.

Universalkonstante — Nicht editierbares Blockdiagramm-Objekt, das ein bestimmtes ASCII-Zeichen oder eine numerische Standardkonstante ausgibt, wie zum Beispiel π.

Unterdiagramm — Blockdiagrammabschnitt, der innerhalb einer Struktur liegt.

URL — Uniform Resource Locator. Eine logische Adresse einer Ressource auf einem Server, in der Regel im Web. Zum Beispiel ist `http://www.ni.com/` die URL für die Internetseite von National Instruments.

V

Vektor	1D-Array.
Verbindung	Datenpfad zwischen Knoten.
Verbindungsknotenpunkt	Punkt, in dem drei oder mehr Verbindungssegmente zusammenlaufen.
Verbindungssegment	Einzelnes horizontales oder vertikales Verbindungsstück.
Verbindungsstrichleitung	Abgeschnittene Leitung, die an einem unverbundenen VI- oder Funktionssymbol angezeigt wird, wenn Sie das Verbindungswerkzeug über das Symbol bewegen.
Verbindungswerkzeug	Werkzeug, mit dem Datenpfade zwischen Anschlüssen definiert werden.
Verbindungszweig	Abschnitt einer Verbindung, der alle Verbindungssegmente von Verbindungspunkt zu Verbindungspunkt, Anschluss zu Verbindungspunkt oder Anschluss zu Anschluss enthält, sofern keine Verbindungspunkte dazwischen liegen.
Verzeichnis	Struktur, mit der sich Dateien zweckmäßig in Gruppen unterteilen lassen. Ein Verzeichnis ist mit einer Adresse vergleichbar, die den Speicherort der Dateien anzeigt. Verzeichnisse können Dateien oder Unterverzeichnisse mit Dateien enthalten.
VI	*Siehe* virtuelles Instrument (VI).
VI der höchsten Programmebene	Das oberste VI einer VI-Hierarchie. Diese Bezeichnung dient zur Unterscheidung des VIs von seinen SubVIs.
VI-Klasse	Referenz auf ein virtuelles Instrument, mit der auf die Eigenschaften und Methoden eines VIs zugegriffen werden kann.
VI-Bibliothek	Spezielle Datei, die eine Sammlung von VIs für einen bestimmten Zweck enthält.
VI-Server	Mechanismus zur programmatischen Steuerung von VIs und LabVIEW-Applikationen (lokal und über Netzwerk).
virtuelles Instrument (VI)	Programm in LabVIEW, mit dem das Aussehen und die Funktion eines physikalischen Instruments nachgebildet wird.

VISA	Virtual Instrument Software Architecture. Bibliothek mit Schnittstelle zur Steuerung von GPIB-, VXI-, RS-232-Geräten und anderen Instrumententypen.
VISA	*Siehe* VISA.
Voreinstellung	Standardwert. Wird bei vielen VI-Eingängen verwendet, wenn Sie keinen Wert angeben.
VXI	VME eXtensions for Instrumentation (Bus).

W

Werkzeug	Spezieller Cursor zur Ausführung bestimmter Operationen.
While-Schleife	Schleifenstruktur, mit der ein bestimmter Blockdiagrammabschnitt so lange wiederholt wird, bis eine bestimmte Bedingung erfüllt ist.

Z

Zählanschluss	Anschluss für eine FOR-Schleife, dessen Wert bestimmt, wie oft das Subdiagramm der Schleife ausführt wird.
Ziehen	Verwenden des Cursors auf dem Bildschirm, um Objekte auszuwählen, zu verschieben, zu kopieren oder zu löschen.
Ziehkreise oder -punkte	Punkte, die an den Rändern eines Objektes erscheinen, und mit denen Sie dessen Größe anpassen können.
zweidimensional	Das Vorhandensein von zwei Dimensionen, wie zum Beispiel bei einem Array, das mehrere Zeilen und Spalten enthält.

Stichwortverzeichnis

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

Y

Z