LabVIEW™

Funktionen- und VI-Referenzhandbuch

NATIONAL INSTRUMENTS™

Ausgabe Juli 1998
Artikelnummer 321988A-01

Deutschsprechende Zweigstellen

National Instruments Germany GmbH Konrad-Celtis-Straße 79 D-81369 München	National Instruments Ges.m.b.H. Plainbachstraße 12 A-5101 Salzburg-Bergheim	National Instruments Switzerland Sonnenbergstraße 53 CH-5408 Ennetbaden

Internet-Unterstützung

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

Mailbox Unterstützung–Bulletin Board Support (BBS)

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

Telefon–Unterstützung

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

Internationale Zweigstellen

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

National Instruments - Firmenhauptsitz

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

Wichtige Informationen

Garantie

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

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

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

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

Copyright

Warenzeichen

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

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

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

Inhaltsverzeichnis

Über dieses Handbuch

Kapitel 1
Einführung in die G-Funktionen und in die VIs

TEIL I
G-Funktionen und VIs

Kapitel 2
Nachschlage-Übersicht über Funktionen von G und VIs

Kapitel 3
Strukturen

Kapitel 4
Numerische Funktionen

Kapitel 5
Boolesche Funktionen

Kapitel 6
String-Funktionen

Kapitel 7
Array-Funktionen

Kapitel 8
Cluster-Funktionen

Kapitel 9
Vergleichsfunktionen

Kapitel 10
Zeit-, Dialog- und Fehlerfunktionen

Kapitel 11
Dateifunktionen

Kapitel 12
Funktionen zur Anwendungssteuerung

Kapitel 13
Fortgeschrittene Funktionen

Teil II
Datenerfassungs-VIs

Kapitel 14
Einführung zu den LabVIEW Datenerfassungs-VIs

Kapitel 15
Einfache Analogeingangs-VIs

Kapitel 16
Mittlere Analogeingangs-VIs

Kapitel 25
Fortgeschrittene Digital-I/O-VIs

Kapitel 26
Einfache-Counter-VIs

Kapitel 27
Mittlere Counter-VIs

Kapitel 28
Fortgeschrittene Counter-VIs

Kapitel 29
Kalibrierungs- und Konfigurations-VIs

Kapitel 30
Signalkonditionierungs-VIs

Teil III
Instrumenten-I/O-Funktionen und -VIs

Kapitel 31
Einführung in LabVIEW Geräte-I/O-VIs

Kapitel 32
Vorlage-VIs für Instrumententreiber

Kapitel 33
VISA-Referenzbibliothek

Kapitel 34
Traditionelle GPIB-Funktionen

Kapitel 35
GPIB-488.2-Funktionen

Kapitel 36
Serielle Anschluß-VIs

Teil IV
Analyse-VIs

Kapitel 37
Einführung in die Analyse unter LabVIEW

Kapitel 38
Signalerzeugungs-VIs

Teil V
Kommunikations-VIs und Funktionen

Kapitel 48
TCP-VIs

Kapitel 49
UDP-VIs

Kapitel 50
DDE-VIs

Kapitel 51
ActiveX-Automationsfunktionen

Kapitel 52
AppleEvent-VIs

Anhang B
DAQ-Hardware-Leistungsfähigkeit

Anhang C
GPIB mehrzeilige Schnittstellen-Nachrichten

Anhang D
Kundenbetreuung

Stichwörterverzeichnis

Abbildungen

Tabellen

Über dieses Handbuch

Das *LabVIEW Funktionen- und VI-Referenzhandbuch* beschreibt alle virtuellen Instrumente (VIs) und deren Funktionen, einschließlich der folgenden:

- VIs zur Unterstützung von Geräten zur Datenerfassung
- VIs für GPIB, VXI-Bus und seriellen Anschluß-I/O-Operationen
- digitale Signalverarbeitung, -filterung und numerische und statistische VIs
- VIs zur Netzwerkkommunikation und Kommunikation zwischen den Anwendungen

Dieses Handbuch stellt eine Ergänzung zum *LabVIEW-Benutzerhandbuch* dar, mit dem Sie vertraut sein sollten.

Dieses Handbuch gibt einen Überblick über alle der verfügbaren Funktionen und VIs des Development Systems von LabVIEW. Um jedoch spezifischere Informationen über Parameter für jede Funktion und jedes VI zu erhalten, lesen Sie bitte die Online-Referenz (die über das Menü **Hilfe»Online-Referenz** aufgerufen werden kann) oder im Hilfefenster (das über das Menü **Hilfe»Hilfe anzeigen** aufgerufen werden kann).

Gliederung dieses Handbuchs

Dieses Handbuch behandelt die folgenden fünf Themenbereiche: Funktionen und VIs von G, VIs zur Datenerfassung, Geräte-I/O-VIs, VIs zur Analyse und VIs zur Kommunikation. Kapitel 1, *Einführung in die G-Funktionen und in die VIs*, stellt die verfügbaren Funktionen und VIs des Development Systems von LabVIEW vor.

- Teil I, *G-Funktionen und VIs*, umfaßt die Kapitel 2–13, in denen die einzigartigen Funktionen der Programmiersprache G beschrieben sind.
- Teil II, *Datenerfassungs-VIs*, umfaßt Kapitel 14–30, in denen VIs zur Datenerfassung (DAQ) beschrieben sind.
- Teil III, *Instrumenten-I/O-Funktionen und -VIs*, umfaßt Kapitel 31–36, in denen die Geräte-I/O-VIs und -Funktionen beschrieben sind.

- Teil IV, *Analyse-VIs*, umfaßt Kapitel 37–47, in denen die VIs zur Analyse beschrieben sind.
- Teil V, *Kommunikations-VIs und Funktionen*, umfaßt Kapitel 49–53, in denen die VIs zur Kommunikation beschrieben sind.

Darüber hinaus enthält dieses Handbuch die folgenden Anhänge und ein Stichwortverzeichnis:

- Anhang A, *Fehlercodes*, umfaßt Tabellen, in denen die Analog-I/O- und Digital-I/O-Fähigkeiten von Datenerfassungsgeräten von National Instruments zusammengefaßt sind.
- Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, listet von IEEE 488 definierte Befehle auf.
- Anhang C, *GPIB mehrzeilige Schnittstellen-Nachrichten*, beschreibt grundlegende Konzepte, die zum Betrieb des GIPB benötigt werden.
- Anhang D, *Kundenbetreuung*, enthält Formulare, mit denen Sie Angaben sammeln können, die uns beim Lösen Ihrer technischen Probleme helfen werden, sowie ein Formular, mit dem Sie uns Ihre Meinung von der Produktdokumentation zukommen lassen können.
- Das *Stichwörterverzeichnis* enthält eine alphabetische Aufstellung von in diesem Handbuch beschriebenen VIs, nebst Seitennummern, wo die VIs gefunden werden können.

Schreibkonventionen in diesem Handbuch

In diesem Handbuch werden die folgenden Schreibkonventionen verwendet:

<>	In spitzen Klammern stehen Tastenbezeichnungen, wie z.B. `<Umschalt>`. Spitze Klammern mit Zahlen, die durch eine Ellipse getrennt sind, geben einen Wertebereich an, der mit einem Bit- oder Signalnamen verknüpft ist z.B., `DBIO<3..0>`.
[]	Eckige Klammern umschließen optionale Elemente, z.B., [`Antwort`].
-	Ein Bindestrich zwischen zwei oder mehreren Tastenbezeichnungen, die innerhalb spitzer Klammern stehen, bedeutet, daß die Tasten gleichzeitig zu drücken sind. z.B., `<Strg-Alt-Entf>`.

»	Das Symbol **»** führt Sie durch geschachtelte Menüpunkte und Dialogfelder zu einer Zielaufgabe. Die Folge **Datei»Seite einrichten»Optionen» Schriftart ersetzen** weist Sie an, das Menü **Datei** herunterzurollen, den Punkt **Seite einrichten** auszuwählen**, Optionen** zu wählen **und** schließlich in dem letzten Dialogfeld die Option **Schriftart ersetzen** auszuwählen**.**
Fettdruck	Fettgedruckter Text bezeichnet die Menünamen, Menüpunkte, Parameter, Dialogfelder, Schaltflächen oder Optionen von Dialogfeldern, Icons, Fenstern, Register von Windows 95 oder LEDs.
kursiver Fettdruck	Fettgedruckter, kursiver Text bezeichnet das Ziel einer Aktivität, einen Hinweis, einen Vorsichtshinweis oder eine Warnung.
Strg	Bezeichnungen von Steuertasten erscheinen mit einem Großbuchstaben am Anfang.
kursiv	Kursiver Text dient zur Hervorhebung und bezeichnet Variablen, Querverweise oder die Einführung eines Schlüsselkonzepts. Daneben bezeichnet diese Schreibweise auch Text, von dem sie das angemessene Wort oder den angemessenen Wert, wie unter Windows 3.x, bereitstellen.
kursiv monospace	Kursiver Text dieser Schriftart bedeutet, daß die angemessenen Worte und Werte anstelle dieser Objekte bereitgestellt werden müssen.
monospace	Text in dieser Schriftart sollte von Ihnen wörtlich über die Tastatur eingegeben werden, wie Abschnitte von Codes, Programmierbeispiele und Syntaxelemente. Dieser Schriftsatz wird außerdem für die Bezeichnungen von Diskettenlaufwerken, Pfaden, Verzeichnissen, Programmen, Unterprogrammen, Subroutinen, Gerätenamen, Operationen, Variablen, Dateinamen und -erweiterungen und für Äußerungen und Kommentare von Programmen eingesetzt.
fettgedruckt monospace	Fettgedruckter Text in diesem Schriftsatz bezeichnet die vom Computer automatisch auf dem Bildschirm ausgegebenen Meldungen und Antworten. Dieser Schriftsatz hebt ebenso Befehlszeilen hervor, die sich von anderen Beispielen unterscheiden.
Pfade	Bei Pfadangaben in diesem Handbuch werden die Laufwerknamen, Verzeichnisse, Ordner und Dateien mit Hilfe eines Backslashes voneinander getrennt.

Verwandte Dokumentationsmaterialien

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

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

Mitteilungen von Kunden

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

1

Einführung in die G-Funktionen und in die VIs

Dieses Kapitel enthält grundlegende Informationen über die im LabVIEW Development System verfügbaren Funktionen und virtuellen Instrumente (VIs).

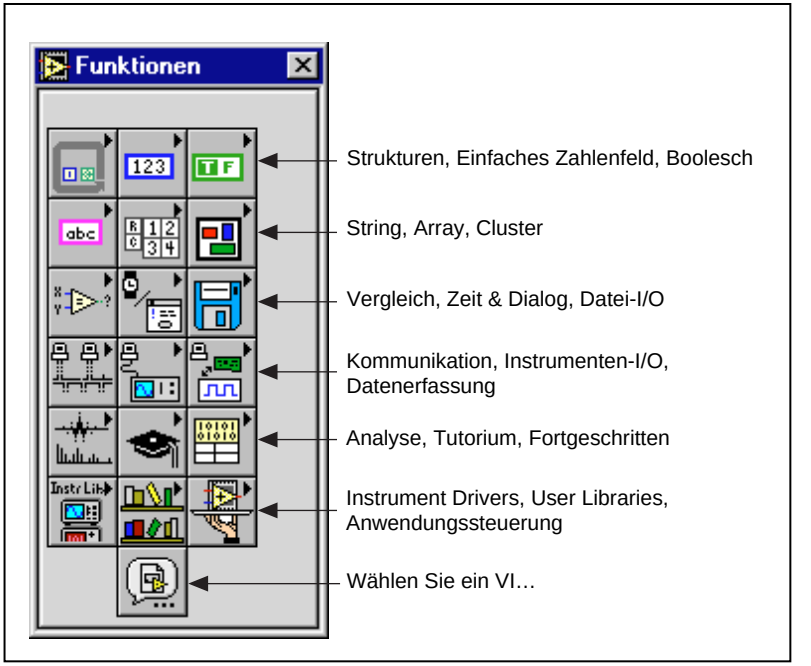
Das Development System beinhaltet eine Sammlung von VIs, die mit der Programmiersprache G, mit Hardware-Geräten zur Datenerfassung (DAQ), Instrumenten und anderen Kommunikationsschnittstellen zusammenarbeiten.

G-Funktionen und VIs auffinden

Funktionen sind elementare Knoten in der Programmiersprache G. Sie entsprechen Operatoren oder Bibliotheksfunktionen in herkömmlichen Sprachen. Funktionen sind keine VIs und besitzen daher kein Frontpanel oder Blockdiagramm. Beim Kompilieren erzeugen die Funktionen Maschinencode.

Funktionen können im Blockdiagramm von der **Funktionenpalette** ausgewählt werden. Wenn das Blockdiagrammfenster aktiv ist, wählen Sie hierzu **Fenster**»**Funktionenpalette**. Daneben können Sie aber auch auf die **Funktionen**palette zugreifen, indem Sie an der Stelle im Blockdiagrammfenster, an der Sie die Funktion einsetzen möchten, das Popup-Menü aufrufen.

Die folgende Abbildung zeigt die auf der Funktionenpalette verfügbaren Funktionen und VIs.



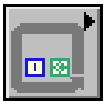
Viele Kapitel über die **Funktionenpalette** enthalten Informationen zu Funktionsbeispielen. Der Pfad zu diesen Beispielen für LabVIEW beginnt mit `examples\`.

Überblick über die Funktionen und VIs

Die folgenden Funktionen und VIs stehen in der **Funktionenpalette** zur Verfügung:

Strukturen

G-Strukturen umfassen While-Schleifen-, FOR-Schleifen-, Case- und Sequenzstrukturen. Diese Palette enthält außerdem die globalen und lokalen Variablen-Knoten und Formel-Elemente.



Numerische Funktionen

Numerische Funktionen führen arithmetische Operationen, Umwandlungen, trigonometrische, logarithmische und komplexe mathematische Operationen aus. Diese Palette enthält zusätzlich numerische Konstanten wie π.

Boolesche Funktionen

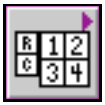
Boolesche Funktionen können Boolesche und logische Operationen ausführen.

String-Funktionen

String-Funktionen manipulieren Strings und wandeln Zahlen in Strings und umgekehrt um. Diese Palette beinhaltet außerdem die Zusätzliche Funktionen Stringumwandlung und die Funktionen Stringumwandlung.

Array-Funktionen

Array-Funktionen stellen Arrays zusammen, nehmen diese auseinander und verarbeiten sie.

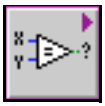


Cluster-Funktionen

Cluster-Funktionen stellen Elemente eines Clusters zusammen, greifen auf diese zu und verändern sie.

Vergleichsfunktionen

Vergleichsfunktionen vergleichen Daten (größer als, kleiner als usw.) und Operationen, die auf Vergleichen gründen (z.B. Auffinden der Minimal- und Maximalbereiche für eine Gruppe oder ein Array von Werten).



Zeit- und Dialogfunktionen

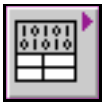
Zeit- und Dialogfunktionen manipulieren Zeitfunktionen und zeigen Dialogkästchen an. Diese Palette beinhaltet außerdem VIs zur Fehlerbehandlung.

Datei-I/O-Funktionen

Datei-I/O-Funktionen manipulieren Dateien und Verzeichnisse. Diese Palette beinhaltet außerdem die Unterpaletten Fortgeschrittene Dateifunktionen, Binärdatei-VIs und Dateikonstanten.

Fortgeschrittene Funktionen

Fortgeschrittene Funktionen sind hochspezialisierte Funktionen. Der Code Interface Node ist ein Beispiel einer fortgeschrittenen Funktion. Die Palette **Fortgeschrittene** beinhaltet auch Datenmanipulationsfunktionen und Occurrence-Funktionen.



DAQ

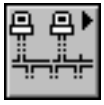
DAQ-VIs erfassen und erzeugen analoge und digitale Daten in Echtzeit und führen Zähloperationen aus. Für weiterführende Informationen lesen Sie bitte Kapitel 14, *Einführung zu den LabVIEW Datenerfassungs-VIs*.

Geräte I/O

Geräte-I/O-VIs kommunizieren über GPIB, VISA oder serielle Kommunikation mit Instrumenten. Für weiterführende Informationen lesen Sie bitte Kapitel 31, *Einführung in LabVIEW Geräte-I/O-VIs*.

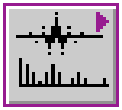
Kommunikation

Kommunikations-VIs tauschen mit anderen Anwendungen im Netzwerk Daten mit Hilfe der Protokolle TCP/IP, DDE, ActiveX, Apple Events, PPC oder UDP aus. Lesen Sie hierzu Kapitel 48, *TCP-VIs*, bis Kapitel 53, *Programm-zu-Programm-Kommunikations-VIs*, für weitere Informationen.



Analyse VIs

Analyse-VIs dienen folgenden Funktionen: Messung, Signalerzeugung, digitale Signalverarbeitung, Filterung, Fensterung, Wahrscheinlichkeit und Statistik, Kurvenanpassung, lineare Algebra, Arrayoperationen und VIs für zusätzliche numerische Methoden. Für weiterführende Informationen lesen Sie bitte Kapitel 37, *Einführung in die Analyse unter LabVIEW*.



VI auswählen

VI auswählen ermöglicht Ihnen, über ein Datei-Dialogfeld ein VI auszuwählen und anschließend auf einem Diagramm anzuordnen.

Tutorial

Tutorial-VIs bieten Ihnen Beispiele zur Verwendung, während Sie das *LabVIEW Benutzerhandbuch* durcharbeiten.

Gerätetreiberbibliothek

Gerätetreiber bilden einen Satz von VIs für GPIB, VISA, serielle und CAMAC-Geräte. National Instruments sowie andere Hersteller vertreiben diese Gerätetreiber. Jeder in `instr.lib` eingefügte Treiber erscheint in der Palette.

Benutzerbibliothek

Die **Benutzerbibliothek**-Palette beinhaltet automatisch alle VIs in Ihrem `user.lib` Verzeichnis, wodurch verbesserter Zugriff auf die von Ihnen geschriebenen und am häufigsten verwendeten SubVIs besteht.

Anwendungssteuerung

Die **Anwendungssteuerung**-Palette enthält Hilfefunktionen, Menüfunktionen, Druck-VIs und Server-VIs.

Teil I

G-Funktionen und VIs

Teil I, *G-Funktionen und VIs*, bietet eine Einführung zu den Beschreibungen der G-Funktionen und VIs. In diesem Teil sind folgende Kapitel enthalten:

- Kapitel 2, *Nachschlage-Übersicht über Funktionen von G und VIs*, stellt die G-Funktionen und VIs vor. In diesem Kapitel werden auch die Unterschiede zwischen Funktionen und VIs beschrieben.
- Kapitel 3, *Strukturen*, beschreibt die in G verfügbaren Strukturen.
- Kapitel 4, *Numerische Funktionen*, beschreibt die Funktionen, die arithmetische Operationen sowie komplexe, logarithmische, trigonometrische und Umwandlungsoperationen durchführen. Des weiteren werden hier häufig benutzte Konstanten beschrieben, wie die numerische Konstante, die Konstante des Aufzähltyps Enum und die Ringkonstante sowie zusätzliche numerische Konstanten.
- Kapitel 5, *Boolesche Funktionen*, beschreibt die Funktionen, die logische Operationen ausführen.
- Kapitel 6, *String-Funktionen*, beschreibt die Stringfunktionen, einschließlich derer, die Strings in Zahlen und umgekehrt umwandeln.
- Kapitel 7, *Array-Funktionen*, beschreibt die Funktionen für Array-Operationen.
- Kapitel 8, *Cluster-Funktionen*, beschreibt die Funktionen für Cluster-Operationen.
- Kapitel 9, *Vergleichsfunktionen*, beschreibt die Funktionen, die Vergleiche oder bedingte Tests ausführen.
- Kapitel 10, *Zeit-, Dialog- und Fehlerfunktionen*, beschreibt die Timing-Funktionen, die Sie benutzen können, um die aktuelle Zeit zu erlangen, verstrichene Zeit zu messen oder eine Operation für eine bestimmte Zeitspanne zu unterdrücken. In diesem Kapitel wird auch die Fehlerbehandlung besprochen.

- Kapitel 11, *Dateifunktionen*, beschreibt die Low-Level-VIs und Funktionen, die Dateien, Verzeichnisse und Pfade verarbeiten. In diesem Kapitel werden auch Dateikonstanten und die High-Level-Datei-VIs beschrieben.
- Kapitel 12, *Funktionen zur Anwendungssteuerung*, beschreibt die Funktionen der Anwendungssteuerung.
- Kapitel 13, *Fortgeschrittene Funktionen*, beschreibt die Funktionen, die fortgeschrittene Operationen ausführen. In diesem Kapitel werden auch die Hilfe,- Datenverarbeitungs- und Synchronisationsfunktionen sowie das VI-Element und der VISA-Speicher beschrieben.

2

Nachschlage-Übersicht über Funktionen von G und VIs

Dieses Kapitel stellt die Funktionen von G und VIs vor, von denen die Kapitel 3–13 Beschreibungen enthalten.

Funktionen sind grundlegende Knoten in der Programmiersprache G. Sie arbeiten wie Operatoren oder Bibliotheksfunktionen in herkömmlichen Sprachen. Funktionen sind keine VIs und besitzen daher kein Frontpanel oder Blockdiagramm. Bei ihrer Kompilierung erzeugen sie Maschinencode.

VIs sind “virtuelle Instrumente”; Sie tragen diese Bezeichnung, da sie mit ihrem Aussehen die Funktionen physischer Instrumente nachahmen.

Funktionen von G werden in der **Funktionenpalette** innerhalb des Blockdiagramms ausgewählt. Sie können die **Funktionenpalette** über **Fenster**»**Funktionenpalette** anzeigen, wenn das Blockdiagrammfenster aktiv ist. Sie können außerdem auf die **Funktionenpalette** zugreifen, indem Sie an der Stelle im Blockdiagrammfenster, an der Sie die Funktion einfügen möchten, das Popup-Menü aufrufen.

Die folgende Abbildung zeigt die auf der **Funktionenpalette** verfügbaren G-Funktionen und -VIs. Hinter den abgedunkelten Feldern verbergen sich weitere Funktionen, die aber in anderen Kapiteln dieses Handbuchs behandelt werden.

Viele Kapitel über die **Funktionenpalette** enthalten Informationen über Beispiele von Funktionen.

Überblick über Funktionen von G

Eine kurze Beschreibung von jeder der sieben verfügbaren G-Funktionen- und VI-Paletten finden Sie im Kapitel 1, *Einführung in die G-Funktionen und in die VIs*.

Einführung in Polymorphismus

Die folgenden Abschnitte enthalten einige allgemeine Informationen über den Polypmorphismus von G-Funktionen.

Polymorphismus

Polymorphismus ist die Fähigkeit einer Funktion, sich an verschiedene Typen oder Repräsentierungen von Eingabedaten anzupassen. Die Mehrzahl der Funktionen ist polymorph. VIs sind nicht polymorph. Alle Funktionen, die mit numerischen Eingabedaten arbeiten, können numerische Repräsentierungen jeglicher Art akzeptieren (außer einigen Funktionen, die keine komplexen Zahlen zulassen).

Funktionen sind zu verschiedenen Graden polymorph; keine, einige oder alle Eingaben können polymorph sein. Einige Funktioneneingaben akzeptieren Zahlen oder Boolesche Werte. Einige akzeptieren Zahlen oder Strings. Einige lassen nicht nur skalare Zahlen, sondern auch Zahlenarrays, Zahlencluster, Arrays von Zahlenclustern usw. zu. Einige akzeptieren nur eindimensionale Arrays, wobei die Arrayelemente von jedem Datentyp sein können. Einige Funktionen lassen alle Datentypen zu, komplexe Zahlen eingeschlossen.

Einheiten-Polymorphismus

Um ein VI zu erstellen, das die Wurzel, den Effektivwert einer Kurvenform, berechnet, muß eine mit der Kurvenform verbundene Einheit festgelegt werden. Es wird ein separates VI für Spannungskurven, Stromkurven, Temperaturkurven usw. benötigt. LabVIEW besitzt eine polymorphe Einheitenfähigkeit, wodurch ein VI in der Lage ist, dieselbe Berechnung unabhängig von den an den Eingängen erhaltenen Einheiten auszuführen.

Sie können eine polymorphe Einheit erstellen, indem Sie $*n* eingeben, wobei *n* eine Zahl ist (z.B. $1). Sie können sich diesen Eintrag als Platzhalter für die tatsächliche Einheit vorstellen. Wenn LabVIEW das VI aufruft, setzt das Programm in diesem VI die Einheiten, die Sie eingeben, für jedes Auftreten von $*n* ein.

LabVIEW behandelt eine polymorphe Einheit als eine einzigartige Einheit. Eine polypmorphische Einheit kann nicht in eine andere Einheit umgewandelt werden und durchläuft das gesamte Diagramm genau wie jede andere Einheit. Wenn die Einheit mit einer Anzeige, die ebenfalls über die Abkürzung $1 verfügt, verbunden ist, stimmen die Einheiten überein, und das VI kann dann kompilieren.

Sie können die $1-Kombination wie jede andere Einheit verwenden. Wenn z.B. die Eingabe mit 3 Sekunden multipliziert und dann zu einer Anzeige weitergegeben wird, dann muß die Anzeige mit $1 s Einheiten arbeiten. Wenn die Anzeige mit anderen Einheiten arbeitet, zeigt das Blockdiagramm eine ungültige Verbindung an. Wenn Sie mehr als eine polymorphe Einheit verwenden müssen, können Sie die Abkürzungen $2, $3 usw. einsetzen.

Ein Aufruf an ein SubVI mit polymorphen Einheiten berechnet die Ausgabeeinheiten auf der Grundlage der an seinen Eingängen erhaltenen Einheiten. Stellen Sie sich z.B. vor, daß Sie ein VI erstellen, das an seinen zwei Eingängen mit polymorphen Einheiten $1 und $2 arbeitet und Daten in der Form $1 $2 / s ausgibt. Wenn ein Aufruf an das VI am $1-Eingang Eingaben in der Einheit m/s erhält und an dem $2-Eingang in der Einheit kg, dann berechnet LabVIEW die Ausgabeeinheit als kg m / s^2.

Stellen Sie sich vor, ein anderes VI verfügt über zwei Eingänge in der Form $1 und $1/s und berechnet eine Ausgabe in der Form 1^2. Wenn ein Aufruf an dieses VI an dem $1-Eingang Eingaben in der Einheit m/s erhält und an dem $1/s-Eingang in der Einheit m/s^2, dann berechnet LabVIEW die Ausgabeeinheit als m^2 / s^2. Wenn ein Aufruf an dieses VI jedoch an dem $1-Eingang Eingaben in der Einheit m erhält und an dem $1/s-Eingang in der Einheit kg, dann erklärt LabVIEW eine der Eingaben zum Einheitenkonflikt und berechnet (falls möglich) die Ausgabe von dem anderen Eingang.

Ein polymorphes VI kann ein polymorphes SubVI besitzen, da LabVIEW die verschiedenen Einheiten auseinanderhält.

Numerische Umwandlung

Jede numerische Repräsentierung kann in jede andere numerische Repräsentierung umgewandelt werden. Wenn zwei oder mehrere numerische Eingänge mit verschiedenen Repräsentierungen zu einer Funktion verbunden werden, gibt die Funktion normalerweise eine Ausgabe in der größeren oder breiteren Einteilung zurück. Die Funktionen wandeln die kleineren Repräsentierungen vor der Ausführung in die größte um.

Einige Funktionen, wie z.B. Dividieren, Sinus und Kosinus, erzeugen immer Ausgaben mit einem Fließkomma. Wenn Sie ganze Zahlen an deren Eingängen eingeben, wandeln diese Funktionen die ganzen Zahlen vor der Ausführung der Berechnung in Zahlen mit zwei Dezimalstellen hinter einem Fließkomma um (Fließkommazahlen [2stellig]).

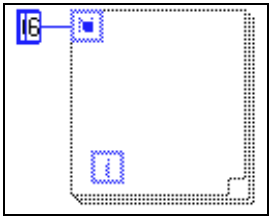
Für Skalarmengen mit einem Fließkomma werden gewöhnlicherweise am besten Fließkommazahlen mit doppelter Stellengenauigkeit verwendet. Zahlen mit einer Dezimalstelle hinter dem Komma sparen wenig oder keine Zeit in der Ausführung und laufen sehr viel schneller über. Die Analyse-Bibliotheken z.B. arbeiten mit Fließkommazahlen (2stellig). Extended Fließkommazahlen sollten nur verwendet werden, wenn es notwendig ist. Die Leistung und Präzision von arithmetischen Operationen mit extended Fließkommazahlen hängt von der jeweiligen Plattform ab.

Für ganze Zahlen wird gewöhnlichweise am besten ein längerer Integer-Wert verwendet.

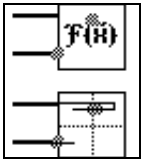
Wenn ein Ausgang mit einem Ziel verbunden wird, das mit einer anderen numerischen Repräsentierung als der von der Quelle arbeitet, wandelt G die Daten entsprechend folgender Regeln um:

- Ganze Zahlen mit oder ohne Vorzeichen in Fließkommazahlen - Die Umwandlung ist genau, außer bei der Umwandlung von langen ganzen Zahlen in Fließkommazahlen mit einfacher Genauigkeit. In diesem Fall reduziert G die Präzision von 32 Bit zu 24 Bit.
- Fließkommazahlen in ganze Zahlen mit oder ohne Vorzeichen - G verwandelt Werte außerhalb des Bereichs in den Minimal- oder Maximalwert der ganzen Zahl. Die Mehrzahl der Integer-Objekte, wie z.B. das Iterationsterminal einer FOR-Schleife, rundet die Fließkommazahlen auf bzw. ab. G rundet Bruchzahlen mit 0,5 zur nächsten geraden ganzen Zahl, z.B. 6,5 eher zu 6 anstatt auf 7.
- Ganze Zahl in ganze Zahl - G verwandelt Werte außerhalb des Bereichs nicht in den Maximal- oder Minimalwert der ganzen Zahl. Wenn die Quelle kleiner als das Ziel ist, erweitert G das Vorzeichen einer vorzeichenbehafteten Quelle und plaziert Nullen in die überzähligen Bits einer vorzeichenlosen Quelle. Wenn die Quelle größer als das Ziel ist, kopiert G von dem Wert nur die niedrigstwertigen Bits.

In einem Blockdiagramm wird wie in dem folgenden Beispiel ein *Formatumwandlungspunkt* an den Rand des Terminals, an dem die Umwandlung stattfand, gesetzt, um darauf hinzuweisen, daß eine automatische numerische Umwandlung stattgefunden hat.



Da VIs und Funktionen viele Terminals besitzen können, kann ein Formatumwandlungspunkt auf der Innenseite eines Icons erscheinen, wenn eine Verbindung vor dem Verlassen des Icons/Anschlusses einen internen Terminalrand kreuzt, wie in der folgenden Abbildung gezeigt.



Verschieben eines verbundenen Icons dehnt die Verbindung. Ein Formatumwandlungspunkt kann dazu führen, daß ein VI mehr Speicherplatz beansprucht und seine Ausführungszeit verlängert wird. Datentypen in VIs sollten möglichst gleich gehalten werden.

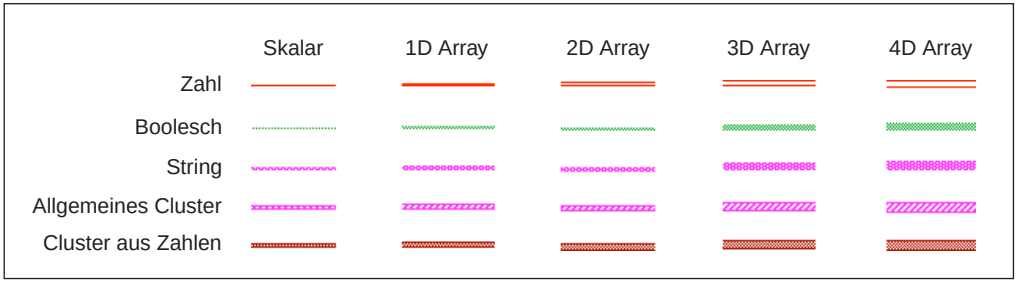
Überlauf und Unterlauf

G kontrolliert nicht auf Überlauf- und Unterlaufzustände von Integer-Werten. Überlauf und Unterlauf für Fließkommazahlen entspricht dem IEEE 754 Standard für binäre Fließkommaarithmetik.

Fließkommaoperationen geben Keine Zahl (NaN) und ± Unendlich akkurat weiter. Wenn ein NaN oder ± Unendlich explizit oder implizit in einen Integer- oder Booleschen Wert umgewandelt wird, wird ein Wert ausgegeben, der zwar reell erscheint, doch bedeutungslos ist. Division durch Null, z.B., ergibt ±Unendlich; die Umwandlung dieses Werts in einen Word-Integer ergibt den Wert 32768, welcher der größte in diesem Format darstellbare Wert ist.

Verbindungsstile

Der Verbindungsstil zeigt den Datentyp für jedes Terminal, wie in der folgenden Tabelle aufgelistet, an. Polymorphe Funktionen tragen den Verbindungsstil für den am häufigsten verwendeten Datentyp.

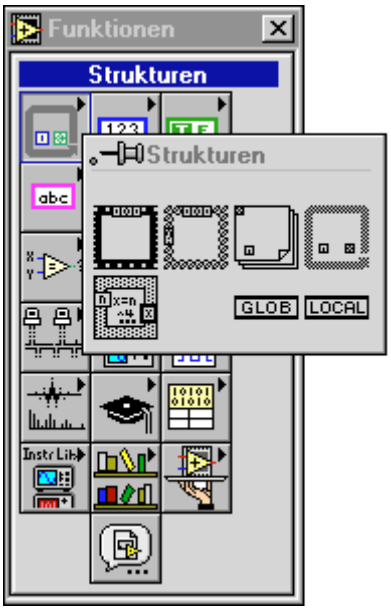


3

Strukturen

Dieses Kapitel beschreibt die in G verfügbaren Strukturen.

Die **Strukturenpalette** wird über das Menü **Funktionen»Strukturen** aufgerufen. Die folgende Abbildung zeigt die in der **Strukturenpalette** verfügbaren Optionen.



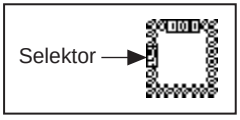
Beispiele für den Einsatz dieser Strukturen in LabVIEW finden Sie in `examples\general\structs.llb`.

Strukturen-Überblick

Die nachfolgenden Strukturen sind in G verfügbar.

Case-Struktur

Sie besitzt ein oder mehrere Unterdiagramme, oder *Cases*, von denen genau eins abläuft, wenn die Struktur abläuft. Ob es abläuft, hängt von dem Booleschen, String- oder numerischen Skalarwert ab, der mit der Außenseite des Terminals, dem *Selektor,* verbunden ist.



Für weitere Informationen über den Einsatz von Case-Strukturen in LabVIEW lesen Sie bitte im *LabVIEW Benutzerhandbuch* das Kapitel 4, *Case- und Sequenzstrukturen und das Formel-Element.*

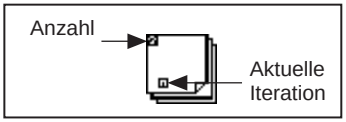
Sequenzstruktur

Sie setzt sich aus einem oder mehreren Unterdiagrammen, oder *Rahmen*, zusammen, die nacheinander ablaufen. Als Option können lokale Sequenzvariablen hinzugefügt werden, mit denen Informationen von einem Rahmen an nachstehende Rahmen weitergegeben werden können. Diese Option ist über das Popup-Menü der Struktur zugänglich.

Für weitere Informationen über den Einsatz von Sequenzstrukturen in LabVIEW lesen Sie bitte im *LabVIEW Benutzerhandbuch* das Kapitel 4, *Case- und Sequenzstrukturen und das Formel-Element.*

FOR-Schleife

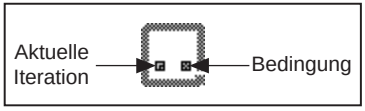
Sie führt ihre Unterdiagramme *n*-Mal aus, wobei *n* dem Wert in dem Zählerterminal entspricht. Als Option können Shift-Register hinzugefügt werden, durch die Informationen von einer Iteration an die nächste weitergegeben werden können. Diese Option ist über das Popup-Menü der Struktur zugänglich.



Für weitere Informationen über den Einsatz von FOR-Schleifen in LabVIEW lesen Sie bitte im *LabVIEW Benutzerhandbuch* das Kapitel 3, *Schleifen und Diagramme*.

While-Schleife

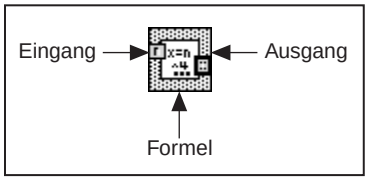
Sie führt ihre Unterdiagramme solange aus, bis ein mit dem *Bedingungsanschluß* verbundener Boolescher Wert FALSE ist. Als Option können Shift-Register hinzugefügt werden, durch die Informationen von einer Iteration an die nächste weitergegeben werden können. Diese Option ist über das Popup-Menü der Struktur zugänglich.



Für weitere Informationen über den Einsatz von While-Schleifen in LabVIEW lesen Sie bitte im *LabVIEW Benutzerhandbuch* das Kapitel 3, *Schleifen und Diagramme.*

Formel-Element

Es führt mathematische Formeln auf dem Blockdiagramm aus.



Für weitere Informationen über das Formel-Element lesen Sie bitte im *LabVIEW Benutzerhandbuch* das Kapitel 4, *Case- und Sequenzstrukturen und das Formel-Element.*

Globale Variable

Sie ist ein eingebautes LabVIEW-Objekt, das Sie definieren, indem Sie ein VI von spezieller Art erstellen. Dieses Objekt besitzt auf dem Frontpanel Bedienelemente, die den Datentyp der globalen Variable festlegen. Es können von dieser globalen Variablen sowohl Werte gelesen als auch zu ihr geschrieben werden.

Für weitere Informationen über lokale Variablen lesen Sie bitte im *LabVIEW Referenzhandbuch zur Programmierung in G* das Kapitel 22, *Globale und lokale Variablen.*

Lokale Variable

Durch sie kann zu einem der Bedienelemente oder zu einer der Anzeigen auf dem Frontpanel eines VIs geschrieben oder von einem gelesen werden. Das Schreiben zu einer lokalen Variable führt zu demselben Ergebnis wie das Weitergeben von Daten an ein Terminal, mit dem Unterschied, daß zu ihr geschrieben werden kann, obwohl es sich bei ihr um ein Bedienelement handelt, bzw. von ihr gelesen werden kann, obwohl es sich um eine Anzeige handelt.

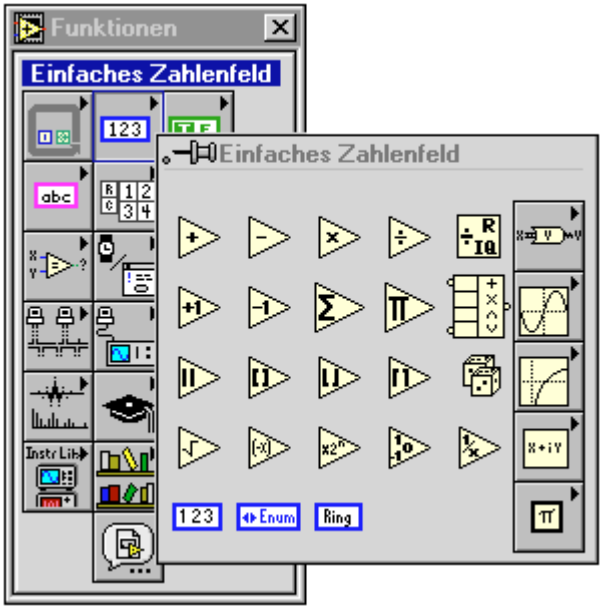
Für weitere Informationen über lokale Variablen lesen Sie bitte im *LabVIEW Referenzhandbuch zur Programmierung in G* das Kapitel 23, *Globale und lokale Variablen*.

4

Numerische Funktionen

Dieses Kapitel beschreibt die Funktionen, die arithmetische, komplexe, Umwandlungs-, logarithmische und trigonometrische Operationen ausführen. Daneben beschreibt es häufig verwendete Konstanten wie die Numerische Konstante, die Enumerated Constant und die Ring-Konstante sowie zusätzliche numerische Konstanten.

Auf die **Numerische** Palette wird über das Menü **Funktionen» Numerische** zugegriffen. Die folgende Abbildung zeigt die auf der **Numerischen** Palette verfügbaren Optionen.



Die **Numerische** Palette beinhaltet folgende Unterpaletten:

- Zusätzliche numerische Konstanten
- Komplex
- Umwandlung

- Logarithmisch
- Trigonometrisch

Beispiele für einige der arithmetischen Funktionen finden Sie in `examples\general\structs.llb`.

Polymorphismus für numerische Funktionen

Die arithmetischen Funktionen akzeptieren numerische Eingabedaten. Mit Ausnahme der in den Funktionsbeschreibungen erwähnten Einschränkungen gleicht die numerische Repräsentierung der Ausgabe der des Eingangs; oder wenn die Eingaben unterschiedliche Repräsentierungen haben, ist die Ausgabe die längste von den Eingaben.

Die arithmetischen Funktionen arbeiten mit Zahlen, Zahlenarrays, Zahlenclustern, Arrays aus Zahlenclustern, komplexen Zahlen usw. Eine formale und rekursive Definition des zulässigen Eingabetyps besteht, wie folgt:

> *Numerischer Typ* = numerischer Skalar || Array [*numerischer Typ*] || Cluster [*numerische Typen*]
>
> Die numerischen Skalare können Fließkommazahlen, ganze Zahlen oder komplexe Zahlen sein. G läßt keine Arrays von Arrays zu.

Arrays können beliebig viele Dimensionen jeglicher Größe aufnehmen. Cluster können aus beliebig vielen Elementen bestehen. Funktionen mit einer Eingabe arbeiten mit jedem Elemente der Struktur.

Für Funktionen mit zwei Eingaben können folgende Eingabekombinationen verwendet werden:

- *Ähnlich* - beide Eingaben weisen dieselbe Struktur auf, und die Ausgabe hat dieselbe Struktur wie die Eingaben
- *Ein Skalar* - eine Eingabe ist ein numerischer Skalar, die andere ist ein Array oder Cluster, und die Ausgabe ist ein Array oder Cluster
- *Array aus* - eine Eingabe ist ein numerisches Array, die andere ist der numerische Typ selbst, und die Ausgabe ist ein Array

Bei ähnlichen Eingaben führt G die Funktion mit den jeweiligen Elementen der Strukturen aus. G kann z.B. zwei Arrays von Element zu Element addieren. Beide Arrays müssen dieselben Dimensionen aufweisen. Arrays mit unterschiedlicher Anzahl von Elementen können addiert werden; die Ausgabe einer solchen Addition hat dieselbe Anzahl von Elementen wie die kleinste Eingabe. Cluster müssen gleichfalls in der Anzahl von

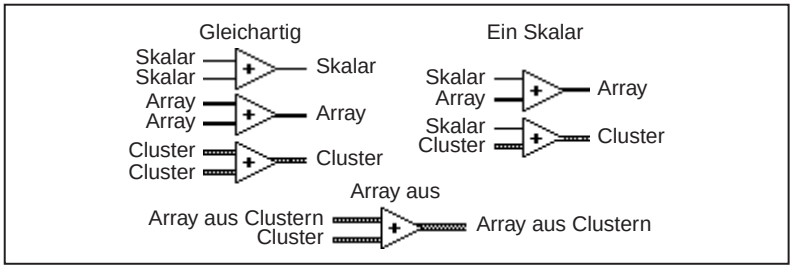
Elementen übereinstimmen, und die jeweiligen Elemente müssen dieselbe Struktur besitzen.

☞ **Hinweis** ***Die Multiplikationsfunktion kann nicht für die Multiplikation von Matrizen eingesetzt werden. Wenn die Multiplikationsfunktion für zwei Matrizen verwendet wird, multipliziert G die erste Zahl in der ersten Reihe der ersten Matrix mit der ersten Zahl in der ersten Reihe der zweiten Matrix usw.***

Bei Operationen mit einem Skalar und einem Array oder Cluster führt G die Funktion mit dem Skalar und den jeweiligen Elementen der Struktur aus. G kann z.B. eine Zahl von allen Elementen eines Arrays subtrahieren, unabhängig von den Dimensionen des Arrays.

Bei Operationen mit einem numerischen Typ und einem Array dieses Typs führt G die Funktion mit jedem Arrayelement aus. Ein Graph z.B. ist ein Array von Punkten, und ein Punkt ist ein Cluster aus zwei numerischen Typen, *x* und *y*. Um einen Graph um 5 Einheiten in der *x*-Richtung und 8 Einheiten in der *y*-Richtung zu versetzen, kann ein Punkt (5, 8) zu dem Graphen addiert werden.

Das Beispiel polymorpher Kombinationen weiter unten veranschaulicht einige der polymorphen Kombinationen der Addierfunktion.



Polymorphismus für transzendente Funktionen

Transzendente Funktionen akzeptieren numerische Eingabedaten. Wenn die Eingabe eine ganze Zahl ist, ist die Ausgabe eine Fließkommazahl (2stellig). Ansonsten hat die Ausgabe dieselbe numerische Repräsentierung wie die Eingabe.

Diese Funktionen arbeiten mit Zahlen, Zahlenarrays, Zahlenclustern, Arrays aus Zahlenclustern, komplexen Zahlen usw.

Polymorphismus für Umwandlungsfunktionen

Alle Umwandlungsfunktionen außer Byte-Array in String, String in Byte-Array, Einheitenkonvertierung und Wandelt Einheitenbasis sind polymorph. Deshalb arbeiten die polymorphen Funktionen mit Skalarwerten, Skalararrays, Skalarclustern, Arrays aus Skalarclustern usw. Die Ausgabe hat dieselbe numerische Darstellung wie die Eingabe, nur mit dem neuen Typ.

Wenn ganze Zahlen mit und ohne Vorzeichen verglichen werden und der Integer negativ ist, dann wird die ganze Zahl vor dem Vergleich in eine positive umgewandelt. Dadurch erhalten Sie also nicht das erwartete Ergebnis. Wenn Sie z.B. als eine Eingabe –1 mit der Repräsentierung I32 eingeben und 5 mit der Repräsentierung U32 als die andere Eingabe, wird im Ergebnis ausgegeben, daß 5 der kleinere Wert ist, da 5 kleiner als 4,294,967,295 ist.

Polymorphismus für komplexe Funktionen

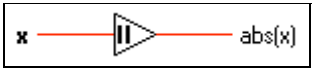
Komplexe Funktionen arbeiten mit Skalarwerten, Skalararrays, Skalarclustern, Arrays aus Skalarclustern usw. Die Ausgabe hat dieselbe Zusammensetzung wie die Eingabe, nur mit einem neuen Typ.

Beschreibungen der arithmetischen Funktionen

Die folgenden Funktionen stehen zur Verfügung.

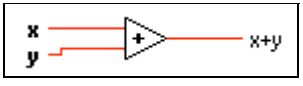
Absoluter Wert

Gibt die absoluten Werte einer Eingabe aus.



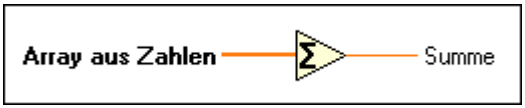
Addieren

Berechnet die Summe der Eingaben.



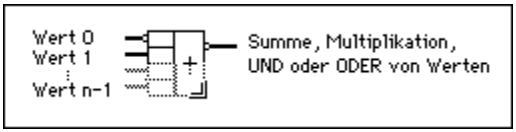
Array-Elemente addieren

Gibt die Summe aller Elemente in einem **Array aus Zahlen** aus.



Mehrfacharithmetik

Führt arithmetische Operationen mit zwei oder mehreren numerischen, Cluster- oder Booleschen Eingaben aus.



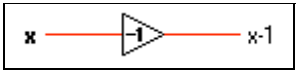
Die Operation (Multiplizieren, UND oder ODER) wird ausgewählt, indem Sie mit der rechten Maustaste auf die Funktion klicken und **Ändert Modus** auswählen.

Die Eingaben und Ausgaben dieser Funktion können umgekehrt werden, indem Sie das Popup-Menü des jeweiliges Terminals aufrufen und **Invertieren** auswählen. Beim Addieren kann durch **Invertieren** eine Eingabe oder die Ausgabe negiert werden. Beim Multiplizieren wird durch **Invertieren** der reziproke Wert einer Eingabe verwendet oder der reziproke Wert der Ausgabe erzeugt. Bei Funktionen UND oder ODER kann mit **Invertieren** eine Eingabe oder Ausgabe logisch negiert werden.

☞ **Hinweis** ***Eingänge können zu diesem Knoten hinzugefügt werden, indem Sie das Popup-Menü eines Eingangs aufrufen und Eingang hinzufügen auswählen oder indem Sie das Positionierwerkzeug in der unteren linken oder rechten Ecke des Knotens plazieren und es ziehen.***

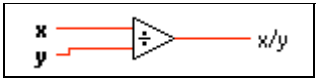
Dekrementieren

Subtrahiert von dem Eingabewert eine 1.



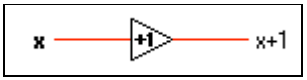
Dividieren

Berechnet den Quotienten der Eingaben.



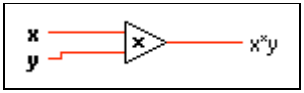
Inkrementieren

Addiert zum Eingabewert eine 1.



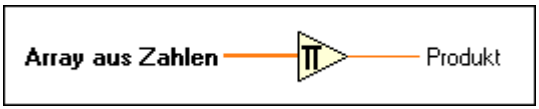
Multiplizieren

Gibt das Produkt der Eingaben aus.



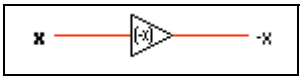
Array-Elemente multiplizieren

Gibt das Produkt aller Elemente in einem **Array aus Zahlen** aus.



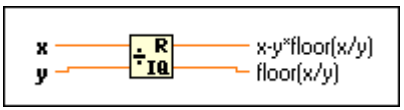
Negieren

Negiert die Eingabewerte.



Quotient & Rest

Berechnet den ganzzahligen Quotienten und den Rest der Eingaben.



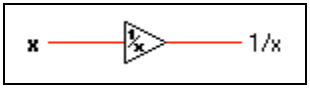
Wenn der ganzzahlige Eingabewert von **y** Null beträgt, beträgt der Quotient Null und der Dividend **x** bildet den Rest. Wenn bei Eingaben mit Fließkomma **y** Null beträgt, ist der Quotient unendlich und der Rest wird automatisch auf Keine Zahl gesetzt.

Zufallszahl (0–1)

Erzeugt eine Fließkommazahl (2stellig) zwischen 0 und 1, wobei die Werte 0 und 1 ausgeschlossen bzw. nicht eingeschlossen sind. Es handelt sich hierbei um eine Rechteckverteilung.

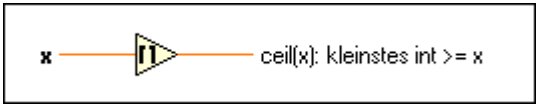
Kehrwert

Dividiert 1 durch den Eingabewert.



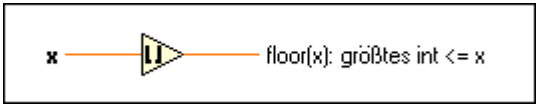
Auf nächst größere Zahl runden

Rundet die Eingabe auf die nächstgrößere ganze Zahl auf. Wenn z.B. die Eingabe 3,1 beträgt, beträgt das Ergebnis 4. Wenn die Eingabe -3,1 beträgt, ist das Ergebnis -3.



Auf nächst kleinere Zahl runden

Rundet die Eingabe auf die nächstniedrigere ganze Zahl ab. Wenn z.B. die Eingabe 3,8 beträgt, beträgt das Ergebnis 3. Wenn die Eingabe -3,8 beträgt, ist das Ergebnis -4.

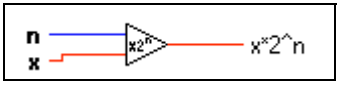


Auf nächste ganze Zahl runden

Rundet die Eingabe auf die nächste ganze Zahl. Wenn der Eingabewert in der Mitte zwischen zwei ganzen Zahlen steht, (z.B. 1,5 oder 2,5), gibt die Funktion die nächste gerade ganze Zahl aus.

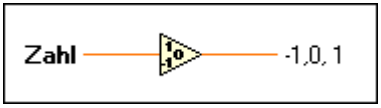
Multipliziere mit Potenz von 2

Multipliziert eine Eingabe (**x**) mit 2 hoch der anderen Eingabe (**n**). Wenn **n** eine Fließkommazahl ist, rundet diese Funktion den Wert **n** vor dem Skalieren von **x** entweder ab oder auf (0,5 wird auf 0 abgerundet; 0,51 wird auf 1 aufgerundet). Wenn **x** eine ganze Zahl ist, bildet diese Funktion das Äquivalent zu einer arithmetischen Stellenverschiebung.



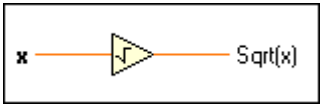
Vorzeichen

Gibt eine 1 aus, wenn der Eingabewert größer als 0 ist; gibt eine 0 aus, wenn der Eingabewert 0 beträgt, und gibt eine -1 aus, wenn der Eingabewert kleiner als 0 ist. In anderen Programmiersprachen wird diese Funktion normalerweise die `signum-` oder `sgn`-Funktion genannt.



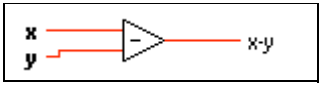
Quadratwurzel

Berechnet die Quadratwurzel des Eingabewerts. Wenn **x** negativ ist, lautet die Quadratwurzel Keine Zahl, es sei denn, es handelt sich bei **x** um eine komplexe Zahl.



Subtrahieren

Berechnet die Differenz zwischen den Eingaben.



Benutzerspezifische arithmetische Konstanten

Sie können die folgenden Konstanten festlegen:

Numerische Konstanten

Verwenden Sie diese Konstante, wenn ein konstanter numerischer Wert in das Blockdiagramm eingespeist werden soll. Dieser Wert wird gesetzt, indem Sie mit dem Bedienwerkzeug innerhalb der Konstante klicken und einen Wert eingeben. Das Datenformat und die Datenrepräsentierung können verändert werden.

Wenn das VI gerade abläuft, kann der Wert der numerischen Konstante nicht verändert werden. Die Konstante kann mit einem Label versehen werden.

Enumerated Constant

Der Aufzählungstyp Enum stellt Assoziationen zwischen vorzeichenlosen ganzen Zahlen und Strings her. Wenn von einer Enumerated Constant ein Wert angezeigt wird, ist der String anstelle der mit ihm assoziierten Zahl zu sehen. Wenn Sie einen unveränderlichen Satz von Strings benötigen, sollten Sie diese Konstante verwenden. Sie setzen den Wert, indem Sie mit dem Bedienwerkzeug in der Konstante klicken. Den String setzen Sie mit dem Beschriftungswerkzeug; geben Sie den String ein. Ein weiteres Objekt wird hinzugefügt, indem Sie auf die Konstante klicken und **Objekt davor einfügen** oder **Objekt danach einfügen** wählen.

Wenn das VI gerade abläuft, kann der Wert einer Enumerated Constant nicht verändert werden. Diese Konstante kann mit einem Label versehen werden.

Ring-Konstante

Ringe stellen Beziehungen zwischen vorzeichenlosen ganzen Zahlen und Strings her. Wenn von einer Ring-Konstante ein Wert angezeigt wird, ist die Zahl anstelle des mit ihr assoziierten Strings zu sehen. Wenn Sie einen unveränderlichen Satz von Strings benötigen, sollten Sie diese Konstante verwenden. Sie setzen den Wert, indem Sie mit dem Bedienwerkzeug in der Konstante klicken. Den String setzen Sie mit dem Beschriftungswerkzeug; geben Sie den String ein. Ein weiteres Objekt wird hinzugefügt, indem Sie auf die Konstante klicken und **Objekt davor einfügen** oder **Objekt danach einfügen** wählen.

Wenn das VI gerade abläuft, kann der Wert einer Enumerated Constant nicht verändert werden. Diese Konstante kann mit einem Label versehen werden.

Beschreibungen der Umwandlungsfunktionen

Die folgende Abbildung zeigt die auf der **Umwandlungsunterpalette** verfügbaren Optionen.

Die folgenden Funktionen konvertieren eine numerische Eingabe in eine spezielle Repräsentierung:

- In Byte Integer
- In Word-Integer
- In Long-Integer
- In vorzeichenlosen Byte-Integer-Wert
- In vorzeichenlosen Word-Integer-Wert
- In vorzeichenlosen Long-Integer
- In Single
- In Double
- In Extended
- In Extended Komplex
- In Single Komplex
- In Double Komplex

Wenn diese Funktionen eine Fließkommazahl in eine ganze Zahl (Integer) umwandeln, runden sie die Ausgabe zur nächsten ganzen Zahl bzw. der nächsten geraden Zahl, sollte der Bruchteil 0,5 betragen. Sollte das Ergebnis außerhalb des Bereiches für ganze Zahlen liegen, geben diese Funktionen den Minimal- oder den Maximalwert für den Typ ganze Zahlen aus. Wenn diese Funktionen eine ganze Zahl in eine kleinere ganze Zahl umwandeln, kopieren sie die am wenigsten signifikanten Bits, ohne auf Überfluß zu kontrollieren. Wenn sie eine ganze Zahl in eine größere ganze Zahl konvertieren, erweitern sie das Vorzeichen einer ganzen Zahl mit Vorzeichen und füllen eine vorzeichenlose ganze Zahl mit Nullen auf.

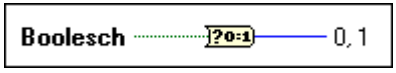
Sie müssen bei der Umwandlung von Zahlen zu kleineren Repräsentierungen mit weniger Stellen vorsichtig sein, insbesondere, wenn Sie ganze Zahlen umwandeln, weil die Konvertierungsroutinen von G nicht auf Überfluß achten.

Boolesches Array in Zahl

Konvertiert ein **Boolesches Array** in einen vorzeichenlosen Long-Integer-Wert (ganze Zahl), indem das Array als die Repräsentierung des Zweierkomplements eines Integer mit dem 0-ten. Element als dem niedrigstwertigen Bit interpretiert wird.

Boolesche in (0,1)

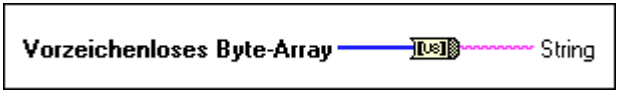
Konvertiert einen Booleschen Wert in einen Word-Integer –0 und 1 stehen für die jeweiligen Eingabewerte FALSE bzw. TRUE.



Boolesch kann ein Skalar, ein Array oder ein Cluster aus Booleschen Werten, ein Array aus Clustern aus Booleschen Werten usw. sein. Lesen Sie hierzu in Kapitel 5, *Boolesche Funktionen*, den Abschnitt *Polymorphismus für Boolesche Funktionen*.

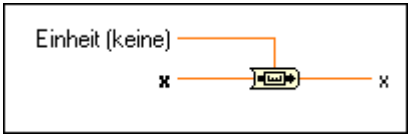
Byte-Array in String

Konvertiert ein Array aus vorzeichenlosen Bytes in einen String.



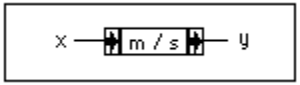
Wandelt Einheitenbasis

Ändert die mit dem Eingang assoziierten Einheiten zu den mit **Einheit** assoziierten Einheiten und gibt die Ergebnisse am Ausgangsterminal aus. Diese Funktion ist mit äußerster Vorsicht einzusetzen. Da die Funktion **Wandelt Einheitenbasis** mit den Basen arbeitet, müssen Sie zunächst die Konvertierung von einer beliebigen Einheit zu deren Basen verstehen, ehe Sie diese Funktion effektiv anwenden können. Diese Funktion kann Einheitenbasen ändern, z.B. Meter in Gramm.



Einheitenkonvertierung

Konvertiert eine physikalische Zahl (eine Zahl mit einer Einheit) in eine reine Zahl (eine Zahl ohne Einheit) oder eine reine Zahl in eine physikalische Zahl.



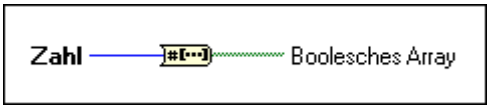
Der String kann innerhalb der Einheit bearbeitet werden, indem Sie den String mit dem Bedienwerkzeug markieren und dann den Text eingeben.

Wenn die Eingabe eine reine Zahl ist, erhält die Ausgabe die festgelegten Einheiten. Eine Eingabe von 13 und eine Einheitenspezifikation von Sekunde(n) ergibt beispielweise den Ausgabewert 13 Sekunden.

Wenn die Eingabe eine physikalische Zahl und die **Einheit** eine kompatible Einheit ist, ist die Ausgabe die Eingabe in den festgelegten Einheiten gemessen. Wenn Sie z.B. 37 m festlegen und die **Einheit** Meter lautet, ist das Ergebnis 37 ohne assoziierte Einheiten. Wenn **Einheit** ft (Fuß) lautet, ist das Ergebnis 121,36 ohne assoziierte Einheiten.

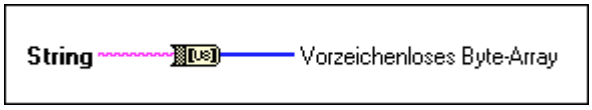
Zahl in Boolesches Array

Konvertiert eine Integer-**Zahl** in ein Boolesches Array aus 8, 16 oder 32 Elementen mit dem 0. Element als dem niedrigstwertigen Bit (LSB = least significant bit) der Repräsentierung der Zweierkomponente des Integer (der ganzen Zahl).



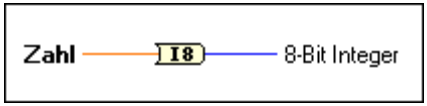
String in Byte-Array

Konvertiert **String** in ein Array aus vorzeichenlosen Bytes.



In Byte-Integer

Konvertiert **Zahl** in einen 8-Bit-Integer in dem Bereich von -128 bis 127.



In Double Komplex

Konvertiert **Zahl** in eine komplexe Zahl mit doppelter Genauigkeit.

In Double

Konvertiert **Zahl** in eine Fließkommazahl (2stellig) mit doppelter Genauigkeit.

In Extended Komplex

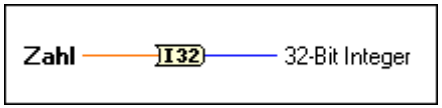
Konvertiert **Zahl** in eine komplexe Zahl mit erweiterter Genauigkeit.

In Extended

Konvertiert **Zahl** in eine Fließkommanzahl mit erweiterter Genauigkeit.

In Long-Integer

Konvertiert **Zahl** in einen 32-Bit-Integer im Bereich von -2^{31} bis $2^{31}-1$.



In Single Komplex

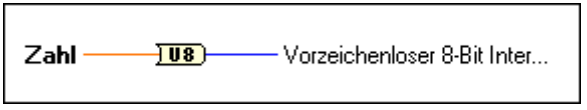
Konvertiert **Zahl** in eine komplexe Zahl mit einfacher Genauigkeit.

In Single

Konvertiert **Zahl** in eine Fließkommazahl mit einfacher Genauigkeit.

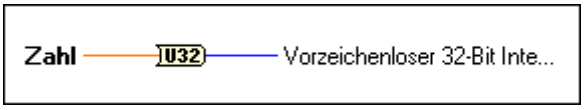
In vorzeichenlosen Byte-Integer-Wert

Konvertiert **Zahl** in einen vorzeichenlosen 8-Bit-Integer im Bereich von 0 bis 255.



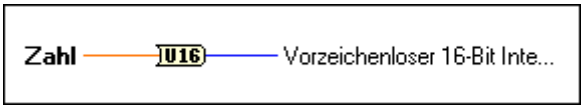
In vorzeichenlosen Long-Integer-Wert

Konvertiert **Zahl** in einen vorzeichenlosen 32-Bit-Integer im Bereich von 0 bis 2^{32} –1.



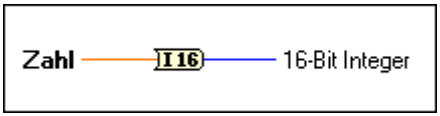
In vorzeichenlosen Word-Integer-Wert

Konvertiert **Zahl** in einen vorzeichenlosen 16-Bit-Integer im Bereich von 0 bis 65,535.



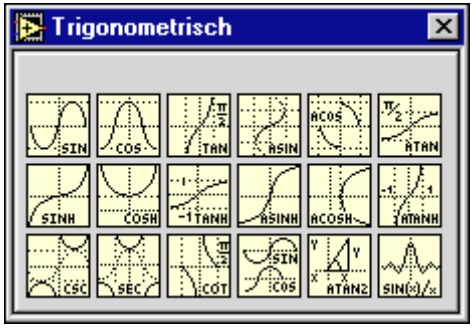
In Word-Integer-Wert

Konvertiert **Zahl** in einen 16-Bit-Integer im Bereich von –32,768 bis 32,767.



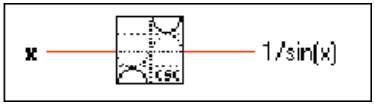
Beschreibungen der trigonometrischen und hyperbolischen Funktionen

Die folgende Abbildung zeigt die auf der **Trigonometrisch**en Unterpalette verfügbaren Optionen.



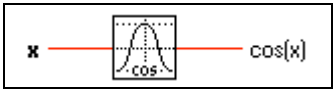
Kosekans

Berechnet den Kosekans von **x**, wobei **x** in Bogengrad angegeben ist. Kosekans ist der Kehrwert des Sinus.



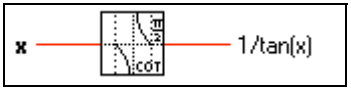
Kosinus

Berechnet den Kosinus von **x**, wobei **x** in Bogengrad angegeben ist.



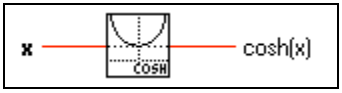
Kotangens

Berechnet den Kotangens von **x**, wobei **x** in Bogengrad angegeben ist. Der Kotangens ist der Kehrwert des Tangens.



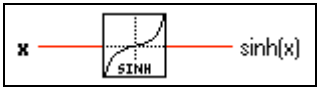
Kosinus Hyperbolicus

Berechnet den hyperbolischen Kosinus von **x**.



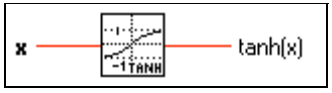
Sinus Hyperbolicus

Berechnet den hyperbolischen Sinus von **x**.



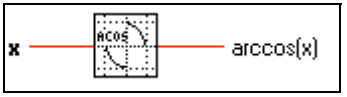
Tangens Hyperbolicus

Berechnet den hyperbolischen Tangens von **x**.



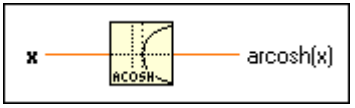
Inverser Kosinus

Berechnet den Arcuskosinus von **x** in Bogengrad. Wenn **x** nicht komplex und kleiner als -1 oder größer als 1 ist, lautet das Ergebnis Keine Zahl.



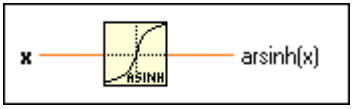
Inverser Kosinus Hyperbolicus

Berechnet den hyperbolischen Arcuskosinus von **x**. Wenn **x** nicht komplex und kleiner als 1 ist, lautet das Ergebnis Keine Zahl.



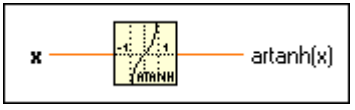
Inverser Sinus Hyperbolicus

Berechnet den hyperbolischen Arcussinus von **x**.



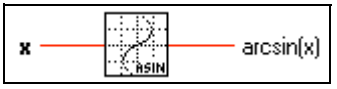
Inverser Tangens Hyperbolicus

Berechnet den hyperbolischen Arcustangens von **x**. Wenn **x** nicht komplex und kleiner als -1 oder größer als +1ist, lautet das Ergebnis Keine Zahl.



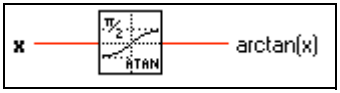
Inverser Sinus

Berechnet den Arcussinus von **x** in Bogengrad. Wenn **x** nicht komplex und kleiner als -1 oder größer als 1 ist, lautet das Ergebnis Keine Zahl.



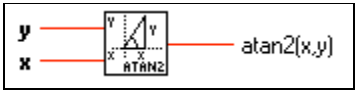
Inverser Tangens

Berechnet den Arcustangens von **x** in Bogengrad (der zwischen $-\pi/2$ und $\pi/2$ liegen kann).



Inverser Tangens (2 Eingaben)

Berechnet den Arcustangens von **y**/**x** in Bogengrad. Diese Funktion kann den Arcustangens für Winkel in jedem der vier Quadranten der x-y-Ebene berechnen, wohingegen die Funktion Inverser Tangens den Arcustangens nur in zwei Quadranten berechnet.

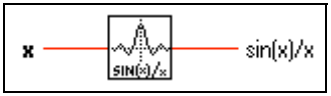


Secans

Berechnet den Secans von **x**, wobei **x** in Bogengrad angegeben ist.

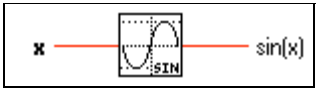
Sinc

Berechnet den Sinus von **x** dividiert durch **x**, wobei **x** in Bogengrad angegeben ist.



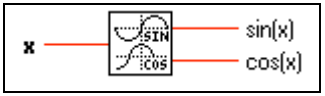
Sinus

Berechnet den Sinus von **x**, wobei **x** in Bogengrad angegeben ist.



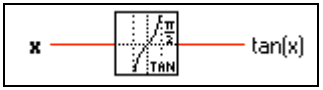
Sinus & Kosinus

Berechnet sowohl den Sinus als auch den Kosinus von **x**, wobei **x** in Bogengrad angegeben ist. Verwenden Sie diese Funktion nur, wenn Sie beide Ergebnisse benötigen.



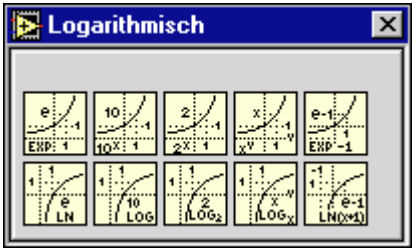
Tangens

Berechnet den Tangens von **x**, wobei **x** in Bogengrad angegeben ist.



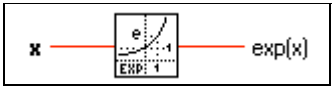
Beschreibungen der logarithmischen Funktionen

Die folgende Abbildung zeigt die auf der **Logarithmischen** Unterpalette verfügbaren Optionen.



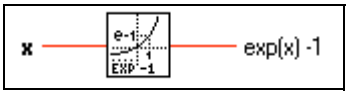
Exponential

Berechnet den Wert von e zur **x**ten Potenz erhoben.



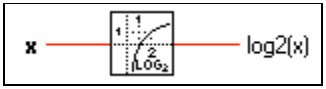
Exponential (Arg) –1

Berechnet den Wert von 1 minus e zur **x**ten Potenz erhoben. Wenn **x** sehr klein ist, liefert die Verwendung dieser Funktion genauere Ergebnisse, als wenn von der Ausgabe der Exponential Funktion eine 1 subtrahiert wird.



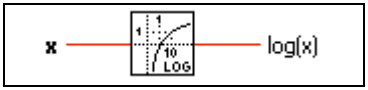
Logarithmus der Basis 2

Berechnet den Logarithmus von **x** zur Basis 2. Wenn **x** den Wert 0 hat, ist der **log2(x)** = ∞. Wenn **x** nicht komplex und kleiner als 0 ist, ist der **log2(x)** = Keine Zahl.



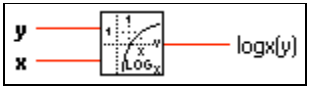
Logarithmus der Basis 10

Berechnet den Logarithmus von **x** zur Basis 10. Wenn **x** den Wert 0 hat, ist der **log(x)** = ∞. Wenn **x** nicht komplex und kleiner als 0 ist, ist der **log(x)** = Keine Zahl.



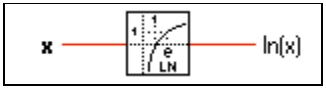
Logarithmus der Basis X

Berechnet den Logarithmus von **y** zur Basis **x** (**x**>0, **y**>0). Wenn **y** den Wert 0 hat, lautet die Ausgabe ∞. Wenn sowohl **x** als auch **y** nicht komplex sind und **x** kleiner als oder gleich 0 ist, oder **y** kleiner als 0 ist, lautet die Ausgabe Keine Zahl.



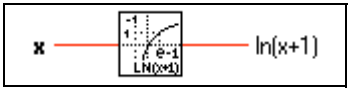
Natürlicher Logarithmus

Berechnet den Logarithmus von **x** zur natürlichen Basis e. Wenn **x** den Wert 0 hat, ist der **ln(x)** –∞. Wenn **x** nicht komplex und kleiner als 0 ist, ist der **ln(x)** = Keine Zahl.



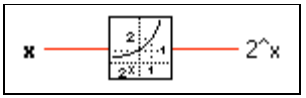
Natürlicher Logarithmus (Arg +1)

Berechnet den natürlichen Logarithmus (**x** + 1). Wenn **x** einen Wert in der Nähe von 0 besitzt, ist diese Funktion genauer, als wenn eine 1 zu **x** addiert und dann die Funktion Natürlicher Logarithmus verwendet wird. Wenn **x** gleich -1 ist, ist das Ergebnis ∞. Wenn **x** nicht komplex und kleiner als -1 ist, lautet das Ergebnis Keine Zahl.



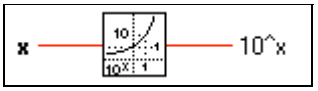
2 hoch x

Berechnet 2 zur **x**-ten Potenz erhoben.



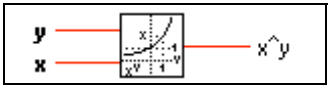
10 hoch x

Berechnet 10 zur **x**-ten Potenz erhoben.



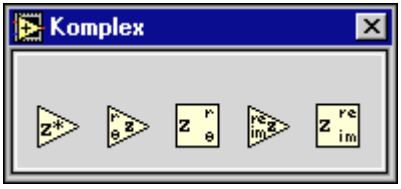
X hoch y

Berechnet **x** zur **y**ten Potenz erhoben. Wenn **x** nicht komplex ist, muß **x** größer als 0 sein, außer wenn **y** einen ganzzahligen Wert besitzt. Ansonsten lautet das Ergebnis Keine Zahl. Wenn **y** gleich Null ist, ist $\mathbf{x}^{\mathbf{y}} = 1$ für alle Werte von **x**, einschließlich Null.



Beschreibungen der Komplex-Funktionen

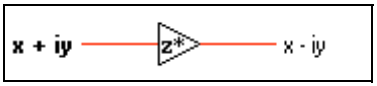
Die folgende Abbildung zeigt die auf der **Komplex**-Unterpalette verfügbaren Optionen.



Die Funktionen Polar in Komplex und Re/Im in Komplex erstellen aus zwei in kartesischer oder polarer Form gegebenen Werten komplexe Zahlen, und die Funktionen Komplex in Polar und Komplex in Re/Im teilen eine komplexe Zahl in ihre kartesischen oder polaren Komponenten auf.

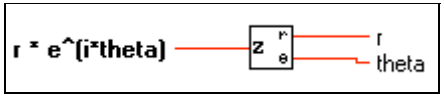
Konjugiert-komplex

Erzeugt den konjugierten Komplex **x + iy**.



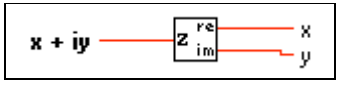
Komplex in Polar

Teilt eine komplexe Zahl in ihre Polarkomponenten auf.



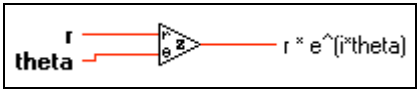
Komplex in Re/Im

Teilt eine komplexe Zahl in ihre kartesischen Komponenten auf.



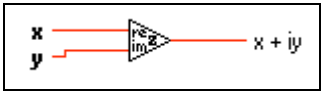
Polar in Komplex

Erstellt aus zwei als Polarausdrücke gegebenen Werten eine komplexe Zahl.



Re/Im in Komplex

Erstellt aus zwei in kartesischer Form gegebenen Werten eine komplexe Zahl.



Beschreibungen der zusätzlichen numerischen Konstanten

Die folgende Abbildung zeigt die auf der Palette **Zusätzliche numerische Konstanten** verfügbaren Optionen.

Zusätzliche benutzerspezifische Konstanten

Sie können die folgenden Konstanten festlegen.

Listenfeld Symbol-Ring-Konstante

Diese Ringkonstante vergibt an Objekte in einem Bedienelement eines Listenfeldes Symbole. Normalerweise wird diese Konstante in die Objektsymbolattribute eingebunden.

Farbfeldkonstante

Verwenden Sie diese Konstante, um einen gleichbleibenden Farbwert in das Blockdiagramm einzuspeisen. Der Wert wird durch Anklicken der Konstante mit dem Bedienwerkzeug gesetzt; wählen Sie dort die gewünschte Farbe aus.

Wenn das VI gerade abläuft, kann der Wert der Farbfeldkonstante nicht verändert werden. Sie können diese Konstante mit einem Label versehen.

Fehler-Ring-Konstante

Diese Konstante ist ein vordefinierter Ring aus Fehlern, die sich auf die Speicherbelegung, Netzwerkkommunikation, Druckvorgänge und Dateien-I/O beziehen. Fehler, die sich auf DAQ, GPIB, VISA sowie auf serielle VIs und Funktionen beziehen, gehören nicht zu den Optionen dieses Rings.

Feststehende Konstanten

Die folgenden Konstanten sind Festwerte.

Avogadrosche Konstante (1/mol)

Gibt den Wert 6,0220e23 aus.

Logarithmus von e; Basis 10

Gibt den Wert 0,43429448190325183 aus.

Elementarladung (c)

Gibt den Wert 1,6021892e–19 aus.

Gravitationskonstante (Nm^2/kg^2)

Gibt den Wert 6,6720e–11 aus.

Molare Gaskonstante (J/mol K)

Gibt den Wert 8,31441 aus.

e

Gibt den Wert 2,7182818284590452e+0 aus.

Natürlicher Logarithmus von Pi

Gibt den Wert 1,14472988584940020 aus.

Natürlicher Logarithmus von 2

Gibt den Wert 0,69314718055994531 aus.

Natürlicher Logarithmus von 10

Gibt den Wert 2,30234095236904570 aus.

Minus Unendlich

Gibt den Wert –∞ aus.

Pi

Gibt den Wert 3,14159265358979320 aus.

Pi dividiert durch 2

Gibt den Wert 1,57079632679489660 aus.

Pi multipliziert mit 2

Gibt den Wert 6,28318530717958650 aus.

Plancksche Konstante (J/Hz)

Gibt den Wert 6,6262e–34 aus.

Plus Unendlich

Gibt den Wert ∞ aus.

Kehrwert von e

Gibt den Wert 0,36787944117144232 aus.

Kehrwert von Pi

Gibt den Wert 0,31830988618379067 aus.

Rydberg-Konstante (/m)

Gibt den Wert 1,097373177e7 aus.

Lichtgeschwindigkeit im Vakuum (m/sek)

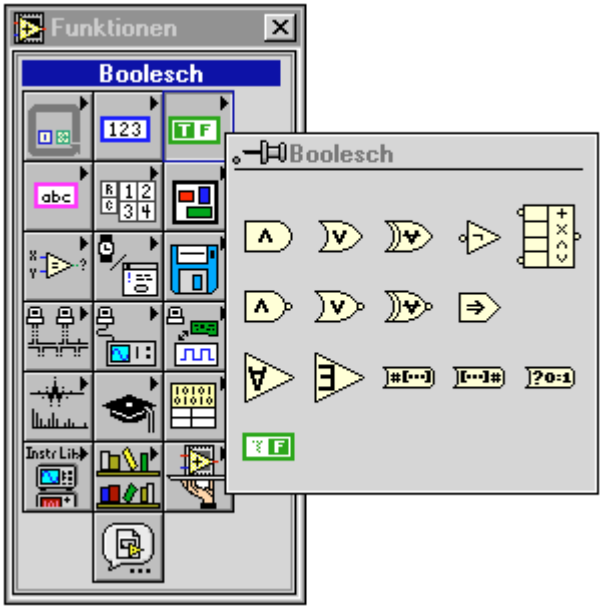
Gibt den Wert 299 792 458 aus.

5

Boolesche Funktionen

Dieses Kapitel beschreibt die Funktionen, die logische Operationen ausführen.

Die folgende Abbildung zeigt die Palette **Boolesch**, die über **Funktionen» Boolesch** aufgerufen werden kann.



Beispiele für einige der Booleschen Funktionen finden Sie in `examples\general\structs.llb`.

Polymorphismus für Boolesche Funktionen

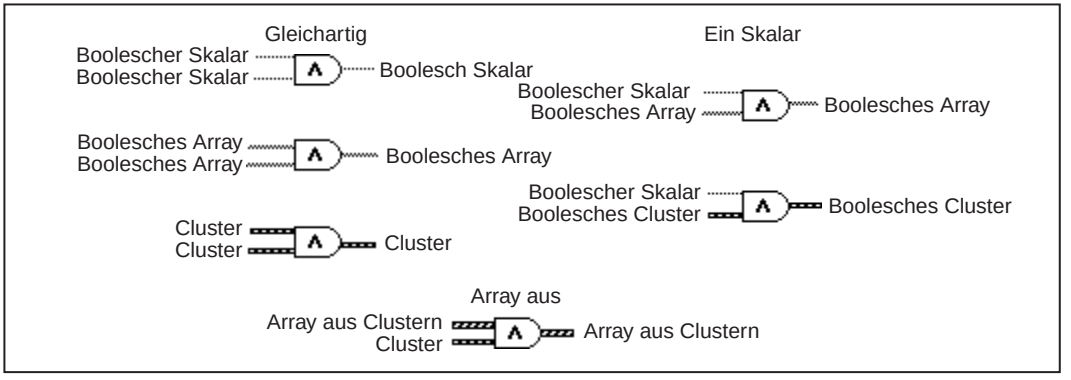
Die logischen Funktionen arbeiten mit entweder Booleschen oder numerischen Eingabedaten. Handelt es sich um eine numerische Eingabe, führt G eine bitweise Operation aus. Handelt es sich bei der Eingabe um einen Integer-Wert, hat die Ausgabe dieselbe Repräsentierung. Handelt es sich bei der Eingabe um eine Fließkommazahl, rundet G die Fließkommazahl zu einem Long-Integer, und die Ausgabe ist ein Long-Integer.

Die logischen Funktionen arbeiten mit Arrays aus Zahlen oder Booleschen Werten, Clustern aus Zahlen oder Booleschen Werten, Arrays aus Clustern aus Zahlen oder Booleschen Werten usw.

Eine formale und rekursive Definition der zulässigen Eingabetypen lautet, wie folgt:

> *Logische Typen* = Boolescher Skalar || numerischer Skalar || Array [*logischer Typ*] || Cluster [*logische Typen*]
>
> mit der Einschränkung, daß komplexe Zahlen und Arrays aus Arrays nicht zugelassen sind.

Logische Funktionen mit zwei Eingaben können dieselben Eingabekombinationen wie arithmetische Funktionen haben. Die logischen Funktionen unterliegen jedoch der Einschränkung, daß die Basisoperationen nur zwischen zwei Booleschen Werten oder zwei Zahlen durchgeführt werden können. Sie können z.B. kein UND zwischen einem Booleschen Wert und einer Zahl haben. Die nachstehende Abbildung zeigt einige Beispiele für Kombinationen von Booleschen Werten mit der arithmetischen Funktion UND.

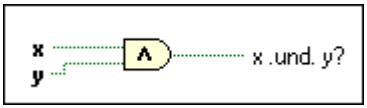


Beschreibungen der Booleschen Funktionen

Die folgenden Booleschen Funktionen stehen zur Verfügung:

Und

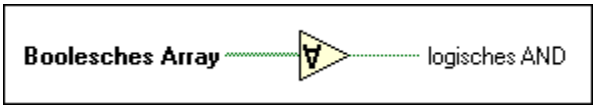
Berechnet das logische UND von Eingaben.



☞ **Hinweis** ***Diese Funktion führt bei numerischen Eingaben bitweise Operationen aus.***

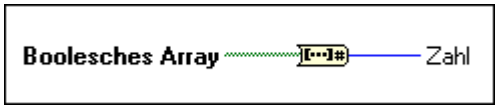
UND Array-Elemente

Gibt TRUE aus, wenn alle Elemente im **Booleschen Array** wahr sind; ansonsten wird FALSE ausgegeben.



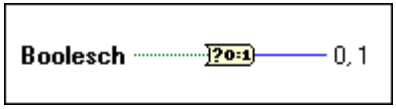
Boolesches Array in Zahl

Konvertiert ein **Boolesches Array** zu einem vorzeichenlosen Long-Integer, indem das Array als die Repräsentierung des Zweierkomplements eines Integers mit dem 0. Element des Arrays als dem niedrigstwertigen Bit interpretiert wird.



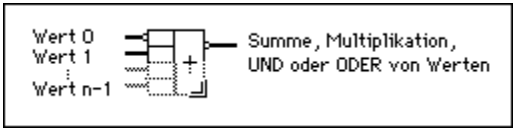
Boolesch in (0,1)

Konvertiert einen **Booleschen** Wert in einen Word-Integer, wobei 0 und 1 für die Eingabewerte FALSE und TRUE stehen.



Mehrfacharithmetik

Führt arithmetische Operationen mit zwei oder mehreren numerischen, Cluster- oder Booleschen Eingaben aus.



Sie wählen die Operation (Multiplizieren, UND oder ODER) aus, indem Sie mit der rechten Maustaste auf die Funktion klicken und **Ändert Modus** wählen.

Die Eingaben oder die Ausgaben dieser Funktionen können umgekehrt werden, indem Sie mit der rechten Maustaste auf die einzelnen Terminals klicken und **Invertieren** wählen. In der Funktion Multiplizieren können Sie durch **Invertieren** den reziproken Wert einer Eingabe verwenden oder den reziproken Wert der Ausgabe erzeugen. In den Funktionen UND oder ODER kann mit **Invertieren** eine Eingabe oder die Ausgabe logisch negiert werden.

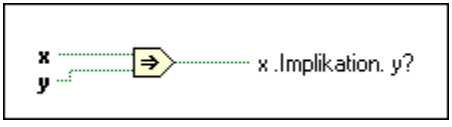
☞ **Hinweis** ***Eingänge können zu diesem Knoten hinzugefügt werden, indem Sie mit der rechten Maustaste auf einen Eingang klicken und Eingang hinzufügen wählen oder indem Sie das Positionierwerkzeug in der unteren linken oder rechten Ecke des Knotens plazieren und ziehen.***

Exklusives Oder

Berechnet das logische ausschließende ODER der Eingaben.

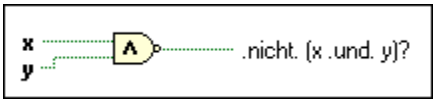
Implikation

Berechnet das logische ODER von **y** und der logischen Negation von **x**. Die Funktion negiert **x** und berechnet anschließend das logische ODER von **y** und das des negierten **x**.



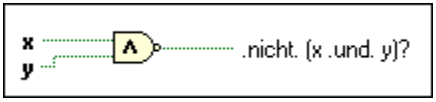
Nicht

Berechnet die logische Negation der Eingabe.



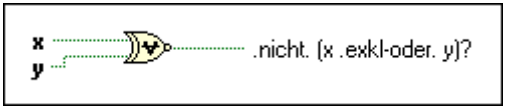
Nicht UND

Berechnet das logische NAND (die negierte Konjunktion) der Eingaben.



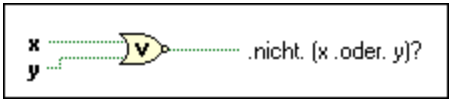
Kein Exklusives ODER

Berechnet die logische Negation des logischen ausschließenden ODER der Eingaben.



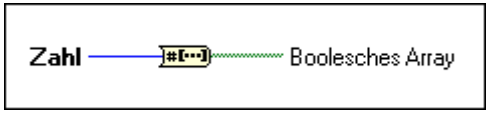
Nicht ODER

Berechnet das logische NOR (WEDER-NOCH) der Eingaben.



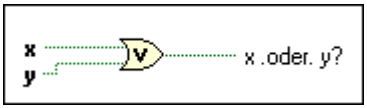
Zahl in Boolesches Array

Konvertiert **Zahl** in ein Boolesches Array aus 8, 16 oder 32 Elementen, wobei das 0. Element mit dem niedrigstwertigen Bit (LSB) der Repräsentierung des Zweierkomplements des Integers korrespondiert.



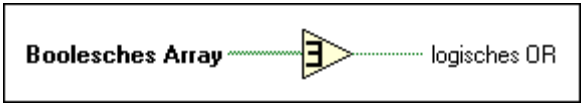
ODER

Berechnet das logische ODER der Eingaben.



ODER Arrayelemente

Gibt FALSE aus, wenn alle Elemente im **Booleschen Array** unwahr sind; ansonsten wird TRUE ausgegeben.



Boolesche Konstante

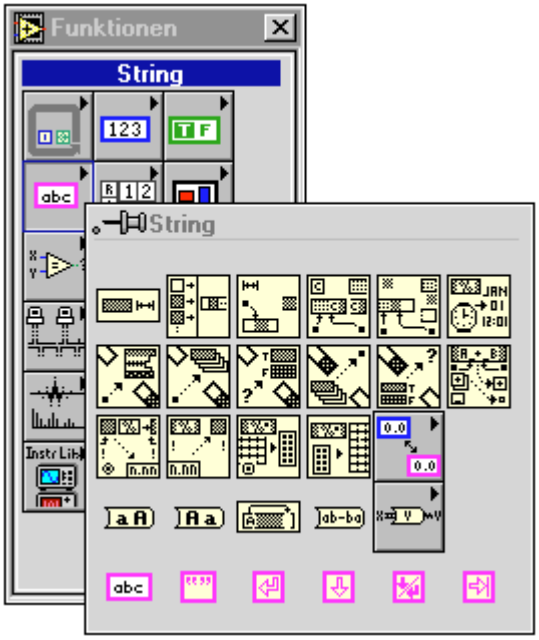
Mit dieser Funktion kann ein konstanter TRUE/FALSE-Wert in das Blockdiagramm eingespeist werden. Sie setzen diesen Wert, indem Sie mit dem Bedienwerkzeug auf den **T**- oder **F**-Teil der Konstante klicken. Wenn das VI gerade abläuft, kann dieser Wert nicht geändert werden. Diese Konstante kann mit einem Label versehen werden.

6

String-Funktionen

Dieses Kapitel beschreibt die String-Funktionen, einschließlich der Umwandlungsfunktionen Strings in Zahlen und Zahlen in Strings.

Die folgende Abbildung zeigt die **Stringpalette**, die über das Menü **Funktionen»String** aufgerufen wird.



Überblick über Polymorphismus von String-Funktionen

Dieser Abschnitt gibt Beschreibungen vom Polymorphismus für String-Funktionen, von Zusätzliche Funktionen Stringumwandlung und Funktionen Stringumwandlung.

Polymorphismus von String-Funktionen

Die Funktionen Stringlänge, In Großbuchstaben, In Kleinbuchstaben, String umkehren und String drehen akzeptieren Strings, Cluster, String- und Clusterarrays. Daneben lassen die Funktionen In Großbuchstaben und In Kleinbuchstaben auch Zahlen, Zahlencluster und Zahlenarrays zu, indem diese als Buchstaben im ASCII-Code interpretiert werden (sehen Sie hierzu im Anhang C, *GPIB mehrzeilige Schnittstellen-Nachrichten*, nach, welche Zahlen mit jedem Buchstaben korrespondieren). Die Länge- und Genauigkeitsvorgaben müssen skalar sein.

Polymorphismus von Zusätzliche Funktionen Stringumwandlung

Die Funktionen In Dezimal, In Hex, In Oktal, In technisches Format, In Dezimalbruch, In exponentielles Format akzeptieren Cluster und Arrays aus Zahlen und erzeugen Cluster und Arrays aus Strings. Die Funktionen Von Dezimal, Von Hex, Von Oktal und Von Exp./Wiss./Techn. akzeptieren Cluster und Arrays aus Strings und erzeugen Cluster und Arrays aus Zahlen. Die Länge- und Genauigkeitsvorgaben müssen skalar sein.

Polymorphismus von Funktionen Stringumwandlung

Die Funktionen Pfad zu String und String in Pfad sind polymorph. Sie arbeiten mit Skalarwerten, Skalararrays, Skalarcluster, Arrays aus Skalarclustern usw. Die Ausgabe verfügt über dieselbe Zusammensetzung wie die Eingabe, nur mit dem neuen Typ.

Überblick über Format-Strings

Viele G-Funktionen akzeptieren eine **Format-String**-Eingabe, die das Verhalten der Funktion steuert. Ein Format-String setzt sich aus einem oder mehreren Bezeichnern zusammen, die festlegen, welche Aktion auszuführen ist, um einen bestimmten Parameter zu verarbeiten. Die Funktionen In String formatieren und Aus String suchen können mehrfache Bezeichner in dem Format-String verwenden, einen für jede skalierbare Ein- oder Ausgabe zu der Funktion. Zeichen in dem String, die keinen

Teil der Bezeichners bilden, werden wörtlich in den Ausgabestring übernommen (in dem Fall von In String formatieren) oder werden mit dem Eingabestring genau verglichen (im Fall von Aus String suchen), mit Ausnahme von speziellen Steuercodezeichen. Sie können diese Codes für nichtdarstellbare Zeichen, den Backslash und das Prozentzeichen innerhalb eines jeden Format-Strings verwenden. Diese Codes ähneln den in der Programmiersprache C verwendeten.

Tabelle 6-1 zeigt die speziellen Steuercodezeichen. Es gibt kein plattformabhängiges EOL-Zeichen (end-of-line = Ende der Zeile). Wird ein solches benötigt, verwenden Sie bitte die Ende-der Zeile-Konstante von der **Stringpalette**.

Tabelle 6-1. Spezielle Steuercodezeichen

Codezeichen	Bedeutung
\r	Wagenrücklauf
\t	Tabulator
\b	Rückwärtsschritt
\n	Neue Zeile
\f	Formularvorschub
\s	Leerzeichen
\xx	Zeichen im hexadezimalen ASCII-Code xx (verwendet 0 bis 9 und Großbuchstaben A bis F)
\\	\
%%	%

Beachten Sie bitte, daß für die Funktionen Aus String suchen und Formatieren & Zerlegen jedes Leerzeichen in dem Format-String mit dem Maß an Leerflächen (Leerzeichen, Tabulatoren und Formularvorschübe) in dem Eingabestring übereinstimmt.

Die Funktionen Formatieren & Anhängen, Formatieren & Zerlegen, Array in Tabellen-String und Tabellenstring in Array verwenden im Format-String nur einen Bezeichner, weil diese Funktionen über nur eine Eingabe verfügen, die konvertiert werden kann. Alle überzähligen Bezeichner, die in diese Funktionen eingegeben werden, werden als Direktstrings ohne besondere Bedeutung behandelt.

In Funktionen, die einen String als Ausgabe erzeugen; wie z.B. In String formatieren, Formatieren & Anhängen und Array in Tabellen-String; besitzt ein Bezeichner folgende Syntax. Bei den Elementen in Doppelklammern ([ ]) handelt es sich um optionale Elemente.

```
%[–][+][^][0][Breite][.Genauigkeit][{Einheit}]Umwand-
lungscode
```

In Funktionen, die einen String durchsuchen; wie z.B. Aus String suchen, Formatieren & Zerlegen und Tabellenstring in Array; besitzt ein Bezeichner die folgende, vereinfachte Syntax:

```
%[Breite]Umwandlungscode
```

Tabelle 6-2 zeigt die verfügbare String-Syntax.

Tabelle 6-2. String-Syntax

Syntax-Element	Beschreibung
%	Steht den Formatieranweisungen voran.
– (optional)	Der Parameter wird innerhalb seiner Länge linksbündig, anstatt rechtsbündig, ausgerichtet.
+ (optional)	Gilt bei numerischen Parametern; das Vorzeichen wird eingeschlossen, selbst wenn es sich um eine positive Zahl handelt.
^ (optional)	Verwendet technisches Format, wenn zusammen mit dem `e-` oder `g-` Umwandlungscodezeichen verwendet (der Exponent ist immer ein Vielfaches von 3).
`0` (optional)	Füllt alle überzähligen Stellen links neben einem Parameter mit Nullen, anstatt mit Leerzeichen, auf.

Tabelle 6-2. String-Syntax (Fortsetzung)

Syntax-Element	Beschreibung
Breite (optional)	Gibt eine genaue, beim Durchsuchen zu verwendende Feldlänge an. Beim Verarbeiten des Parameters durchsucht G lediglich die vorgegebene Anzahl von Zeichen. Gibt die Mindestfeldlänge von Zeichen bei der Formatierung der Ausgabe an. Diese Breite ist keine Maximallänge: G verwendet so viele Zeichen, wie zur Formatierung des Parameters ohne dessen Kürzung notwendig sind. G füllt die Felder links oder rechts neben dem Parameter von der jeweiligen Ausrichtung abhängig mit Leerzeichen auf. Ist keine Länge oder die Länge mit Null vorgegeben, dann ist die Ausgabe so lang wie notwendig, um alle konvertierten Eingabeparameter zu enthalten.
.	Trennt die Länge von der Genauigkeit.
Genauigkeit (optional)	Gibt für Fließkommaparameter die Anzahl der Stellen rechts vom Dezimalpunkt an. Wenn das Syntaxelement Länge nicht mit einem Punkt abgeschlossen wurde, fügt G einen Bruchteil von sechs Ziffern ein. Wurde Länge mit einem Punkt abgeschlossen und keine Genauigkeit vorgegeben, fügt G keinen Bruchteil ein. Gibt für Stringparameter die Maximallänge des Feldes an. G kürzt Strings, die diese Länge überschreiten.
`{Einheit}` (optional)	Überschreibt die Wahleinheit eines VI, wenn eine physikalische Menge (ein Wert mit einer dazugehörigen Einheit) konvertiert wird. Es muß sich hierbei um eine gültige Einheit handeln.

Tabelle 6-2. String-Syntax (Fortsetzung)

Syntax-Element	Beschreibung
Umwandlungscode-zeichen	Einzelbuchstaben, die vorgeben, daß Parameter, wie folgt, zu durchsuchen oder zu formatieren sind: `d` dezimaler Integer (ganze Zahl) `x` hex Integer `o` oktaler Integer `b` binärer Integer `f` Fließkommazahl in gebrochenem Format `e` Fließkommazahl in wissenschaftlicher Notation `g` Fließkommazahl im `e`-Format, wenn der Exponent kleiner als –4 oder größer als die Genauigkeit ist, oder ansonsten im `f`-Format `s` String
Dezimal-Trennzeichen	Dienen, wie folgt, als Formattrennzeichen zur Begrenzung: `%,;` Komma Dezimaltrennzeichen `%.;` Punkt Dezimaltrennzeichen `%;` Standardtrennzeichen des Systems

Die Umwandlungscodezeichen in G ähneln den in der Programmiersprache C verwendeten. G verwendet die Umwandlungscodes jedoch, um das Textformat und nicht um den Datentyp eines Parameters zu bestimmen.

Die Umwandlungscodes `d`, `x`, `o`, `b`, `f`, `e` und `g` können zur Verarbeitung jedes numerischen Datentyps in G, komplexe Zahlen und Enums eingeschlossen, eingesetzt werden.

Zur Verarbeitung der realen und imaginären Teile komplexer Zahlen als einzelne Parameter können die Bezeichner eingesetzt werden.

Zur Verarbeitung von Stringparametern, Pfadparametern oder Aufzählungen kann das `s`-Umwandlungscodezeichen verwendet werden.

Beachten Sie bitte, daß für Enums sowohl ein numerisches als auch ein String-Umwandlungscodezeichen verwendet werden kann, abhängig davon, ob Sie den numerischen oder den symbolischen (String-) Wert von Enums wünschen.

Um mit C kompatibel zu sein, behandelt G ein u-Umwandlungscodezeichen (vorzeichenloser Integer) wie ein d, und ignoriert ein l oder L-Zeichen, das dem Umwandlungszeichen vorangeht. In G bestimmt jedoch der Datentyp die Größe eines Integer und, ob der Integer ein Vorzeichen trägt oder nicht.

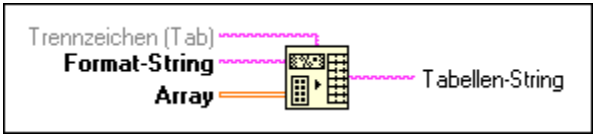
Beispiele für die Verwendung von Format-Strings befinden sich in den Beschreibungen der Funktionen In String formatieren und Aus String suchen weiter hinten im diesem Kapitel.

Beschreibungen der String Funktionen

Die folgenden String-Funktionen sind verfügbar.

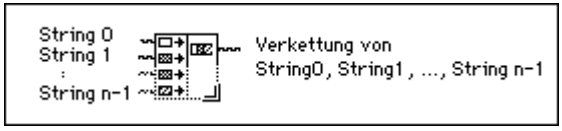
Array in Tabellen-String

Konvertiert ein **Array** mit beliebig vielen Dimensionen in einen **Tabellen-String**. Ein **Tabellen-String** ist eine Tabelle in Stringform, in der die Spaltenelemente durch ein Abgrenzungszeichen getrennt, die Reihen durch ein plattformabhängiges EOL-Zeichen gekennzeichnet und - bei Arrays mit drei und mehr Dimensionen - Seiten getrennt sind.



Strings verknüpfen

Verknüpft Eingabestrings und eindimensionale Arrays aus Strings zu einen einzelnen Ausgabestring. Bei einem Array als Eingabe verknüpft diese Funktion jedes Element des Arrays.



In String formatieren

Konvertiert Eingabe-Argumente in **resultierender String**, dessen Format von dem **Format-String** bestimmt wird. Die Anzahl der Argumente kann erhöht werden, indem Sie mit der rechten Maustaste auf den Knoten klicken und **Parameter hinzufügen** wählen oder das Positionierwerkzeug über der unteren linken oder rechten Ecke des Knotens plazieren und ihn dann dehnen, bis Sie die gewünschte Anzahl von Argumenten erhalten haben.

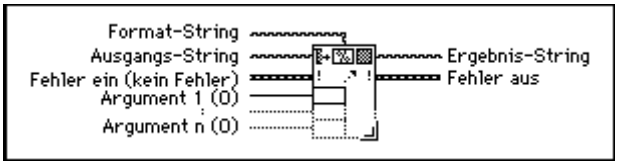


Tabelle 6-3 zeigt die Fehler, die im **Fehlerausgang** der Funktion In String formatieren angezeigt werden können.

Tabelle 6-3. Mögliche Fehler in Funktion In String formatieren

Fehler	Code	Beschreibung
Typ des Bezeichners stimmt nicht	81	Der Datentyp eines Bezeichners in dem Format-String stimmt nicht mit dem Datentyp des korrespondierenden Eingabe-Arguments überein.
Unbekannte Formatangabe (Bezeichner)	82	Der Format-String enthält einen ungültigen Bezeichner.
Zu wenige Format-Bezeichner	83	Es sind mehr Argumente als Format-Bezeichner vorhanden.
Zu viele Format-Bezeichner	84	Es sind mehr Format-Bezeichner als Argumente vorhanden.

☞ **Hinweis** ***Wenn ein Fehler eintritt, enthält die Quellkomponente des Fehlerausgabe-Clusters einen String in der Form*** "`In String formatieren (arg n)`"***, wobei n das erste Argument ist, bei dem der Fehler auftrat.***

Wenn ein Konstant-String eines Blockdiagramms mit dem **Format-String** verbunden wird, kontrolliert G während der Kompilierzeit nach Fehlern im **Format-String**. Diese Fehler müssen behoben werden, ehe das VI ausgeführt werden kann. In diesem Fall können während der Ablaufzeit keine Fehler auftreten.

Beispiele für Bezeichner (auch Format-Bezeichner oder Formatangabe)

Die in Tabelle 6-4 unterstrichenen Zeichen (_) stellen Leerzeichen in der Ausgabe dar. Die drei letzten Einträge sind Beispiele für Eingaben physikalischer Mengen.

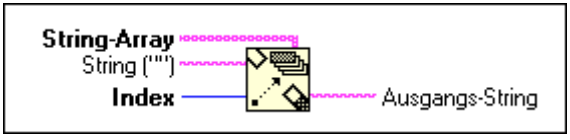
Tabelle 6-4. Bezeichner

Format-String	Argument(e)	Resultierender String
Punktezahl = %2d%%	87	Punktezahl = 87%
Ebene = \n%–7,2e V	0,03642	Ebene = 3,64e–2 V
Name: %s, %s.	Smith John	Name: Smith, John.
Temp: %05,1f %s	96,793 Fahrenheit	Temp: 096,8 Fahrenheit
String: %10,5s.	Hallo, Welt	String:_____Hallo.
%5,3f	5,67 N	5,670 N
%5,3{mN}f	5,67 N	5670,000 mN
%5,3{kg}f	5,67 N	5,670 ?kg

Der letzte Eintrag in der Tabelle zeigt den Fall an, daß die Einheit im Bezeichner mit der Eingabeeinheit in Konflikt steht.

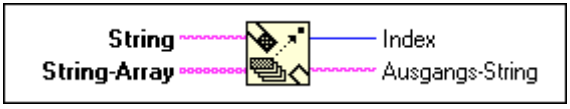
Indexieren & Anhängen

Wählt einen String, wie von dem **Index** des **String-Arrays** festgelegt, aus und hängt diesen String an den **String**.



Indexieren & Zerlegen

Vergleicht jeden String im **String-Array** mit dem Anfang des **String**, bis eine Übereinstimmung auftritt.



Pattern vergleichen

Sucht im **String** am **Offset** nach **Gültiger Ausdruck** und spaltet, wenn eine Übereinstimmung gefunden wird, den **String** in drei Substrings.

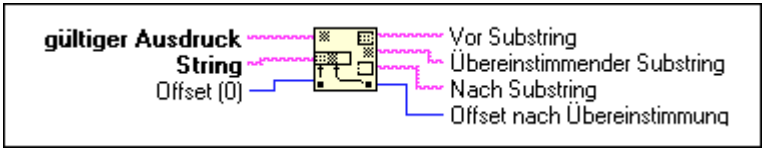


Tabelle 6-5. Sonderzeichen für die Funktion Pattern vergleichen

Sonderzeichen	Von der Funktion Pattern vergleichen interpretiert als ...
.	beliebiges Zeichen.
?	Null oder ein Beispiel des Ausdruckes der ? vorausgeht.
\	Bricht die Interpretation von Sonderzeichen ab (z.B. \? wird als Fragezeichen gelesen). Sie können außerdem auch die folgenden Konstruktionen für Leerzeichen und nichtdarstellbare Zeichen verwenden: \b Rückwärtsschritt \f Formularvorschub \n Neue Zeile \s Leerzeichen \r Wagenrücklauf *xx* beliebiges Zeichen, wobei *xx* der Hex-Code mit den Zahlen 0 bis 9 und den Großbuchstaben A bis F ist \t Tabulator

Tabelle 6-5. Sonderzeichen für die Funktion Pattern vergleichen (Fortsetzung)

Sonderzeichen	Von der Funktion Pattern vergleichen interpretiert als ...
^	Wenn ^ das erste Zeichen von **Gültiger Ausdruck** ist, wird es mit dem **Offset** des **String** verankert. Die Übereinstimmung kommt nicht zustande, außer wenn **Gültiger Ausdruck** mit dem Abschnitt des **String** übereinstimmt, die mit dem Zeichen am **Offset** beginnt. Wenn ^ nicht das erste Zeichen ist, wird es als reguläres Zeichen behandelt.
[ ]	Umschließt alle alternativen Zeichen. [abc] z.B. stimmt mit a, b oder c überein. Das folgende Zeichen hat eine besondere Bedeutung, wenn es in eckigen Klammern verwendet wird: – (Bindestrich) Zeigt einen Bereich an, wenn er zwischen Zahlen oder großen oder kleinen Buchstaben eingesetzt wird (z.B. [0−5],[a−g] oder [L−Q]). Die folgenden Zeichen haben nur dann eine besondere Bedeutung, wenn sie das erste Zeichen in der Klammer bilden: ~ Schließt den Satz von Zeichen, einschließlich nichtdarstellbarer Zeichen, aus. [~0−9] stimmt mit jedem anderen Zeichen als 0 bis 9 überein. ^ Schließt den Satz mit Hinblick auf alle darstellbaren Zeichen (und den Leerzeichen) aus. [^0−9] ergibt die Leerzeichen sowie alle darstellbaren Zeichen, außer 0 bis 9.
+	Stimmt mit der längsten Zahl von Beispielen des Ausdrucks vor + überein; es muß mindestens ein Beispiel gegeben werden, damit eine Übereinstimmung eintritt.
*	Stimmt mit der längsten Zahl der Beispiele des Ausdrucks vor * in **Gültiger Ausdruck**, einschließlich keiner Beispiele, überein.
$	Wenn $ das letzte Zeichen von **Gültiger Ausdruck** ist, verankert es die Übereinstimmung mit dem letzten Element von **String**. Die Übereinstimmung versagt, außer wenn **Gültiger Ausdruck** mit dem String zusammenpaßt und auch das letzte Zeichen in dem String beinhaltet. Wenn $ nicht das letzte Zeichen ist, wird es als reguläres Zeichen behandelt.

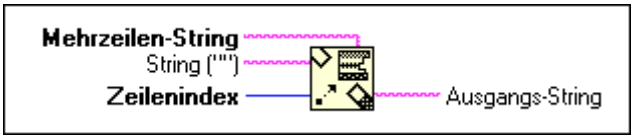
Tabelle 6-6 zeigt Beispiele von Strings für die Funktion Pattern vergleichen.

Tabelle 6-6. Strings für die Funktion Pattern vergleichen

Vorgegebene Zeichen	Regulärer Ausdruck
VOLT	`VOLT`
Alle Versionen von Volt in Groß- und Kleinbuchstaben, das heißt, VOLT, Volt, volt usw.	`[Vv][Oo][Ll][Tt]`
Ein Leerzeichen, ein Pluszeichen oder ein Minuszeichen.	`[ +–]`
Eine Sequenz von einer oder mehreren Ziffern.	`[0–9]+`
Keine oder mehrere Leerzeichen.	`\s*` oder `*` (das heißt, ein Leerzeichen von einem Sternchen gefolgt)
Ein oder mehrere Leerzeichen, Tabulatoren, neue Zeilen oder Wagenrückläufe	`[\t \r \n \s]+`
Ein oder mehrere Zeichen, die keine Ziffern sind.	`[~0–9]+`
Das Wort Ebene, nur wenn es in der Offsetposition des String anfängt.	`^Ebene`
Das Wort Volt, nur wenn es am Ende des Strings auftritt.	`Volt$`
Der längste String innerhalb der Klammern.	`(.*)`
Der längste String innerhalb von Klammern, der aber keine Klammern in sich selbst enthält.	`([~( )]*)`
Das Zeichen [	`[ [ ]`

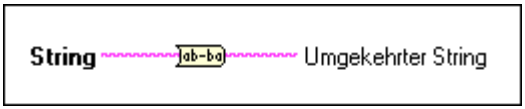
Zeile auswählen & anhängen

Wählt aus einem **Mehrzeilen-String** eine Zeile und hängt diese an den **String**.



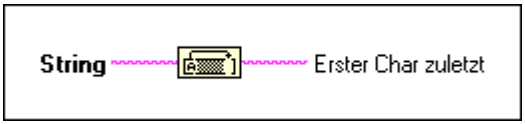
String umkehren

Erzeugt einen String, dessen Zeichen sich in der umgekehrten Reihenfolge von denen im **String** befinden.



String drehen

Plaziert das erste Zeichen des **String** in die letzte Position von **Erster Char zuletzt**, wodurch die anderen Zeichen um eine Stelle nach vorne geschoben werden. Der String *abcd* wird z.B. zu *bcda*.



Aus String suchen

Durchsucht den Eingabestring und konvertiert ihn entsprechend dem **Format-String**. Die Anzahl der Parameter kann erhöht werden, indem Sie mit der rechten Maustaste auf den Knoten klicken und **Parameter hinzufügen** wählen oder das Positionierungswerkzeug über der unteren linken oder rechten Ecke des Knotens plazieren und ihn dann dehnen, bis Sie die gewünschte Anzahl von Parametern erhalten haben.

Verwenden Sie die Funktion Aus String suchen, wenn Sie das genaue Format des Eingabestrings kennen.

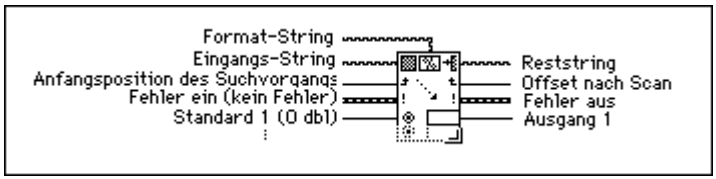


Tabelle 6-7 listet die Fehler der Funktion Aus String suchen auf.

Tabelle 6-7. Fehler der Funktion Aus String suchen

Fehler	Code	Beschreibung
Typ des Bezeichners stimmt nicht	81	Der Datentyp eines Bezeichners in dem Format-String stimmt nicht mit dem Datentyp der korrespondierenden Ausgabe überein.
Unbekannte Formatangabe (Bezeichner)	82	Der Format-String enthält einen ungültigen Bezeichner.
Zu wenige Format-Bezeichner	83	Es sind mehr Argumente als Format-Bezeichner vorhanden.
Zu viele Format-Bezeichner	84	Es sind mehr Format-Bezeichner als Argumente vorhanden.
Scan hat versagt	85	Die Funktion Aus String suchen konnte den Eingabestring nicht in den vom Bezeichner vorgegebenen Datentyp konvertieren.

☞ **Hinweis** ***Wenn ein Fehler eintritt, enthält die Quellenkomponente des Fehlerausgangsclusters einen String in der Form*** **"**`Aus String suchen (arg n)`**"*, wobei n das erste Argument ist, bei dem der Fehler aufgetreten ist.***

Wenn eine Stringkonstante eines Blockdiagramms mit dem Format-String verbunden wird, kontrolliert G während der Kompilierzeit auf Fehler im Format-String. Diese Fehler müssen zunächst behoben werden, ehe das VI ablaufen kann. In diesem Fall kann nur der Scan-hat-versagt-Fehler während der Ausführzeit auftreten.

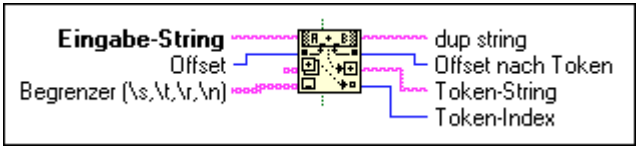
Tabelle 6-8 listet Beispiele von Aus String suchen auf.

Tabelle 6-8. Beispiele der Funktion Aus String suchen

Eingabestring	Format-String	Standard(s)	Ausgabe(n)	Verbleibender String
abc xyz 12.3+56i 7200	%s %s%f%2d	— — 0&0*i* (CDB) —	abc xyz 12.3+56i 72	00
Q+1.27E–3 tail	Q%f t	—	1.27E–3	ail
0123456789	%3d%3d	—	12 345	6789
X:9.860 Z:3.450	X:%fY:%f	100 (I32) 100.0 (DBL)	10 100.0	Z: 3450
set49.4.2	set%d	—	49	.4.2

String nach Token durchsuchen

Durchsucht am **Offset** beginnend den **Eingabe-String** und gibt das nächste Token aus, das aufgefunden wird.

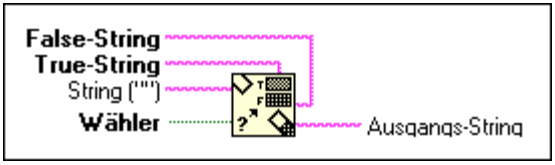


Ein *Token* ist ein Substring vom **Eingabestring**, der von Abgrenzungszeichen eingeschlossen ist oder ein Element mit **Operatoren** übereinstimmt. Normalerweise repräsentieren Token individuelle Schlüsselwörter, numerische Werte oder **Operatoren**, die gefunden werden, wenn eine Konfigurationsdatei oder ein anderes textbasiertes Datenformat analysiert wird. Diese Funktion durchsucht beim Offset beginnend Eingabestrings und gibt das nächste Token aus, das aufgefunden wird.

Weitere Informationen über Im String nach Token suchen und dessen Parameter finden Sie in der Online-Referenz.

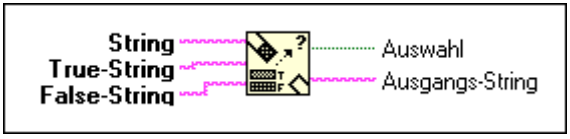
Auswählen & Anhängen

Wählt entsprechend der BOOLESchen **Auswahl** entweder **False-String** oder **True-String** und hängt diesen String an den **String** an.



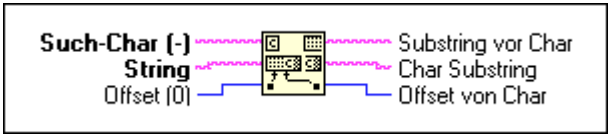
Auswählen & Zerlegen

Untersucht den Anfang von **String**, um zu bestimmen, ob er mit **True-String** oder **False-String** übereinstimmt. Diese Funktion gibt an **Auswahl** einen Booleschen TRUE- oder FALSE-Wert aus, abhängig davon, ob **String** mit dem **True-String** oder **False-String** übereinstimmt.



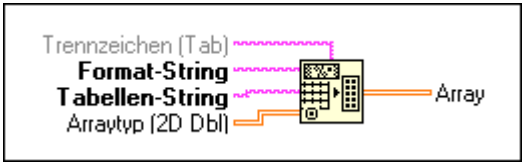
String teilen

Teilt den String am Offset oder sucht am Offset beginnend nach dem ersten Auftreten von **Such-Char** in **String** und teilt den String an diesem Punkt.



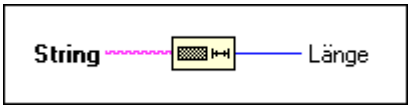
Tabellen-String in Array

Konvertiert **Tabellen-String** in ein numerisches **Array** mit den Dimensionen und der Repräsentierung vom **Arraytyp**. Diese Funktion arbeitet sowohl mit String- als auch mit Zahlenarrays.



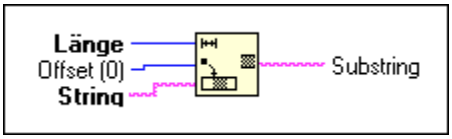
String-Länge

Gibt in **Länge** die Anzahl der Zeichen (Bytes) von **String** aus.



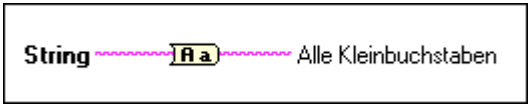
String-Subset

Gibt vom ursprünglichen **String** einen **Substring** aus, der am **Offset** anfängt und die in **Länge** festgelegte Anzahl von Zeichen enthält.



In Kleinbuchstaben

Konvertiert alle alphabetischen Zeichen im **String** zu Kleinbuchstaben. Nichtalphabetische Zeichen bleiben von dieser Funktion unbeeinflußt.



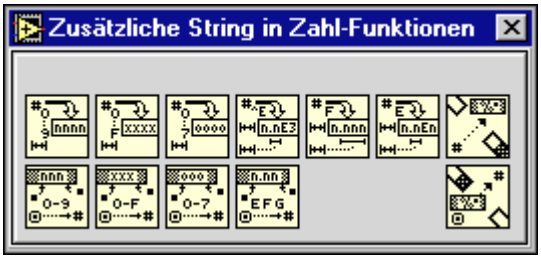
In Großbuchstaben

Konvertiert alle alphabetischen Zeichen im **String** in Großbuchstaben. Nichtalphabetische Zeichen bleiben von dieser Funktion unbeeinflußt.

Beschreibungen von Zusätzliche Funktionen Stringumwandlung

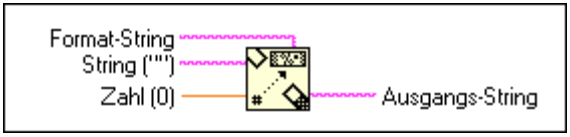
Lesen Sie bitte für allgemeine Informationen über Zusätzliche Funktionen Stringumwandlung den Abschnitt *Polymorphismus von Zusätzliche Funktionen Stringumwandlung* weiter vorn in diesem Kapitel.

Die folgende Abbildung zeigt die auf der Unterpalette **Zusätzliche String in Zahl-Funktionen** verfügbaren Optionen.



Formatieren & Anhängen

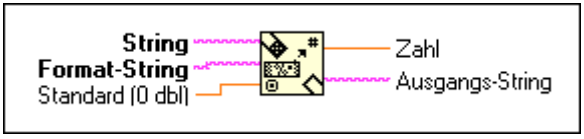
Konvertiert **Zahl** entsprechend dem im **Format-String** festgelegten Format in einen regulären String und hängt das Ergebnis an **String** an.



☞ **Hinweis** ***Die Funktion In String formatieren verfügt über dieselbe Funtionalität wie Formatieren & Anhängen, kann jedoch mehrfache Eingaben verarbeiten und kann somit Informationen gleichzeitig konvertieren. Erwägen Sie daher die Verwendung der Funktion In String formatieren statt dieser, um dadurch Ihr Blockdiagramm zu vereinfachen.***

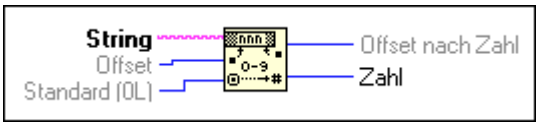
Formatieren & Zerlegen

Sucht am Anfang von **String** nach **Format-String**; formatiert jede Zahl in diesem String-Abschnitt entsprechend dem Umwandlungscode im **Format-String**, und gibt die konvertierte Zahl in **Zahl** sowie den nach der Übereinstimmung verbleibenden Teil von **String** im **Ausgangs-String** aus.



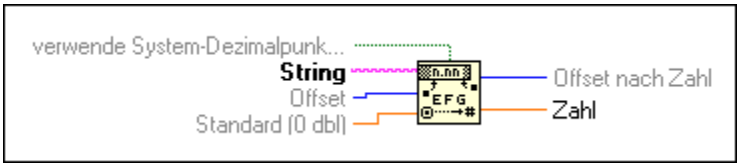
Von Dezimal

Konvertiert die numerischen Zeichen im **String** am Offset beginnend in einen dezimalen Integer-Wert und gibt ihn in **Zahl** aus.



Von Exp./Wiss./Techn.

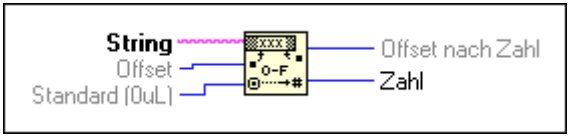
Interpretiert die Zeichen 0 bis 9, Plus, Minus, e, E und den Dezimalpunkt (meistens ein Punkt) im **String** am **Offset** beginnend als eine Fließkommazahl in technisches, exponentiales oder wissenschaftliches Format und gibt diese in **Zahl** aus.



☞ **Hinweis** ***Wenn Sie die Zeichen Unendlich oder Keine Zahl mit String verbinden, gibt diese Funktion die Werte, die Unendlich und Keine Zahl in G besitzen, aus.***

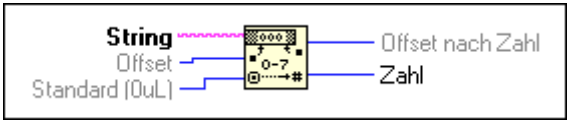
Von Hexadezimal

Interpretiert die Zeichen 0 bis 9, A bis F und a bis f im **String** am **Offset** beginnend als einen Hex-Integer-Wert und gibt diesen in **Zahl** aus.



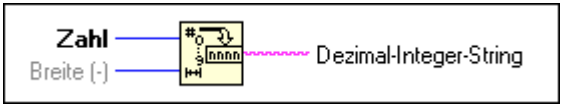
Von Oktal

Interpretiert die Zeichen 0 bis 7 im **String** am **Offset** beginnend als einen oktalen Integer-Wert und gibt ihn in **Zahl** aus. Diese Funktion gibt auch den Index des ersten Zeichens nach der Zahl im **String** aus.



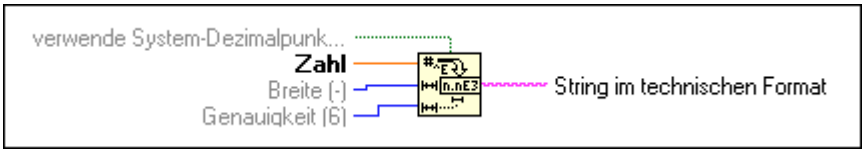
In Dezimal

Konvertiert **Zahl** in einen String aus dezimalen Ziffern mit der in **Breite** festgelegten Zeichenanzahl oder, falls erforderlich, länger.



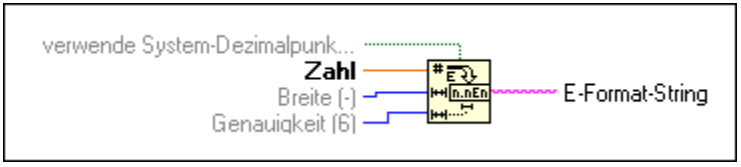
String im technischen Format

Konvertiert **Zahl** in einen Fließkommastring im technischen Format mit der in **Länge** festgelegten Zeichenanzahl oder, falls erforderlich, länger. Das technische Format ähnelt dem E-Format, nur daß der Exponent ein Vielfaches von 3 ist (-3, 0, 3. 6).



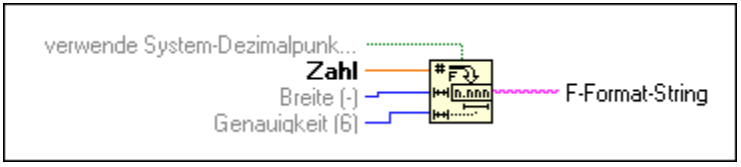
In Exponential

Konvertiert **Zahl** in einen Fließkommastring im E-Format (exponentiale Schreibweise), mit der in **Breite** festgelegten Zeichenzahl oder, falls erforderlich, länger.



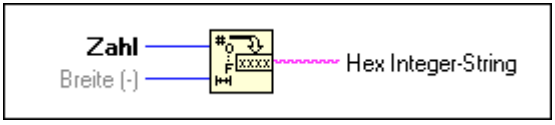
In Dezimalbruch

Konvertiert **Zahl** in einen Fließkommastring im F-Format (in Dezimalbruch-Schreibweise), mit der in **Breite** festgelegten Zeichenanzahl oder, falls erforderlich, länger.



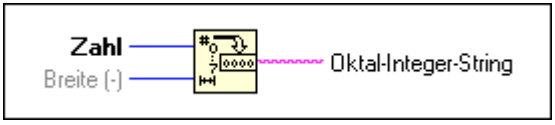
In Hexadezimal

Konvertiert **Zahl** in einen String aus hexadezimalen Ziffern mit der in **Breite** festgelegten Zeichenanzahl oder, falls erforderlich, länger.



In Oktal

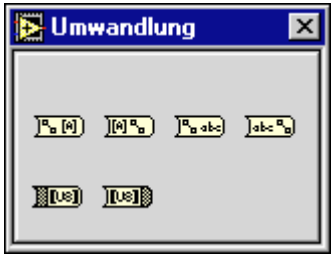
Konvertiert **Zahl** in einen String aus oktalen Ziffern mit der in **Breite** festgelegten Zeichenanzahl, oder, falls erforderlich, länger.



Beschreibungen der Funktion Stringumwandlung

Lesen Sie bitte für allgemeine Informationen die Funktionen Stringumwandlung den Abschnitt *Überblick über Polymorphismus von String-Funktionen* weiter vorn in diesem Kapitel.

Die folgende Abbildung zeigt die Unterpalette **String»Umwandlung**.

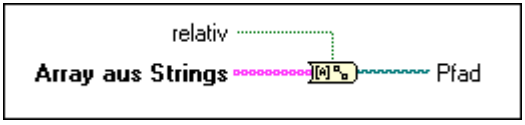


Die Funktion Array aus String in Pfad akzeptiert eindimensionale (1D) Arrays aus Strings; Pfad in Array aus Strings akzeptiert Pfade, und String in Pfad akzeptiert Strings.

Array aus Strings in Pfad

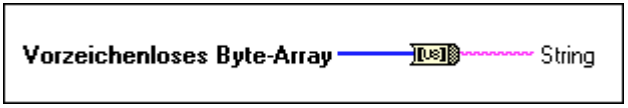
Konvertiert **Array aus Strings** in einen relativen oder absoluten **Pfad**.

Wenn das Array einen leeren String enthält, wird die Verzeichnisangabe vor dem leeren String in der Pfadausgabe gelöscht. Stellen Sie sich den Wechsel so vor, als ob Sie in der Verzeichnishierarchie eine Ebene höher gingen.



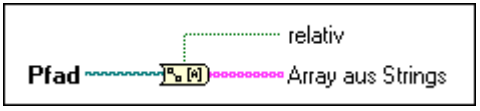
Byte-Array in String

Konvertiert ein Array aus vorzeichenlosen Byte in einen String.



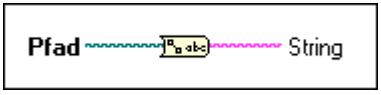
Pfad in Array aus Strings

Konvertiert **Pfad** in einen **Array aus Strings** und zeigt an, ob der Pfad **relativ** ist.



Pfad in String

Konvertiert **Pfad** in einen String, wobei der Pfad im Standardformat der Plattform beschrieben ist.

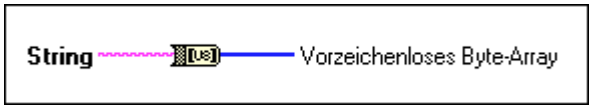


Refnum in Pfad

Gibt den mit der angegebenen **Refnum** assoziierten Pfad aus.

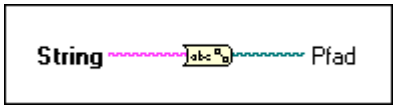
String in Byte-Array

Konvertiert **String** in ein Array aus vorzeichenlosen Bytes.



String in Pfad

Konvertiert einen String in einen Pfad, wobei der Pfad in dem Standardformat der aktuellen Plattform beschrieben ist.



Feststehende Stringkonstanten

Die folgenden feststehenden Stringkonstanten sind verfügbar:

String-Konstante

Verwenden Sie diese Konstante, um einen konstanten ASCII-String ins Blockdiagramm einzuspeisen. Dieser String wird gesetzt, indem Sie mit dem Bedienwerkzeug innerhalb der Konstante klicken und den Wert eingeben. Sie können den Anzeigemodus ändern, so daß Sie nichtdarstellbare Zeichen oder das Hex-Äquivalent der Zeichen sehen können. Sie können außerdem die Konstante in Paßwortanzeigemodus setzen, so daß Sternchen (*) angezeigt werden, wenn Sie Zeichen eingeben.

Wenn das VI abläuft, kann der Wert der Stringkonstante nicht geändert werden. Sie können die Konstante mit einem Label versehen.

Wagenrücklauf

Besteht aus einem konstanten String, der den ASCII-Wert CR (carriage return) enthält.

Leerer String

Besteht aus einem Konstantenstring, der leer ist. Seine Länge beträgt Null.

Ende der Zeile

Besteht aus einem Konstantenstring, der einen plattformabhängigen Ende-der-Zeile-Wert enthält. Der Wert unter Windows ist CRLF; unter Macintosh ist er CR, und unter UNIX ist er LF.

Zeilenvorschub

Besteht aus einem Konstantenstring, der den ASCII-Wert LF (line feed) enthält.

Tabulator

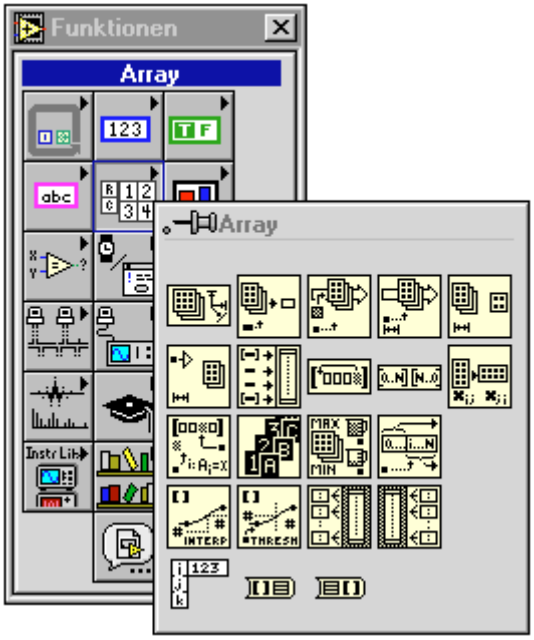
Besteht aus einem Konstantenstring, der den ASCII-Wert HT (horizontal Tab) enthält.

7

Array-Funktionen

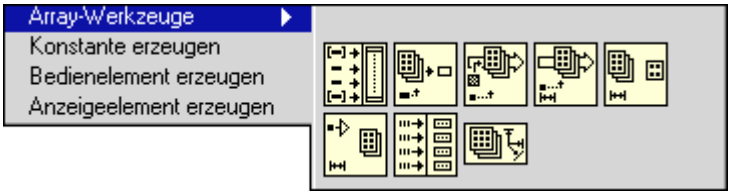
Dieses Kapitel behandelt die Funktionen für Array-Operationen.

Die folgende Abbildung zeigt die **Array**-Palette, die über das Menü **Funktionen»Array** aufgerufen werden kann.



Einige der Array-Funktionen stehen außerdem auf der **Array-Werkzeuge**-Palette von der Mehrzahl der Popup-Menüs

der Anschlüsse oder Verbindungen zur Verfügung. Die nachfolgende Abbildung zeigt dieses Popup-Menü.



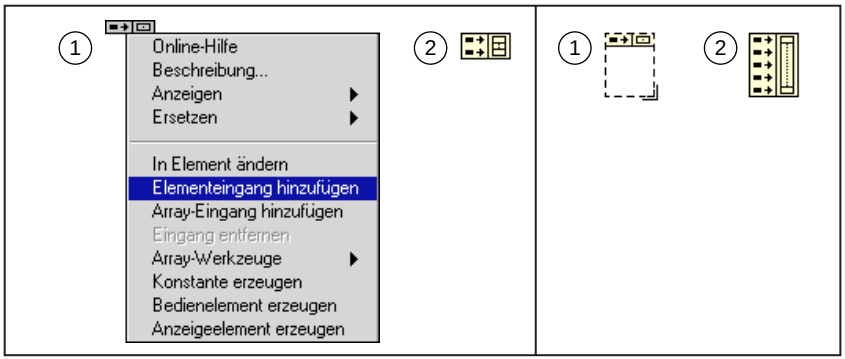
Wenn Sie Funktionen von dieser Palette auswählen, werden diese mit der richtigen Anzahl von Anschlüssen angezeigt, die an dem Objekt, dessen Popup-Menü aufgerufen wurde, verbunden werden müssen.

Beispiele für Array-Funktionen finden Sie in `examples\general\arrays.llb`.

Überblick über die Array-Funktionen

Einige der Array-Funktionen besitzen eine unterschiedliche Anzahl von Anschlüssen. Wenn Sie eine neue Funktion dieser Art auf dem Blockdiagramm ablegen, wird es mit nur einem oder zwei Anschlüssen angezeigt. Es können weitere Anschlüsse hinzugefügt und entfernt werden, indem Sie mit der rechten Maustaste das Popup-Menü aufrufen und die Befehle **Elementeingabe hinzufügen** oder **Array-Eingabe hinzufügen** oder **Eingabe entfernen** (die tatsächlichen Bezeichnungen hängen von der jeweiligen Funktion ab) wählen oder indem Sie von irgendeiner Ecke aus die Größe des Knotens senkrecht neu einstellen. Wenn Sie neue Anschlüsse durch das Popup-Menü hinzufügen möchten, muß der Zeiger auf den Eingabeanschlüssen plaziert werden, um das Popup-Menü aufzurufen.

Sie können den Knoten verkleinern, wenn dadurch keine verbundenen Anschlüsse gelöscht werden. Die Befehle **Elementeingabe hinzufügen** oder **Array-Eingabe hinzufügen** fügen einen Anschluß unmittelbar hinter dem Anschluß ein, dessen Popup-Menü Sie aufgerufen haben. Der Befehl **Eingabe entfernen** entfernt den Anschluß, dessen Popup-Menü Sie aufgerufen haben, selbst wenn dieser verbunden ist. Die folgende Abbildung zeigt die zwei Wege, wie mehr Anschlüsse zu der Funktion **Array erstellen** hinzugefügt werden können.



Indizes für Werte außerhalb des Bereichs

Der Versuch, ein Array über dessen Grenzen hinaus zu indizieren, führt zu einem Standardwert, der von dem Typ des Array-Elements bestimmt wird.

Polymorphismus von Array-Funktionen

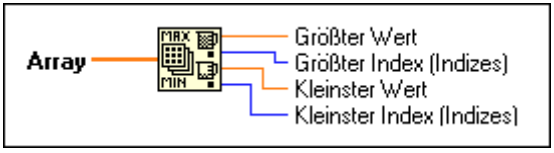
Die Mehrzahl der Array-Funktionen akzeptiert *n*-dimensionale Arrays von beliebigem Typ; die Verbindungsdiagramme in den Funktionsbeschreibungen zeigen jedoch numerische Arrays als Standarddatentypen an.

Beschreibungen der Array-Funktionen

Die folgenden Array-Funktionen sind verfügbar.

Max & Min Array

Sucht nach den ersten Maximal- und Minimalwerten in einem numerischen **Array**. Diese Funktion gibt außerdem den Index oder die Indizes aus, wo die Maximal- und Minimalwerte gefunden wurden.



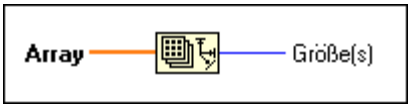
Wenn ein numerisches **Array** eine Dimension besitzt, sind die Ausgaben für **Größter Index** und **Kleinster Index** skalare Integer-Werte. Besitzt ein numerisches **Array** mehr als eine

Dimension, bestehen die Ausgaben aus 1D-Arrays, die die Indizes der Maximal- und Minimalwerte enthalten.

Diese Funktion vergleicht jeden Datentyp entsprechend den in Kapitel 9, *Vergleichsfunktionen*, beschriebenen Regeln.

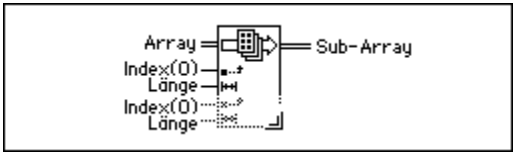
Array-Größe

Gibt die Anzahl der Elemente in jeder Dimension des **Array** aus.



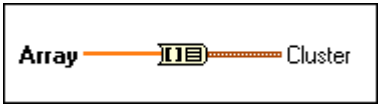
Array-Subset

Gibt einen Teil vom **Array** aus, der am **Index** beginnt und die in **Länge** festgelegte Anzahl von Elementen enthält.



Array in Cluster

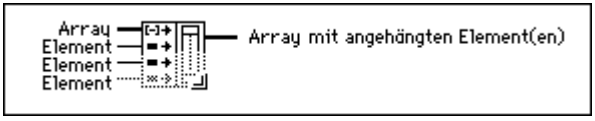
Konvertiert ein 1D-Array in einen Cluster mit demselben Elemententyp wie dem Array-Element. Die Anzahl der Elemente im Cluster kann festgelegt werden, indem Sie mit der rechten Maustaste das Popup-Menü des Knotens aufrufen. Der Standardwert beträgt 9, Die maximale Clustergröße in dieser Funktion liegt bei 256.



Lesen Sie bitte für weitere Informationen über Cluster das Kapitel 8, *Cluster-Funktionen*.

Array erstellen

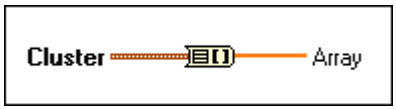
Hängt jede Anzahl von Array- oder Elementen-Eingaben in der Reihenfolge von oben nach unten an, um ein **Array mit angehängten Elementen** zu erstellen.



Um einen Elementeingang in einen Arrayeingang umzuwandeln, müssen Sie das Popup-Menü aufrufen und **In Array ändern** wählen. Um ein Array mit *n*-Dimensionen zu erstellen, muß jedes **Array** generell dieselbe Anzahl von *n* Dimensionen besitzen und jeder **Element**eingang *n*-1 Dimensionen. Um ein 1D Array zu erstellen, müssen Sie Skalarwerte mit den Elementeingängen und 1D Arrays mit Arrayeingängen verbinden. Um ein 2D Array zu erstellen, müssen Sie 1D Array mit Elementeingängen und 2D Arrays mit den Arrayeingängen verbinden.

Cluster in Array

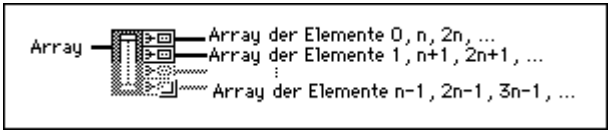
Konvertiert einen Cluster aus identisch eingegeben Komponenten in ein 1D-Array aus Elementen desselben Typs.



Für weitere Informationen über Cluster lesen Sie bitte Kapitel 8, *Cluster-Funktionen*.

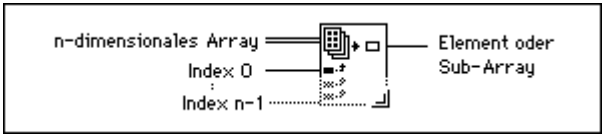
1D Array dezimieren

Teilt die Elemente von **Array** in die Ausgabe Arrays.



Array indizieren

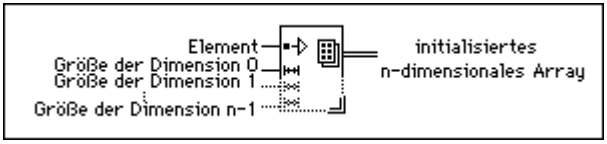
Gibt das **Element** am **Index** eines **Array** aus. Wenn **Array** mehrdimensional ist, müssen Sie zusätzliche **Index**anschlüsse für jede **Array**-Dimension hinzufügen.



Zusätzlich zum Extrahieren eines Array-Elements können Sie eine Komponente mit höheren Dimensionen herausschneiden, indem Sie einen oder mehrere Indexanschlüsse deaktivieren.

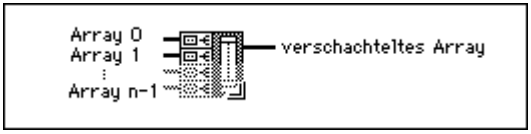
Array initialisieren

Erstellt ein *n*-dimensionales Array, in dem jedes Element mit dem Wert von **Element** initialisiert wird.



1D Arrays überführen

Überführt korrespondierende Elemente der Eingabe-Arrays in ein einziges Ausgabe-Array.

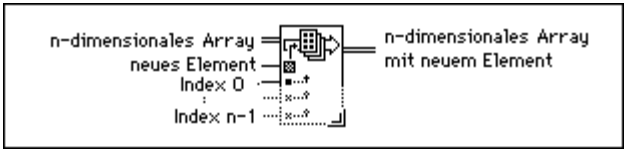


Interpoliert 1D Array

Verwendet den Integer-Teil von **gebrochener Index oder x**, um das Array zu indizieren, und den Bruchteil von **gebrochener Index oder x**, um die Werte des indexierten Elements und dessen benachbartes Element zu interpolieren.

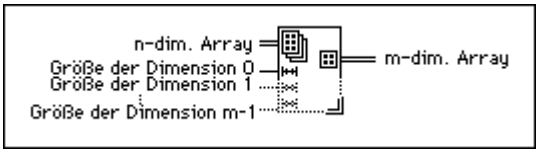
Array-Elemente ersetzen

Ersetzt das Element am **Index** von **Array** mit **Neues Element**.



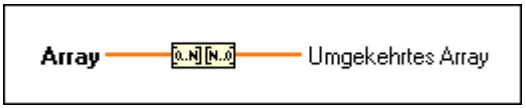
Array umformen

Ändert die Dimensionen eines Arrays entsprechend dem Wert von **Dimensionsgröße**. Die Funktion ist nachstellbar; **M-Dim Array** besitzt eine Dimension für jede Eingabe von **Dimensionsgröße**. Sie können diese Funktion z.B. dazu verwenden, ein 1D Array in ein 2D Array oder umgekehrt umzuwandeln. Mit dieser Funktion können Sie außerdem die Größe eines 1D Arrays vergrößern und verkleinern.



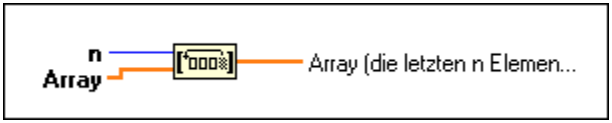
1D Array umkehren

Kehrt die Reihenfolge der Elemente in **Array** um.



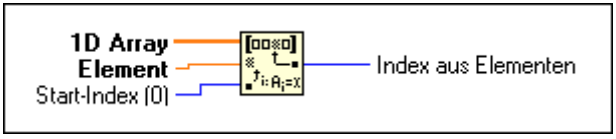
1D Array drehen

Dreht die Elemente von **Array** mit der von **n** vorgegebenen Stellenanzahl und Richtung um.



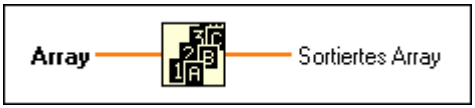
1D Array durchsuchen

Durchsucht **1D Array** mit dem **Start-Index** beginnend nach **Element**.



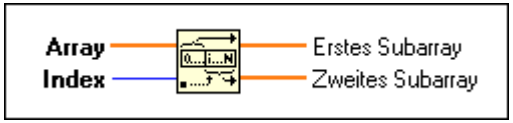
1D Array sortieren

Gibt eine sortierte Version von **Array** aus, in der die Elemente in ansteigender Reihenfolge angeordnet sind. Die Regeln zum Vergleich jeden Datentyps sind im Kapitel 9, *Vergleichsfunktionen*, beschrieben.



1D Array zerlegen

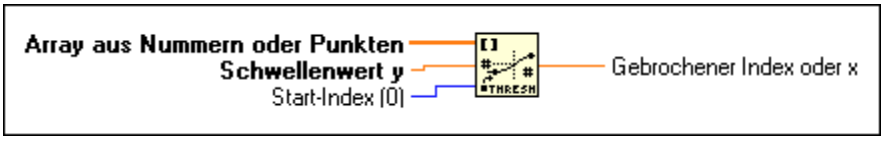
Teilt **Array** am **Index** und gibt die zwei Teile aus.



Schwellwert 1D Array

Vergleicht **Schwellwert y** am **Start-Index** beginnend mit den Werten in **Array aus Zahlen oder Punkten**, bis ein solches Paar aufeinanderfolgender Elemente gefunden wird, daß der **Schwellwert y** größer als der Wert des ersten Elements und kleiner als oder gleich dem Wert des zweiten Elements ist.

Die Funktion berechnet dann den gebrochenen Abstand zwischen dem ersten Wert und **Schwellwert y** und gibt den gebrochenen Index aus, an welchem **Schwellwert y** mittels linearer Interpolation innerhalb von **Array aus Nummern oder Punkten** plaziert würde.

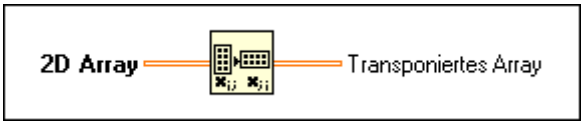


Stellen Sie sich z.B. vor, daß **Array aus Nummern oder Punkten** ein Array aus vier Zahlen [4, 5, 5, 6] ist, der **Start-Index** 0 und **Schwellwert y** 5 beträgt. Entsprechend dem ersten von der Funktion gefundenen Wert von 5, beträgt dann der **gebrochene Index oder x** 1. Stellen Sie sich vor, daß die Array-Elemente 6, 5, 5, 7, 6, 6 sind, der **Start-Index** 0 und **Schwellwert y** 6 oder kleiner ist. Die Ausgabe ist 0. Wenn **Schwellwert y** für denselben Zahlensatz größer als 7 ist, ist die Ausgabe 5. Wenn **Schwellwert y** 14,2 beträgt, der **Start-Index** 5 ist und die Werte in dem Array am Index 5 beginnend 9,1: 10,3 12,9 und 15,5 lauten, fällt **Schwellwert y** zwischen die Elemente 7 und 8, weil sich 14,2 in der Mitte zwischen 12,9 und 15,5 befindet. Der Wert von **Gebrochener Index oder x** lautet 7,5, in der Mitte zwischen 7 und 8.

Wenn es sich bei der Array-Eingabe um ein Punkte-Array handelt, in dem jeder Punkt einen Cluster von x- und y-Koordinaten darstellt, ist die Ausgabe entsprechend der interpolierten Position von **Schwellwert y** der interpolierte x-Wert, anstatt der gebrochene Index des Arrays. Wenn sich die interpolierte Position von **Schwellwert y** in der Mitte der Indexwerte 4 und 5 des Arrays mit x-Werten von -2,5 und 0 befindet, ist die Ausgabe kein Indexwert von 4,5, wie er im numerischen Array lauten würde, sondern stattdessen ein x-Wert von -1,25.

2D Array transponieren

Ordnet die Elemente von **2D Array** derart neu an, daß **2D Array**[*i*,*j*] das **Transponierte Array**[*i*,*j*] wird.

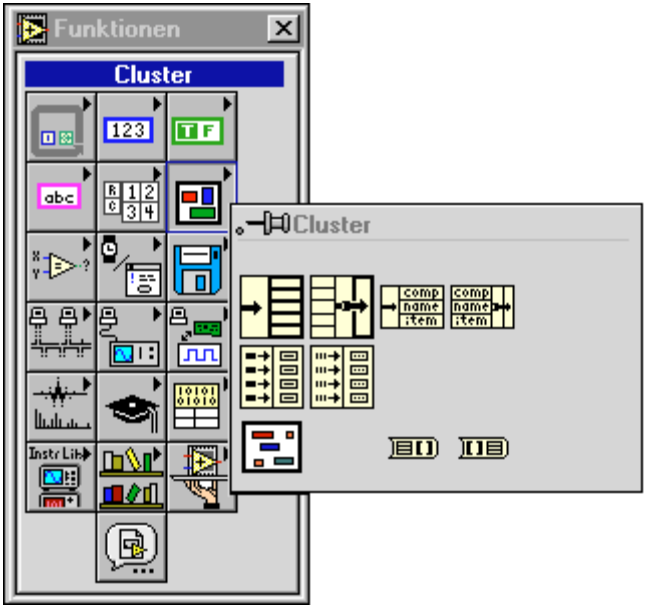


8

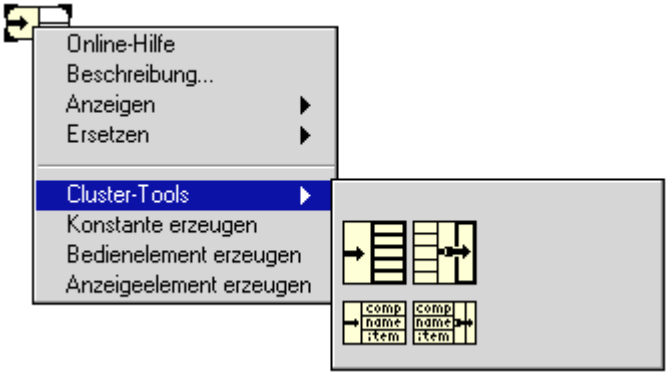
Cluster-Funktionen

Dieses Kapitel beschreibt die Funktionen für Cluster-Operationen.

Die folgende Abbildung zeigt die **Clusterpalette**, auf die Sie über das Menü **Funktionen»Cluster** zugreifen können.



Einige der Cluster-Funktionen stehen auch auf der **Cluster-Tools**-Palette der meisten Popup-Menüs der Anschlüsse und Verbindungen. Die folgende Abbildung zeigt das Popup-Menü.



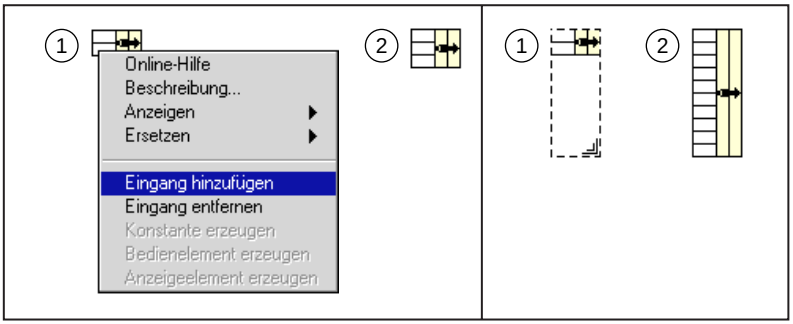
Wenn die Funktionen von dieser Palette ausgewählt werden, werden Sie mit der korrekten Anzahl von zu verbindenden Anschlüssen für das jeweilige Objekt, zu dem das Popup-Menü gehört, angezeigt.

Überblick über die Cluster-Funktionen

Einige der Cluster-Funktionen besitzen eine veränderliche Anzahl von Anschlüssen. Wenn eine neue Funktion dieser Art abgelegt wird, erscheint es auf dem Blockdiagramm mit nur einem oder zwei Anschlüssen. Weitere Anschlüsse können über die Optionen des Popup-Menüs **Eingang hinzufügen** oder **Eingang entfernen** oder durch Verändern der Größe des Knotens mit Hilfe des Positionierwerkzeugs hinzugefügt bzw. entfernt werden. Wenn Sie einen Anschluß über das Popup-Menü hinzufügen möchten, muß der Cursor auf dem Eingabeanschluß plaziert werden, um auf das Popup-Menü zuzugreifen.

Sie können einen Knoten schrumpfen, wenn dadurch die verbundenen Anschlüsse nicht gelöscht werden. Die Option **Eingang hinzufügen** fügt einen Anschluß unmittelbar hinter dem Anschluß ein, dessen Popup-Menü aufgerufen wurde. Die Option **Eingang entfernen** entfernt den Eingang, dessen Popup-Menü aufgerufen wurde, selbst wenn es sich um einen verbundenen Anschluß handelt.

Die folgende Abbildung zeigt die zwei Wege, auf denen weitere Anschlüsse zu der Funktion Elemente bündeln hinzugefügt werden können.



Polymorphismus von Cluster-Funktionen

Die Funktionen **Elemente bündeln** und **Aufschlüsseln** zeigen solange keinen Datentypen für ihre individuellen Eingabe- und Ausgabeanschlüsse an, bis Objekte mit diesen Anschlüssen verbunden wurden. Wenn diese Anschlüsse verbunden sind, sehen sie den Datentypen der korrespondierenden Bedien- oder Anzeigeelemente auf dem Frontpanel ähnlich.

Anordnung der Clusterelemente festlegen

Die Clusterelemente besitzen eine logische Reihenfolge, die nicht mit ihrer Lage innerhalb der Schale zusammenfällt. Das erste Objekt, das in den Cluster eingefügt wird, ist Element 0, das zweite ist 1 usw. Wenn ein Element gelöscht wird, paßt sich die Anordnung automatisch an. Die aktuelle Anordnung kann verändert werden, indem Sie vom Popup-Menü des Clusters die Option **Cluster-Anordnung** wählen.

Anklicken eines Elements mit Hilfe des Cursors Cluster-Anordnung setzt das Element an die innerhalb der Werkzeugpalette angezeigte Stellennummer in der Clusteranordnung. Sie können diese Stellenanordnung ändern, indem Sie in dieses Feld eine neue Zahl eingeben. Wenn die Stellenanordnung mit derjenigen übereinstimmt, die Sie wünschen, klicken Sie auf die Schaltfläche **Eingabe**, um sie festzulegen und den Bearbeitungsmodus der Cluster-Anordnung zu beenden. Klicken Sie auf die Schaltfläche **X**, um zu der alten Anordnung zurückzukehren.

Die Cluster-Anordnung legt die Anordnung fest, in der die Elemente als Anschlüsse der Funktionen Elemente bündeln und Aufschlüsseln auf dem Blockdiagramm angezeigt werden.

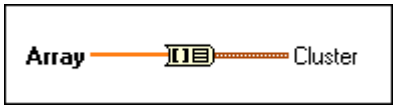
Die Funktionen Nach Namen bündeln und Nach Namen aufschlüsseln geben Ihnen flexibleren Zugriff auf Daten in den Clustern. Durch diese Funktionen können Sie auf spezifische Elemente in den Clustern dem Namen nach zugreifen und nur die Elemente aufrufen, auf die Sie zugreifen möchten. Weil diese Funktionen Komponenten dem Namen nach und nicht nach der Clusterposition referenzieren, können Sie die Datenstruktur eines Clusters, ohne Verbindungen zu unterbrechen, ändern, solange Sie den Namen der referenzierten Komponente auf dem Blockdiagramm nicht ändern oder diese entfernen.

Beschreibungen der Cluster-Funktionen

Die folgenden Cluster-Funktionen sind verfügbar.

Array in Cluster

Konvertiert ein 1D Array in einen Cluster aus Elementen desselben Typs wie den Array-Elementen. Rufen Sie das Popup-Menü auf oder verändern Sie dessen Größe, um die Anzahl der Elemente in dem Cluster einzustellen. Der Standardwert ist 9. Die maximale Clustergröße dieser Funktion beträgt 256.

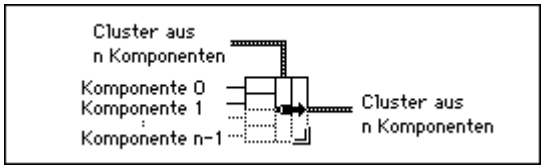


Cluster-Array erstellen

Stellt alle **Komponente**-Eingänge in einem Array aus Clustern dieser **Komponente** von oben nach unten zusammen. Wenn die Eingabe aus vier Single Fließkommazahlen besteht, besteht die Ausgabe aus einem Clusterarray aus vier Elementen mit einer Single Fließkommazahl. Das Element 0 des Arrays besitzt den Wert der obersten Komponente usw.

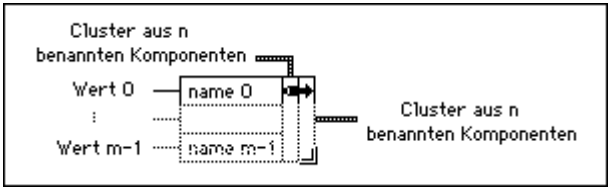
Elemente bündeln

Stellt alle individuellen Eingabekomponenten in einem einzigen Cluster zusammen.



Nach Namen bündeln

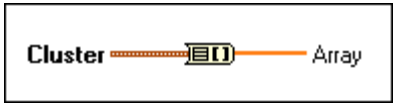
Ersetzt die Komponenten in einem vorhandenen Cluster. Nachdem Sie den Knoten mit dem Cluster verbunden haben, rufen Sie das Popup-Menü der Namens-Anschlüsse auf, um von der Komponentenliste des Clusters wählen zu können.



Der **Cluster**eingang muß immer verbunden werden. Wenn Sie einen Cluster für eine Clusteranzeige erstellen, können Sie eine lokale Variable der Anzeige mit dem **Cluster**eingang verbinden. Wenn Sie einen Cluster für ein Bedienelement eines Clusters von einem SubVI erstellen, können Sie eine Kopie dieses Bedienelements (nach Möglichkeit versteckt) auf dem Frontpanel des VIs plazieren und das Bedienelement mit dem **Cluster**eingang verbinden.

Cluster in Array

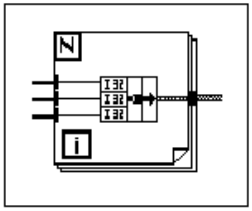
Konvertiert einen Cluster aus identisch eingegebenen Komponenten in ein 1D Array aus Elementen desselben Typs.



Indexieren & Cluster-Array bündeln

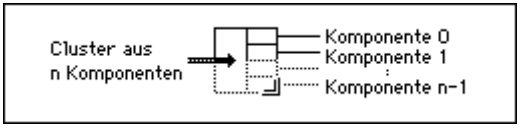
Indexiert einen Satz von Arrays und erstellt ein Clusterarray, in dem das *i*. Element das *i*. Element jedes Eingabearrays enthält.

Diese Funktion ist ein Äquivalent des folgenden Blockdiagramms und ist für die Umwandlung eines Clusters aus Arrays in ein Array aus Clustern nützlich.



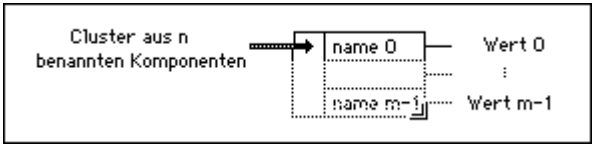
Aufschlüsseln

Zerlegt einen Cluster in dessen individuelle Komponenten.



Nach Namen aufschlüsseln

Gibt die von Ihnen mit Namen angegebenen Clusterelemente aus. Sie können das Element auswählen, auf das Sie zugreifen möchten, indem Sie das Popup-Menü der Namen-Ausgangsanschlüsse aufrufen und einen Namen von der Liste der Elemente in dem Cluster wählen.

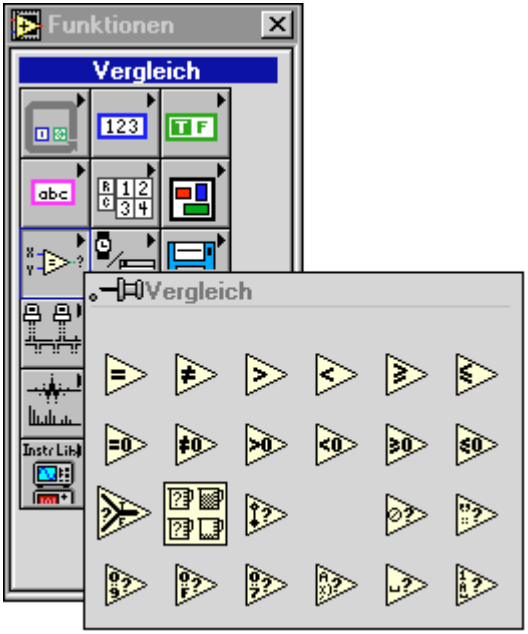


9

Vergleichsfunktionen

Dieses Kapitel beschreibt die Funktionen, die Vergleiche oder bedingte Tests ausführen.

Die folgende Abbildung zeigt die Palette **Vergleichsoperationen**, die über das Menü **Funktionen»Vergleichsoperationen** aufgerufen werden kann.



Überblick über die Vergleichsfunktionen

Dieser Abschnitt stellt die Vergleichsfunktionen vor.

Boolescher Vergleich

Die Vergleichsfunktionen behandeln den Booleschen Wert TRUE als größer als den Booleschen Wert FALSE.

String-Vergleich

Diese Funktionen vergleichen Strings entsprechend der numerischen Äquivalente der ASCII-Zeichen. Demnach ist a (mit einem Dezimalwert von 97) größer als A (mit 65), was wiederum größer als 0 (48) ist, was wiederum größer als das Leerzeichen (32) ist. Diese Funktionen vergleichen am Anfang des String beginnend die einzelnen Zeichen auf der Grundlage Eins-zu-Eins, bis eine Ungleichheit auftritt. An diesem Punkt endet dann der Vergleich. LabVIEW vergleicht z.B. die Strings abcd und abef, bis es c findet, welches einen kleineren Wert als e besitzt. Die Anwesenheit eines Zeichens ist größer als die Abwesenheit eines Zeichens. Der String abcd ist also größer als abc, weil der erste String länger ist.

Die Funktionen, die die Kategorie eines Stringzeichens überprüfen (z.B. die Funktionen **Dezimalstellen?** und **Druckbar?**) bewerten nur das erste Zeichen eines Strings.

Numerischer Vergleich

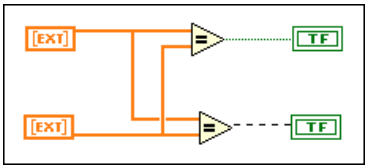
Die Mehrzahl der Vergleichsfunktionen überprüft eine Eingabe oder vergleicht zwei Eingaben und gibt einen Booleschen Wert aus. Die Funktionen konvertieren Zahlen vor dem Vergleichen in dieselbe Repräsentierung. Vergleiche mit einem Wert von Keine Zahl (NaN) geben einen Wert aus, der Ungleichheit anzeigt.

Cluster-Vergleich

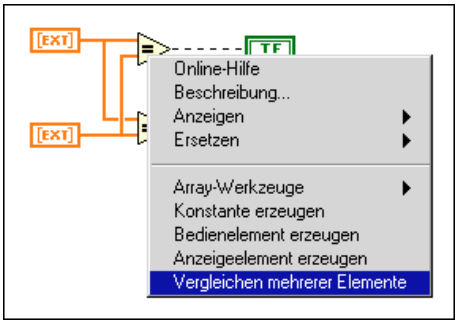
Die Vergleichsfunktionen vergleichen Cluster auf dieselbe Weise, wie sie Strings vergleichen, mit dem 0. Element beginnend jeweils ein einzelnes Element, bis eine Ungleichheit auftritt. Cluster müssen dieselbe Anzahl von Elementen besitzen, von demselben Typ und in derselben Anordnung, wenn sie verglichen werden sollen.

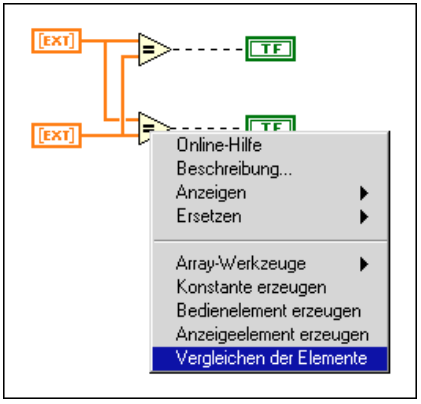
Vergleichsmodi

Einige der Vergleichsfunktionen verfügen über zwei Modi, um Arrays oder Cluster zu vergleichen. Wenn im Modus **Vergleichen mehrerer Elemente** zwei Arrays oder Cluster verglichen werden, gibt die Funktion einen einzelnen Wert aus. Im Modus **Vergleichen der Elemente** vergleicht die Funktion die Elemente individuell und gibt dann ein Array oder einen Cluster aus Booleschen Werten aus. Die folgende Abbildung zeigt die zwei Modi.



Sie können den Vergleichsmodus ändern, indem Sie im Popup-Menü des Knotens, wie in den nachfolgenden Abbildungen gezeigt, **Vergleichen der Elemente** oder **Vergleichen mehrfacher Elemente** auswählen.





Wenn zwei Arrays von ungleicher Länge im Modus **Vergleichen der Elemente** verglichen werden, ignoriert LabVIEW jedes Element in dem größeren Array, dessen Index größer als der Index des letzten Elements in dem kleineren Array ist.

Wenn der Modus **Vergleichen mehrerer Elemente** für den Vergleich von zwei Arrays eingesetzt wird, tritt folgendes auf: (1) LabVIEW sucht in den zwei unterschiedlichen Eingaben nach dem ersten Satz korrespondierender Elemente und verwendet diese, um die Ergebnisse des Vergleichs festzustellen. (2) Wenn alle Elemente identisch sind, außer daß ein Array mehr Elemente besitzt, betrachtet LabVIEW das längere Array als größer als das kürzere. (3) Wenn keine Elemente der zwei Arrays unterschiedlich sind und die Arrays dieselbe Länge besitzen, sind die Arrays gleich. Demzufolge betrachtet LabVIEW das Array [1, 2, 3] als größer als das Array [1,2] und gibt einen einzelnen Booleschen Wert im Modus **Vergleichen der Elemente** aus.

Arrays müssen dieselbe Anzahl von Dimensionen besitzen (z.B. beide sind zweidimensional), und damit ein Vergleich von mehrdimensionalen Arrays sinnvoll ist, muß jede Dimension gleich lang sein.

Bei Clustern, die den Modus **Vergleichen mehrerer Elemente** verwenden, vergleicht LabVIEW anhand der Clusteranordnung. Die zwei Cluster, die LabVIEW vergleicht, müssen dieselbe Anzahl von Elementen besitzen.

Die Vergleichsfunktionen, die nicht über die Modi **Vergleichen mehrerer Elemente** und **Vergleichen der Elemente** verfügen, vergleichen Arrays in derselben Weise wie Strings - mit dem 0. Element beginnend ein Element nach dem anderen, bis eine Ungleichheit auftritt.

Zeichenvergleich

Sie können die Funktionen, die Zeichen vergleichen, einsetzen, um den Typ eines Zeichens zu bestimmen. Die folgenden Funktionen sind Zeichenvergleichsfunktionen.

- Dezimalstellen?
- Anzahl der Hex-Stellen?
- Lexikalische Klasse
- Oktales Zeichen?
- Druckbar?
- Leerfläche?

Wenn es sich bei der Eingabe um einen String handelt, überprüfen die Funktionen das erste Zeichen. Wenn die Eingabe ein leerer String ist, ist das Ergebnis FALSE. Wenn die Eingabe eine Zahl ist, interpretieren die Funktionen sie als einen Code für ein ASCII-Zeichen.

Die Zahlen, die mit jedem ASCII-Zeichen korrespondieren, finden Sie im Anhang C, *GPIB mehrzeilige Schnittstellen-Nachrichten*.

Polymorphismus von Vergleichsfunktionen

Die Funktionen Gleich?, Nicht gleich? und Auswählen arbeiten mit Eingaben jeden Typs, so lange wie die Eingaben demselben Typ angehören.

Die Funktionen Größer oder gleich?, Weniger oder gleich? Weniger?, Größer?, Max & Min und Im Bereich? arbeiten mit Eingaben jeglichen Typs, außer komplexen, Pfad- oder Renum-Eingaben, solange die Eingaben demselben Typ angehören. Sie können Zahlen, Strings, Boolesche Eingaben, Arrays aus String Cluster aus Zahlen, Cluster aus String usw. vergleichen. Sie können jedoch keine Zahl mit einem String oder einen String mit einer Booleschen Eingabe usw. vergleichen.

Die Funktionen, die Werte mit Null vergleichen, akzeptieren numerische Skalare, Cluster und Arrays aus Zahlen. Die Funktionen geben Boolesche Werte in derselben Datenstruktur wie der Eingabe als Ausgabe frei.

Die Funktion Keine Zahl/Pfad/Refnum akzeptiert dieselben Eingabetypen wie die Funktionen, die Werte mit Null vergleichen. Diese Funktionen akzeptieren außerdem Pfade und Refnums. Die Funktion Keine Zahl/Pfad/Refnum gibt Boolesche Werte in derselben Datenstruktur wie die Eingabe aus. Für weitere Informationen über diese Funktionen lesen Sie bitte Kapitel 11, *Dateifunktionen*, und Kapitel 31, *Einführung in LabVIEW Geräte-I/O-VIs*.

Die Funktionen Dezimalstellen?, Anzahl der Hex-Stellen?, Oktales Zeichen?, Druckbar? und Leerstellen? akzeptieren die Eingabe von einem Skalarstring oder einer Zahl, Clusters aus Strings oder von nichtkomplexen Zahlen, Arrays aus String oder nichtkomplexen Zahlen usw. Die Ausgabe besteht aus Booleschen Werten in derselben Datenstruktur wie die Eingabe.

Die Funktion Leere(r) String/Pfad? akzeptiert einen Pfad, einen Skalarstring, Cluster aus String, Arrays aus String usw. Die Ausgabe besteht aus Booleschen Werten in derselben Datenstruktur wie die Eingabe.

Die Funktionen Gleich? Nicht gleich? Keine Zahl/Pfad/Refnum? Leere(r) String/Pfad und Auswählen können mit Pfaden und Refnums eingesetzt werden; andere Vergleichsfunktionen jedoch akzeptieren keine Pfade oder Refnums als Eingabe.

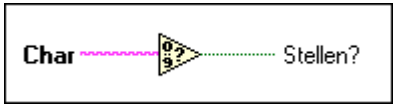
Vergleichsfunktionen, die Arrays und Cluster verwenden, erzeugen normalerweise Boolesche Arrays und Cluster mit derselben Struktur. Sie können das Popup-Menü aufrufen und auf das **Vergleichen mehrerer Elemente** umschalten, wodurch die Funktion dann einen einzelnen Booleschen Wert als Ausgabe freigibt. Die Funktion vergleicht mehrere Elemente, indem der erste Satz von Elementen zur Erzeugung der Ausgabe verwendet wird, außer wenn die ersten Elemente gleich sind; in diesem Fall vergleicht die Funktion den zweiten Satz von Elementen usw.

Beschreibungen der Vergleichsfunktionen

Die folgenden Vergleichsfunktionen sind verfügbar.

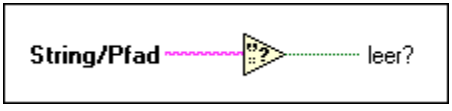
Dezimalstellen?

Gibt TRUE aus, wenn das Zeichen eine Dezimalziffer von 0 bis 9 ist. Ansonsten gibt diese Funktion FALSE aus.



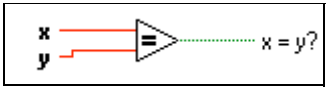
Leere(r) String/Pfad?

Gibt TRUE aus, wenn **String/Pfad** ein leerer String oder Pfad ist. Ansonsten gibt diese Funktion FALSE aus.



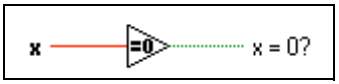
Gleich?

Gibt TRUE aus, wenn **x** gleich **y** ist. Ansonsten gibt diese Funktion FALSE aus.



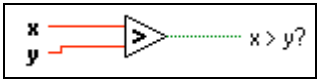
Gleich 0?

Gibt TRUE aus, wenn **x** gleich 0 ist. Ansonsten gibt diese Funktion FALSE aus.



Größer?

Gibt TRUE aus, wenn **x** größer als **y** ist. Ansonsten gibt diese Funktion FALSE aus.



Größer oder gleich?

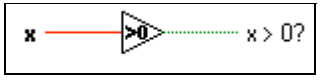
Gibt TRUE aus, wenn **x** größer als oder gleich **y** ist. Ansonsten gibt diese Funktion FALSE aus.

Größer oder gleich 0?

Gibt TRUE aus, wenn **x** größer als oder gleich 0 ist. Ansonsten gibt diese Funktion FALSE aus.

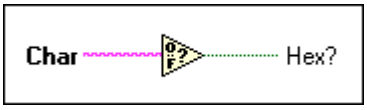
Größer als 0?

Gibt TRUE aus, wenn **x** größer als 0 ist. Ansonsten gibt diese Funktion FALSE aus.



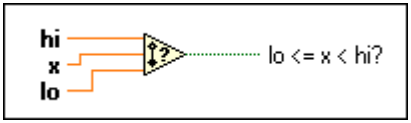
Hex-Stellen?

Gibt TRUE aus, wenn das Zeichen eine Dezimalziffer von 0 bis 9, A bis F oder a bis f ist. Ansonsten gibt diese Funktion FALSE aus.



Im Bereich?

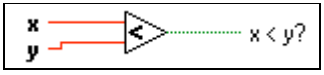
Gibt TRUE aus, wenn **x** größer als oder gleich **lo** und kleiner als **hi** ist. Ansonsten gibt diese Funktion FALSE aus.



☞ **Hinweis** ***Diese Funktion arbeitet immer im Modus* Vergleichen mehrerer Elemente. *Um ein Boolesches Array als eine Ausgabe zu erzeugen, müssen Sie diese Funktion in einer Schleifenstruktur ausführen.***

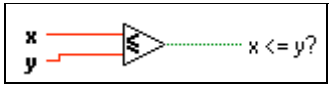
Weniger?

Gibt TRUE aus, wenn **x** kleiner als **y** ist. Ansonsten gibt diese Funktion FALSE aus.



Weniger oder gleich?

Gibt TRUE aus, wenn **x** kleiner als oder gleich **y** ist. Ansonsten gibt diese Funktion FALSE aus.

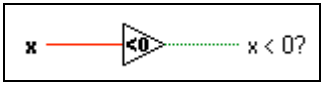


Weniger oder gleich 0?

Gibt TRUE aus, wenn **x** kleiner als oder gleich 0 ist. Ansonsten gibt diese Funktion FALSE aus.

Weniger als 0?

Gibt TRUE aus, wenn **x** kleiner als 0 ist. Ansonsten gibt diese Funktion FALSE aus.



Lexikalische Klasse

Gibt für **Zeichen** als Ausgabe **Klassenzahl** aus.

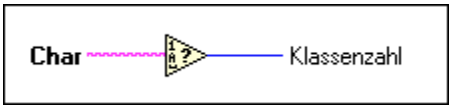
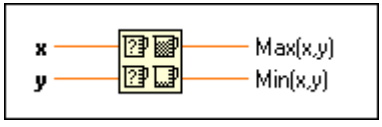


Tabelle 9-1. Beschreibungen von Lexikalische Klassenzahl

Klassenzahl	Lexikalische Klasse
0	Erweiterte Zeichen mit einer Befehls- oder Optionstaste als Präfix (Codes 128 bis 255)
1	Nicht anzeigbare ASCII-Zeichen (Codes 0 bis 31; 9 bis 13 ausgenommen)
2	Leerflächenzeichen: Leertaste/-stelle, Tabulator, Wagenrücklauf, Formularvorschub, Neue Zeile und vertikaler Tabulator (jeweilige Codes 32, 9, 13, 12, 10 und 11)
3	Ziffern 0 bis 9
4	Großbuchstaben A bis Z
5	Kleinbuchstaben a bis z
6	Alle druckbaren, nicht alphanumerischen ASCII-Zeichen

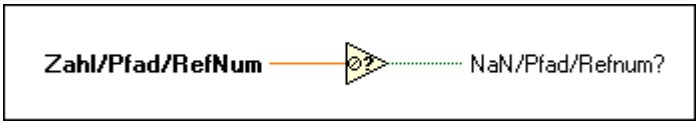
Max & Min

Vergleicht **x** und **y** und gibt den größeren Wert an dem obersten Ausgangs- und den kleineren Wert an dem untersten Ausgangsanschluß aus.



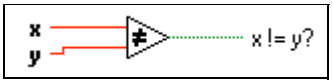
Keine Zahl/Pfad/Refnum?

Gibt TRUE aus, wenn **Zahl/Pfad/Refnum** keine Zahl (NaN), kein Pfad oder keine Refnum ist. Ansonsten gibt diese Funktion FALSE aus. NaN kann das Ergebnis einer Division durch 0 sein, einer Berechnung der Quadratwurzel einer negativen Zahl usw.



Ungleich?

Gibt TRUE aus, wenn **x** ungleich **y** ist. Ansonsten gibt diese Funktion FALSE aus.

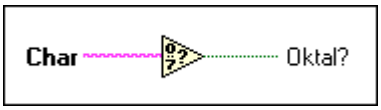


Ungleich 0?

Gibt TRUE aus, wenn **x** ungleich 0 ist. Ansonsten gibt diese Funktion FALSE aus.

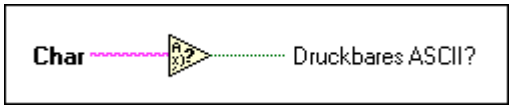
Oktales Zeichen?

Gibt TRUE aus, wenn das Zeichen ein Oktales Zeichen von 0 bis 7 ist. Ansonsten gibt diese Funktion FALSE aus.



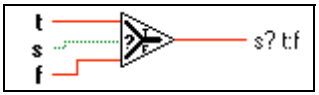
Druckbar?

Gibt TRUE aus, wenn das Zeichen ein druckbares ASCII-Zeichen ist. Ansonsten gibt diese Funktion FALSE aus.



Auswählen

Gibt, von dem Wert **s** abhängig, den mit dem **t**- oder dem **f**-Eingang verbundenen Wert aus. Wenn **s** TRUE ist, gibt diese Funktion den mit **t** verbundenen Wert aus, wenn **s** FALSE ist, gibt diese Funktion den mit **f** verbundenen Wert aus.



Leerfläche?

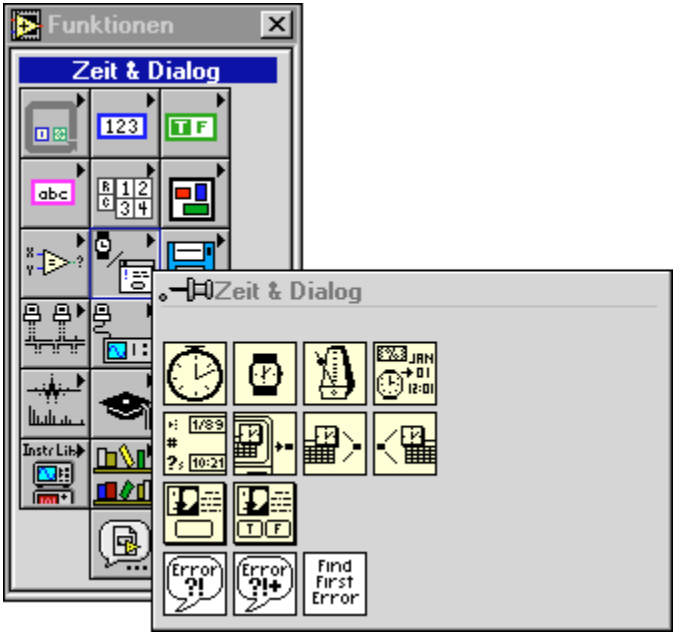
Gibt TRUE aus, wenn das Zeichen ein Leerflächenzeichen ist, wie z.B. Leertaste, Tabulator, Neue Zeile, Wagenrücklauf, Formularvorschub oder vertikaler Tabulator. Ansonsten gibt diese Funktion FALSE aus.

10

Zeit-, Dialog- und Fehlerfunktionen

Dieses Kapitel beschreibt die Timing-Funktionen, mit denen die aktuelle Zeit ausgegeben, vergangene Zeit gemessen oder eine Operation für eine bestimmte Dauer unterbrochen werden kann. Daneben behandelt dieses Kapitel die Fehlerbehandlung.

Die folgende Abbildung zeigt die **Zeit & Dialog**-Palette, die über das Menü **Funktionen»Zeit & Dialog** aufgerufen werden kann.



Überblick über die Zeit-, Dialog- und Fehlerfunktionen

Dieser Abschnitt stellt die Zeit-, Dialog- und Fehlerfunktionen vor.

Timing-Funktionen

Die Funktionen Datum/Zeit zu Sekunden und Sekunden in Datum/Zeit besitzen einen Parameter **Datum/Zeit aufn.**, der sich aus einem Cluster aus vorzeichenbehafteten 32-Bit-Integer-Werten in der folgenden Anordnung zusammensetzt.

Tabelle 10-1. Gültige Werte für Elemente des Datum-/Zeit-Clusters

	Element	Gültige Werte
0	(Sekunde)	0 bis 59
1	(Minute)	0 bis 59
2	(Stunde)	0 bis 23
3	(Tag im Monat)	1 bis 31 als Ausgabe der Funktion; 1 bis 366 als Eingabe
4	(Monat)	1 bis 12
5	(Jahr)	1904 bis 2040
6	(Wochentag)	1 bis 7 (Sonntag bis Sonnabend)
7	(Tag im Jahr)	1 bis 366
8	(DST)	0 bis 1 (0 für Winterzeit, 1 für Sommerzeit)

Die Funktionen Warten (ms) und Wartet bis zum nächsten Vielfachen von ms führen asynchrone Systemaufrufe durch, wobei die Knoten selbst synchron arbeiten. Deshalb schließen sie die Ausführung nicht vor Ablauf einer vorgegebenen Zeitdauer ab. Die Funktionen verwenden asynchrone Aufrufe, damit andere Knoten ablaufen können, während die Timing-Knoten warten.

☞ **Hinweis** ***Zeitwerte außerhalb des Bereichs 082844800 bis 4230328447 Sekunden oder 0.00 Uhr, 1. Jan. 1970, Weltzeit bis 3.14 Uhr, 19. Jan. 2038, Weltzeit konvertieren u.U. nicht auf allen Plattformen zu demselben Datum. Diese Ausnahme besteht insbesondere unter Windows 3.x, das keine Daten vor dem 1. Jan. 1970, Weltzeit unterstützt.***

Überblick über Fehlerbehandlung

Ziehen Sie jedesmal, wenn Sie ein Programm entwerfen, die Möglichkeit in Erwägung, daß etwas schief laufen kann. Für diesen Fall sollten Sie überlegen, wie Ihr Programm das Problem verwaltet. LabVIEW benachrichtigt Sie mit einem Dialogfeld nur dann automatisch, wenn ein paar Ausführungsfehler, in der Mehrzahl in Dateidialogoperationen, auftreten. LabVIEW berichtet nicht alle Fehler. Wenn LabVIEW alle Fehler berichten würde, verlören Sie die Flexibilität festzulegen, was geschehen soll, wenn ein Fehler auftritt, und wie und wann der Benutzer über den Fehler in Ihrem Programm zu informieren ist.

Durchgreifende Fehlerkontrolle, besonders bei I/O-Operationen (Datei, seriell, GPIB, Datenerfassung und Kommunikation) ist in allen Phasen eines Projekts von unschätzbarem Wert. Dieser Abschnitt beschreibt drei I/O-Situationen, in denen Fehler auftreten können.

Der erste Fehlertyp kann auftreten, wenn Ihre Kommunikation verkehrt initialisiert wurde oder falsche Daten an Ihr externes Gerät gesandt wurden. Dieser Problemtyp tritt normalerweise während der Programmentwicklung auf und verschwindet, wenn Sie mit dem Debugging Ihres Programms fertig sind. Trotzdem können Sie mit dem Auffinden einfacher Programmierfehler viel Zeit verbringen, wenn Sie keine Fehlerkontrollen eingebaut haben. Ohne Fehlerkontrolle erfahren Sie lediglich, daß Ihr Programm nicht funktioniert. Sie erfahren weder, warum der Fehler auftritt noch, wo er sich befindet.

Der zweite Fehlertyp kann auftreten, weil ein externes Gerät ausgeschaltet, kaputt oder anderweitig außerstande ist abzuschließen, was es üblicherweise ausführt. Dieser Problemtyp kann jederzeit auftreten; wenn Sie allerdings Fehlerkontrollen in Ihr Programm eingebaut haben, benachrichtigt es Sie umgehend, sobald ein solches Betriebsversagen eintritt.

Der dritte Fehlertyp kann auftreten, wenn Sie LabVIEW oder Ihr Betriebssystem aktualisieren und einen Bug in entweder einem G-Programm oder einem Systemprogramm bemerken. Dieser Fehlertyp bedeutet, daß Sie Fehler überprüfen sollten, die Sie eventuell bisher ignoriert haben, da Sie sich sicher fühlten (z.B. solche von Funktionen, die Dateien schließen oder DAQ-Operationen reinitialisieren). Stellen Sie sicher, daß alle I/O-Operationen auf Fehler überprüft werden.

Es mag u.U. einfacher erscheinen, Fehlerkontrollen zu ignorieren, wenn Fehlerbehandlungscode zum Fehlertesten und zur Fehlermeldung hinzufügt werden muß. Die hier beschriebenen VIs sind so entworfen, daß sie Ihnen die Erstellung von Programmen mit Fehlerkontrollen und -behandlung erleichtern.

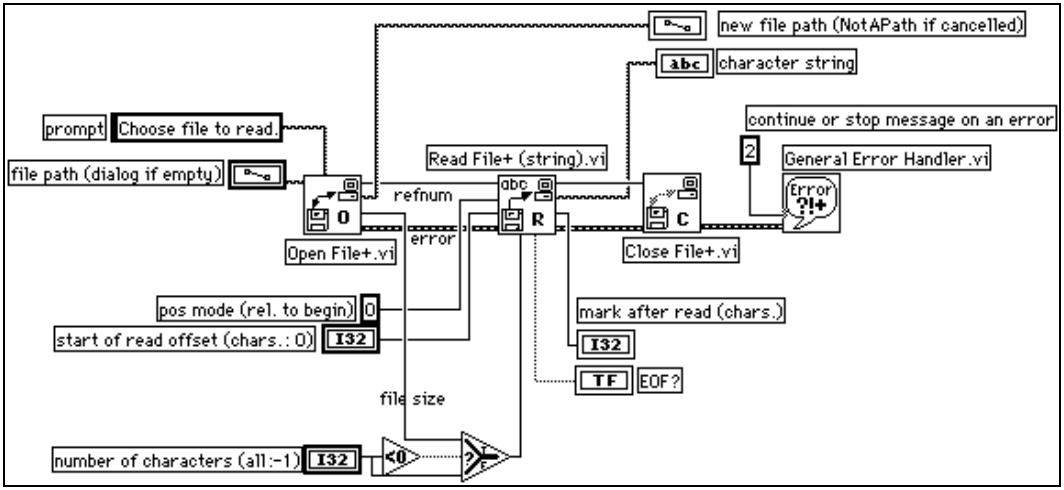
G-Funktionen und Bibliotheks-VIs geben Fehler auf einem von zwei Wegen aus - als numerische Fehlercodes oder als einen Fehlerzustandscluster. Normalerweise geben die Funktionen Fehlercodes frei, während VIs den Fehlercluster gewöhnlich innerhalb eines Fehler-Eingang/Ausgang (Fehler-I/O) genannten Rahmens berücksichtigen.

Fehler-I/O und der Fehlerzustandscluster

Das Konzept vom Fehler-I/O ist logischer Bestandteil der G-Datenflußarchitektur. So wie Dateninformationen können auch Fehlerzustandsinformationen von einem Knoten zum anderen fließen. Jeder Knoten, der Informationen über Fehler benötigt, testet den eingehenden Fehlerzustand und reagiert dementsprechend. Wenn kein Fehler vorhanden ist, läuft der Knoten wie normal ab. Wenn ein Fehler vorhanden ist, entdeckt der Knoten einen Fehler, überspringt die Ausführung und gibt seinen Fehlerzustand an den nächsten Knoten aus, welcher in derselben Weise reagiert. Auf diese Art wird die Kenntnis von dem ersten in einer Serie von Operationen aufgetretenen Fehler an alle Knoten durchgegeben, wobei jeder Knoten auf den Fehler reagiert. Am Ende des Flusses berichtet das Programm dem Benutzer den Fehler.

Fehler-I/O besitzt einen zusätzlichen Vorzug - Sie können mit ihm die Ausführungsfolge unabhängiger Operationen steuern. Obwohl mit der DAQ-Task-ID die Folge der DAQ-Operationen für eine Gruppe gesteuert werden kann, kann mit ihr die Folge für mehrfache Gruppen nicht gesteuert werden. Die DAQ-Task-ID arbeitet nicht mit anderen Typen von I/O-Operationen wie z.B. Fehler-I/O.

Die folgende Abbildung von dem Datei-Utility-VI, `Zeichen aus Datei.vi lesen`, zeigt, wie Fehler-I/O in einem einfachen VI umgesetzt wird.



Die Operation beginnt mit dem `Datei öffnen+.vi`. Wenn es die Datei erfolgreich öffnet, liest das `Lesen (Datei+(String).vi` die Datei und das `Datei schließen +.vi` schließt die Datei. Wenn ein ungültiger Pfad eingegeben wird, entdeckt das `Datei öffnen+.vi` den Fehler und gibt den Fehlerzustand über die anderen zwei VIs an das VI Allgemeiner Error-Handler, der den Fehler berichtet, durch. Beachten Sie bitte, daß auf diesem Blockdiagramm die einzige Anwesenheit von Fehlerbehandlung die Fehlerverbindung und der allgemeine Error-Handler sind. Sie ist weder umständlich noch ablenkend.

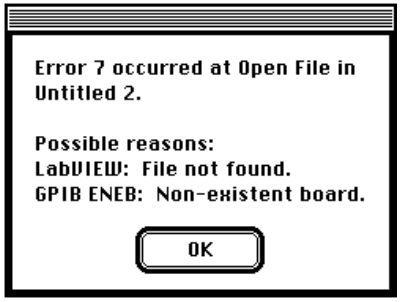
Der Fehlerzustand umfaßt drei Informationen, die in dem Fehlercluster kombiniert sind. Der **Status** ist ein Boolescher Wert - TRUE, wenn ein Fehler vorhanden ist, FALSE, wenn keiner vorhanden ist. Der **Code** besteht aus einem vorzeichenbehafteten 32-Bit-Integer-Wert, der den Fehler kennzeichnet. Ein Fehler-**Code** ungleich Null zusammen mit einem Fehler-**Status** FALSE stellt eine Warnung anstatt einen schwerwiegenden Fehler dar. Ein DAQ-Timeout-Ereignis (Code 10800) wird normalerweise als eine Warnung berichtet. Die **Quelle** besteht aus einem String, der angibt, wo der Fehler aufgetreten ist.

Die Zustandscluster **Fehlereingang** und **Fehlerausgang** für das VI `Datei öffnen+.vi`, an dem der in dem vorangegangenen Beispiel gezeigte Fehler entstanden ist, sind in der folgenden Abbildung gezeigt. Der Cluster

Fehlereingang, dessen Standardwert *Kein Fehler* ist, muß nicht verbunden werden, wenn er der erste in der Kette ist.

Sie finden die Cluster **Fehlereingang** und **Fehlerausgang**, indem Sie auf dem Frontpanel im Menü **Elemente»Array & Cluster** wählen.

Die folgende Abbildung zeigt die Nachricht an, die Sie vom VI Allgemeiner Error-Handler erhalten, wenn Sie einen ungültigen Pfad eingeben.



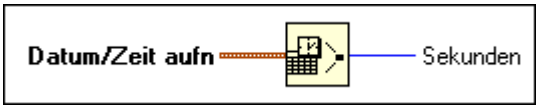
Allgemeiner Error-Handler ist eins von den drei Fehlerbehandlungs-Utility-VIs. Es enthält eine Datenbank von Fehlercodes und -beschreibungen, von der es Nachrichten wie die vorigen erstellt. Das VI Einfacher Error-Handler führt dieselbe grundlegende Operation aus, verfügt jedoch über weniger Optionen. Das dritte VI, Ersten Fehler finden, erstellt den I/O-Cluster von Funktionen oder VIs, die nur skalare Fehlercodes ausgeben.

Beschreibungen der Zeit- und Dialogfunktionen

Die folgenden Zeit- und Dialogfunktionen sind verfügbar.

Datum/Zeit zu Sekunden

Konvertiert einen Cluster aus neun, vorzeichenbehafteten 32-Bit-Integer-Werten — von denen angenommen wird, daß sie die spezifische Ortszeit (Sekunde, Minute, Stunde, Tag, Monat, Jahr, Wochentag, Tag im Jahr und Winter- oder Sommerzeit) für die auf Ihrem Computer konfigurierte Zeitzone angeben — in eine von Zeitzonen unabhängige Zahl aus **Sekunden**, die seit 0.00, Freitag, den 1. Januar 1904, Weltzeit vergangen sind.

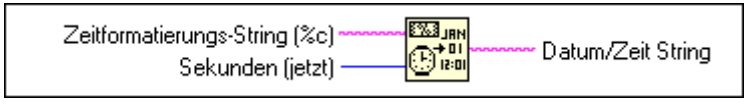


Der Wochentag, Tag im Jahr und die DST-Integer-Werte werden dabei ignoriert. Wenn einer der anderen Integer-Werte außerhalb des in Tabelle 10-1 angegebenen Bereichs liegt, können die Ergebnisse nicht vorausgesagt werden.

Wenn der Tag des Monats als ein Integer-Wert verwendet wird, hat er einen gültigen Bereich von 1 bis 366. Sie können also Julianische Daten angeben, indem Sie den Monat auf Januar stellen und den aktuellen Tag auf den Tag im Jahr. Verwenden Sie z.B. Januar 150 für den 150. Tag im Jahr.

Funktion Datum-/Zeit-String formatieren

Ermöglicht Ihnen, das Datum und die Zeit in einem von Ihnen vorgegebenen Format anzuzeigen.



Der **Datum-/Zeit-String** wird von den **Sekunden (jetzt)** bestimmt, welches die Zahl der Sekunden seit 0.00, 1. Januar 1904, Weltzeit ist, und der **Zeitformatierungs-String** trägt das Format des Ausgabe-Strings.

Wenn **Sekunden** nicht verbunden ist, wird die aktuelle Zeit verwendet. Wenn **Zeitformatierungs- String** nicht verbunden ist, ist der Standard `%c`, was mit der Datum-/Zeitrepräsentierung des aktuellen Ortes korrespondiert.

Die Funktion Datum-/Zeit-String berechnet den **Datum-/Zeit-String**, indem sie **Zeitformatierungs- String** kopiert und jeden der Formatcodes mit den korrespondierenden Werten in der folgenden Tabelle ersetzt.

Tabelle 10-2. Formatcodes für den Zeitformat-String

Formatcode	Wert
%%	ein einzelnes Prozentzeichen
%a	Abkürzung vom Wochentag (z.B. Mi)
%A	ganzer Wochentag (z.B. Mittwoch)
%b	Abkürzung vom Monatsnamen (z.B. Jun)
%B	ganzer Monatsname (z.B. Juni)
%c	die Standardrepräsentierung für das Datum und die Zeit am aktuellen Ort
%d	Tag des Monat (01–31)
%H	Stunde (24-Stundenangabe) (00–23)
%I	Stunde (12-Stundenangabe) (01–12)
%j	Tageszahl im Jahr (001–366)
%m	Monatszahl (01–12)
%M	Minute (00–59)
%p	AM oder PM-Kennzeichnung für Vormittag oder Nachmittag
%S	Sekunden (00–59)
%U	Wochenzahl im Jahr (00–53), bei der die Woche mit Sonntag anfängt
%w	Wochentag als Dezimalzahl (0–6), wobei 0 Sonntag darstellt
%W	Wochenzahl im Jahr (00–53), bei der die Woche mit Montag anfängt
%x	Datumrepräsentierung im aktuellen Ort
%X	Zeitrepräsentierung im aktuellen Ort
%y	Jahr im Jahrhundert (00–99)
%Y	Jahr, einschließlich Jahrhundert (z.B., 1997)
%Z	Name oder Abkürzung der Zeitzone

Zeichen, die innerhalb des **Zeitformat-String**s erscheinen und keinen Teil eines Formatcodes bilden, werden wörtlich in den Ausgabe-String kopiert. Zeitformatcodes (mit % beginnend), die nicht erkannt werden, werden als Zeichen wörtlich ausgegeben.

Zeitformatcodes verfügen immer über führende Nullen, die notwendig sind, um eine konstante Feldlänge zu gewährleisten. Ein optionaler #-Modifizierer vor dem Formatcodebuchstaben entfernt die führenden Nullen von den nachfolgenden Formatcodes:

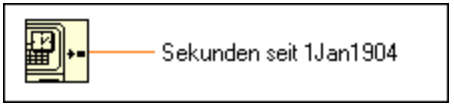
%#d, %#H, %#I, %#j, %#m, %#M, %#S, %#S, %#U, %#w, %#W, %X, %#y, %#Y.

Der # Modifizierer modifiziert das Verhalten von keinen anderen Formatcodes.

☞ **Hinweis** ***Die Formatcodes %c, %x, %X und %Z hängen von der Ortsunterstützung des Betriebssystems ab; die Ausgabe dieser Codes hängt von der Plattform ab. Die Interpretation der Sommerzeitregel kann sich auch mit der Plattform ändern.***

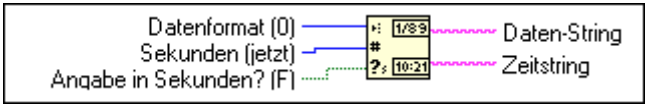
Datum/Zeit in Sekunden holen

Gibt eine von Zeitzonen unabhängige Zahl aus, die die Anzahl der vergangenen Sekunden seit 12.00 Uhr mittags, Freitag, dem 1. Januar 1904, Weltzeit darstellt.



Datum-/Zeit-String holen

Konvertiert eine von Zeitzonen unabhängige Zahl — die als die Anzahl der Sekunden berechnet wird, die seit 12.00 Uhr mittags, Freitag, dem 1. Januar 1904, in Weltzeit dargestellt, vergangen sind — in einen Datum- und Zeit-String für die auf Ihrem Computer konfigurierte Zeitzone.

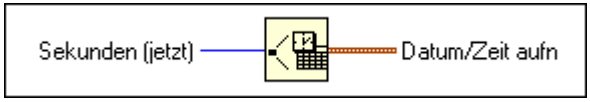


Dialogfeld mit einer Schaltfläche

Zeigt ein Dialogfeld an, das eine Nachricht und eine einzige Schaltfläche enthält. Das Bedienelement **Name der Schaltfläche** ist der Name auf der Schaltfläche des Dialogfeldes.

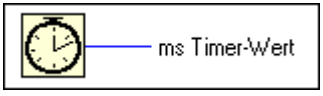
Sekunden in Datum/Zeit

Konvertiert eine von Zeitzonen unabhängige Zahl - die als die Anzahl der **Sekunden** berechnet wird, die seit 0.00 Uhr, Freitag, dem 1. Januar 1904, in Weltzeit dargestellt, vergangen sind - in einen Cluster aus neun, vorzeichenbehafteten 32-Bit-Integer-Werten, die die Ortszeit für die auf Ihrem Computer konfigurierte Zeitzone angeben (Sekunde, Minute, Stunden, Tag des Monats, Monat im Jahr, Wochentag, Tag im Jahr und Winter- oder Sommerzeit). Der Parameter Winterzeit oder Sommerzeit wird gemäß der Sommerzeit-Einstellung in Ihrem Betriebssystem gesetzt und zeigt an, ob der Datum-/Zeit-Cluster aufgrund der Sommerzeit angepaßt wurde.



Tick Count (ms)

Gibt den Wert vom ms Timer aus. Die Basisbezugszeit (Millisekunde Null) ist nicht definiert, weshalb der **ms Timer-Wert** nicht in eine wirkliche Zeit oder ein wirkliches Datum umgewandelt werden kann. Sie müssen vorsichtig sein, wenn Sie diese Funktion zu Vergleichsoperationen einsetzen, da der Wert vom ms Timer nach $2^{32}-1$ auf 0 umspringt.

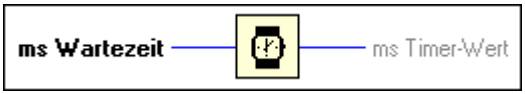


Dialogfeld mit zwei Schaltflächen

Zeigt ein Dialogfeld an, das eine **Nachricht** und zwei Schaltflächen enthält. **Bezeichnung der T-Schaltfläche** und **F-Schaltflächebezeichnung** sind die auf den Schaltflächen des Dialogfeldes angezeigten Namen.

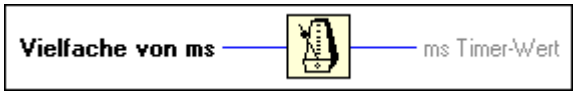
Warten (ms)

Wartet die vorgegebene Anzahl von Millisekunden und gibt dann den Wert vom ms Timer aus.



Wartet bis zum nächsten Vielfachen von ms

Wartet, bis der Wert vom ms Timer ein Vielfaches vom **Vielfache von ms** beträgt. Diese Funktion wird eingesetzt, um Aktivitäten zu synchronisieren. Sie kann in einer Schleife aufgerufen werden, um die Ausführungsrate der Schleife zu steuern. Dabei könnte es jedoch passieren, daß die erste Schleifenperiode kürzer ausfällt.

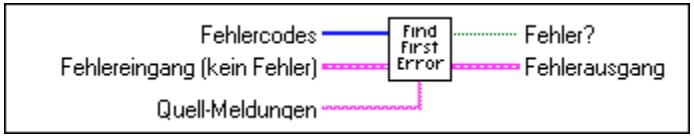


Beschreibungen der Fehlerbehandlungs-VI

Die folgenden Fehlerbehandlungs-VIs sind verfügbar.

Ersten Fehler finden

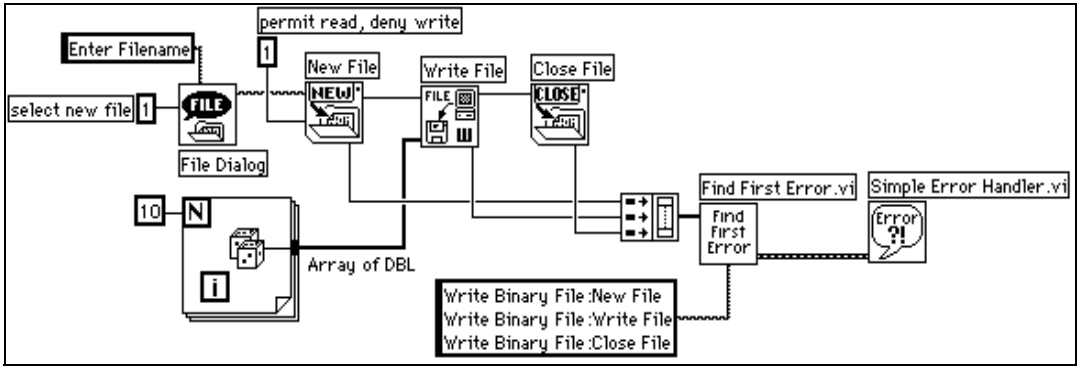
Überprüft den Fehlerstatus von einem oder mehreren Funktionen niedrigerer Ebene oder SubVIs, die einen numerischen Fehlercode ausgeben.



Wenn dieses VI einen Fehler findet, setzt es die Parameter in dem Fehlerausgangscluster. Sie können diesen Cluster mit dem Einfacher oder Allgemeiner Error-Handler verbinden, um Fehler zu identifizieren und für den Benutzer zu beschreiben.

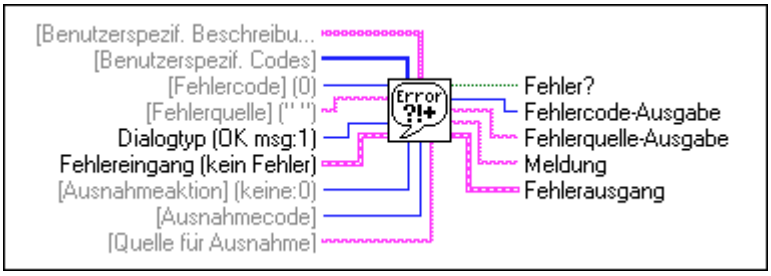
Die folgende Abbildung zeigt, wie die Funktion Ersten Fehler finden in dem Beispiel vom VI Binärdatei schreiben eingesetzt werden kann. Die Funktion Ersten Fehler finden erstellt

aus den einzelnen Fehlernummern einen Fehlercluster, und Einfacher Error-Handler berichtet jeden der Fehler an den Benutzer.



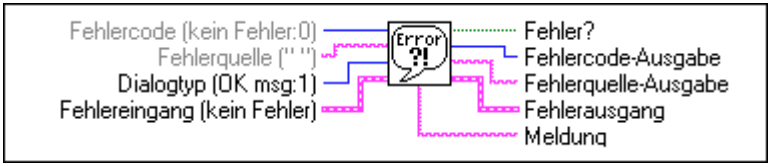
Allgemeiner Error-Handler

Bestimmt, ob ein Fehler aufgetreten ist. Wenn ein Fehler aufgetreten ist, erstellt dieses VI eine Beschreibung des Fehlers und zeigt ein Dialogfeld an, wenn diese Option aktiviert wurde.



Einfacher Error-Handler

Bestimmt, ob ein Fehler aufgetreten ist. Wenn es einen Fehler findet, erstellt dieses VI eine Beschreibung des Fehlers und zeigt ein Dialogfeld an, wenn diese Option aktiviert wurde.



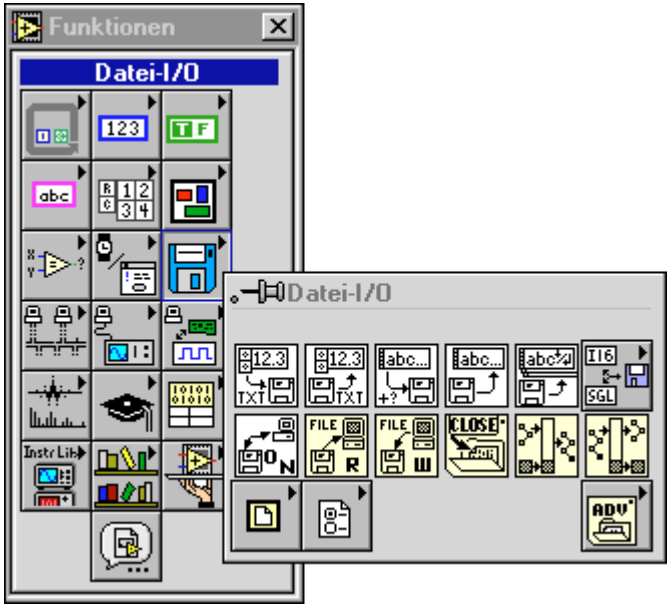
Das VI Einfacher Error-Handler ruft das VI Allgemeiner Error-Handler auf und verfügt über dieselbe grundlegende Funktionalität wie das VI Allgemeiner Error-Handler, nur mit weniger Optionen.

11

Dateifunktionen

Dieses Kapitel beschreibt die Low-Level-VIs und Funktionen, die Dateien, Verzeichnisse und Pfade bearbeiten. Daneben beschreibt diese Kapitel auch Dateikonstanten und High-Level-VIs.

Sie können auf diese Funktionen, Konstanten und -VIs über das Menü **Funktionen»Datei I/O** zugreifen.



Die **Datei I/O**-Palette beinhaltet die folgenden Unterpaletten:

- Fortgeschrittene Dateifunktionen
- Binärdatei-VIs
- Konfigurationsdatei-VIs
- Dateikonstanten

Beispiele für die Dateifunktionen und -VIs finden Sie in `examples\file`.

Überblick über die Datei-I/O-VIs und -Funktionen

Dieser Abschnitt stellt die High-Level- und Low-Level-VIs und die Dateifunktionen vor.

High-Level-Datei-VIs

Sie können High-Level-Datei-VIs zum Schreiben oder Lesen folgender Datentypen einsetzen:

- Strings in Textdateien
- Eindimensionale (1D) oder zweidimensionale (2D) Arrays aus Single Zahlen in Tabellendateien im Textformat
- 1D oder 2D Arrays aus Single Zahlen oder vorzeichenbehafteten Word-Integer-Werten in Byte-Stream-Dateien

Die hier beschriebenen High-Level-VIs rufen die Low-Level-Dateifunktionen zur Ausführung vollständiger und einfach auszuführender Operationen auf. Diese VIs öffnen oder erstellen eine Datei, schreiben oder lesen sie und schließen sie. Wenn ein Fehler auftritt, zeigen diese VIs ein Dialogfeld an, das eine Beschreibung des Problems enthält und Ihnen die Option anbietet, die Ausführung zu unterbrechen oder fortzufahren.

Die High-Level-Datei-VIs befinden sich in der obersten Reihe der Palette und umfassen die folgenden VIs:

- Binärdatei-VIs - auf der Unterpalette
- Zeichen aus Datei lesen
- Aus Spreadsheet-Datei lesen (Tabelle)
- Zeilen aus Datei lesen
- Zeichen in Datei schreiben
- In Spreadsheet-Datei schreiben (Tabelle)

Low-Level-Datei-VIs und Dateifunktionen

Die Low-Level-Datei-VIs und Funktionen führen jeweils eine Dateioperation auf einmal aus. Diese VIs und Funktionen führen zusätzlich zu ihren anderen Funktionen Fehlererkennung aus. Die am häufigsten verwendeten Low-Level-Funktionen und -VIs befinden sich in der zweiten Reihe der Palette. Die verbleibenden Low-Level-Funktionen befinden sich auf der Unterpalette **Fortgeschrittene Dateifunktionen**.

Die hauptsächlichen Low-Level-Dateioperationen umfassen einen Prozeß aus drei Schritten. Zuerst wird eine Datei angelegt oder geöffnet. Danach werden Daten in die Datei geschrieben oder von der Datei gelesen. Letztlich wird die Datei geschlossen. Andere Dateioperationen beinhalten das Anlegen eines Verzeichnisses, das Verschieben, Kopieren oder Löschen von Dateien, das Leeren von Dateien, das Auflisten von Verzeichnisinhalten, das Ändern von Dateieigenschaften und das Verändern von Pfaden.

Beim Anlegen oder Öffnen von einer Datei müssen Sie dessen Speicherplatz angeben. Verschiedene Computer beschreiben den Speicherplatz von Dateien auf unterschiedliche Weise, doch die Mehrzahl der Computersysteme arbeitet mit einem hierarchischen System, um den Speicherplatz von Dateien anzugeben. In einem hierarchischen Dateiensystem legt das Computersystem eine Hierarchie über das Speichermedium. Es werden Dateien innerhalb von Verzeichnissen gespeichert, die Unterverzeichnisse enthalten können.

Wenn Sie eine Datei oder ein Verzeichnis in einem hierarchischen Dateiensystem angeben, müssen Sie den Datei- oder Verzeichnisnamen sowie den Speicherplatz in dem System angeben. Außerdem unterstützen einige Dateiensystem die Verbindung von mehrfachen diskreten Medien, die Datenträger genannt werden. Windows-Systeme z.B. unterstützen mehrere mit einem System verbundene Laufwerke; in der Mehrzahl der Systeme müssen Sie den Namen des Datenträgers mit angeben, um den Speicherplatz einer Datei vollständig anzugeben. Auf anderen Systemen, wie z.B. UNIX, brauchen Sie den Speicherplatz von Dateien auf dem Speichermedium nicht anzugeben, weil das Betriebsystem die physikalische Umsetzung des Dateiensystems von Ihnen fern hält.

Die Methode, wie das Ziel von einer Dateifunktion identifiziert wird, hängt davon ab, ob das Ziel eine offene Datei ist. Wenn das Ziel keine offene Datei ist oder wenn es ein Verzeichnis ist, geben Sie das Ziel mit Hilfe von *Pfad* zu dem Ziel an. Der Pfad beschreibt den das Ziel enthaltenden Datenträger, die Verzeichnisse zwischen der obersten Ebene und dem Ziel und den Namen des Ziels. Wenn das Ziel eine offene Datei ist, verwenden Sie die *File Refnum*, um die zu verändernde Datei zu identifizieren. Bei der File Refnum handelt es sich um eine Kennung, die mit der Datei assoziiert wird, wenn sie geöffnet wird. Wenn die Datei geschlossen wird, dissoziiert der Dateimanager die File Refnum von der Datei. Mit anderen Worten, die Refnum ist hinfällig, sobald eine Datei geschlossen wurde.

Weiterführende Informationen über Pfadangaben in G und Beispiele von Dateifunktionen finden Sie im *LabVIEW Online-Tutorial: Einführung in LabVIEW.*

Byte-Stream- und Datenprotokolldateien

G kann zwei Dateitypen anlegen bzw. auf sie zugreifen - Byte-Stream- und Datenprotokolldateien.

Eine *Byte-Stream*-Datei, wie der Name besagt, ist eine Datei, deren grundlegende Einheit ein Byte ist. Eine Byte-Stream-Datei kann alles mögliche enthalten, angefangen von homogenen Sätzen von einem G-Datentyp bis zu einer beliebigen Sammlung von Datentypen - Zeichen, Zahlen, Boolesche Werte, Arrays, Strings, Cluster usw. Eine ASCII-Textdatei z.B., eine Datei, die diesen Absatz enthält, ist vielleicht die einfachste Byte-Stream-Datei. Eine ähnliche Byte-Stream-Datei ist eine grundlegende Tabellendatei im Textformat, die aus Reihen mit ASCII-Zahlen besteht, wobei die Zahlen mit Tabulatoren und die Reihen mit Wagenrückläufen getrennt sind.

Eine weitere einfache Byte-Stream-Datei ist ein Array aus binären 16-Bit-Integer-Werten oder Single Fließkommazahlen, die von einem Datenerfassungsprogramm (DAQ) erfaßt worden sein können. Eine etwas kompliziertere Byte-Stream-Datei besteht aus binären 16-Bit-Integer-Werten oder Single Fließkommazahlen, denen eine Überschrift aus ASCII-Text vorangeht, die mitteilt, wie und wann die Daten erfaßt worden sind. Die Überschrift könnte dagegen auch ein Cluster aus Erfassungsparametern, wie z.B. Arrays aus Kanälen und Skalierfaktoren, der Scanrate usw., sein.

Eine Excel-Arbeitsblattdatei, im Gegensatz zu einer Excel-Textdatei, ist ebenfalls eine etwas kompliziertere Art von Byte-Stream-Datei, weil sie Text mit eingefügten Excel-spezifischen Formatierdaten enthält, die unsinnig sind, wenn die Datei als Text gelesen würde. Kurzum, Sie können eine Byte-Stream-Datei anlegen, die jeden einzelnen aller G-Datentypen enthält. Byte-Stream-Dateien können über die High-Level-VIs und die Low-Level-VIs und Funktionen angelegt werden.

Eine *Datenprotokoll*-Datei, auf der anderen Seite, besteht aus einer Sequenz identisch strukturierter Aufzeichnungen. Wie in einer Byte-Stream-Datei können die Bestandteile einer Datenprotokollaufzeichnung aus jedem G-Datentypen bestehen. Der Unterschied besteht darin, daß alle Datenprotokollaufzeichnungen demselben Typ entsprechen müssen. Datenprotokolldateien können nur über Low-Level-Dateifunktionen angelegt werden.

Eine Byte-Stream-Datei wird normalerweise durch Anhängen neuer Strings, Zahlen oder Zahlenarrays von beliebiger Länge an die Datei geschrieben. Daten können auch überall innerhalb der Datei überschrieben werden. Eine Datenprotokolldatei wird durch Anhängen von jeweils einer Aufzeichnung geschrieben, und Aufzeichnungen können nicht überschrieben werden.

Eine Byte-Stream-Datei wird durch Angabe des Byte-Offsets oder -Indexes und der Anzahl von Instanzen des angegebenen Byte-Stream-Typen, der gelesen werden soll, gelesen. Eine Datenprotokolldatei wird durch Angabe des Aufzeichnungs-Offsets oder -Indexes und der Anzahl der Aufzeichnungen, die gelesen werden sollen, gelesen.

Byte-Stream-Dateien werden normalerweise für Text-oder Tabellendaten, die von anderen Anwendungen gelesen werden müssen, verwendet. Byte-Stream-Dateien können zur Aufzeichnung fortlaufend erfaßter Daten, die sequentiell oder zufällig in beliebigen Mengen gelesen werden müssen, verwendet werden. Datenprotokolldateien werden normalerweise für die Aufzeichnung mehrfacher Testergebnisse oder Kurvenformen verwendet, die jede für sich gelesen und individuell behandelt werden. Datenprotokolldateien können nur mit Schwierigkeiten von anderen als G-Anwendungen gelesen werden.

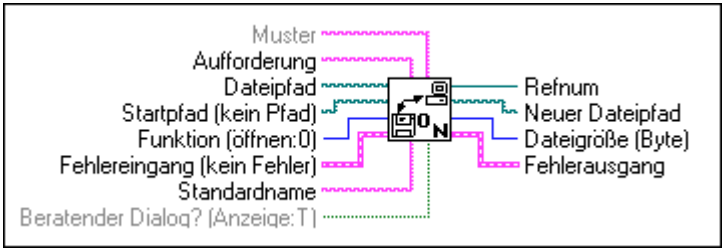
Durchfluß-Parameter

Viele Dateifunktionen enthalten *Durchfluß*-Parameter, die denselben Wert als einen Eingabeparameter ausgeben. Sie können diese Parameter einsetzen, um die Ausführungsreihenfolge der Funktionen zu steuern. Durch Verbinden des Durchfluß-Ausgangs des ersten Knotens, der ausgeführt werden soll, mit dem korrespondierenden Eingang des nächsten Knotens, der ausgeführt werden soll, schaffen Sie eine künstliche Datenabhängigkeit. Ohne diese Durchfluß-Parameter müßten Sie häufig eine Sequenzstruktur einsetzen, um sicherzustellen, daß I/O-Operationen in der richtigen Reihenfolge stattfinden.

Fehler-I/O in Datei-I/O-Funktionen

G verwendet in allen seinen Datei-I/O-Funktionen I/O-Cluster, die sich aus **Fehlereingang** und **Fehlerausgang** zusammensetzen. Mit den I/O-Clustern können Sie mehrere Funktionen zusammenbinden. Wenn ein Fehler in einer Funktion auftritt, gibt diese Funktion den Fehler an die nächste Funktion weiter. Wenn der Fehler an den nachfolgenden Funktionen ankommt, läuft die nachfolgende Funktion nicht ab und gibt

den Fehler an die folgende Funktion weiter usw. Die folgende Abbildung zeigt ein Beispiel von den **Fehlereingangs**- und **Fehlerausgangs**-Clustern.



Obwohl die Fehler-I/O-Cluster angeben, ob ein Fehler aufgetreten ist, ist es ratsam, Error-Handler einzusetzen, um den Fehler an den Benutzer berichten zu lassen. Für weitere Informationen über Fehler-I/O lesen Sie bitte Kapitel 10, *Zeit-, Dialog- und Fehlerfunktionen*, in diesem Handbuch.

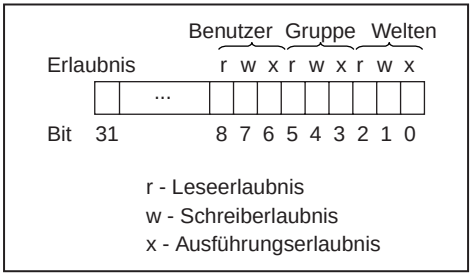
Erlaubnis

Einige der Dateifunktionen haben einen 32-Bit-Integer-Parameter, der **Erlaubnis** oder **Neue Berechtigungen** genannt wird. Diese Funktionen verwenden nur die niedrigstwertigen neun Bits des 32-Bit-Integer-Wertes, um die Datei- und Verzeichniszugriffsberechtigungen zu bestimmen.

(Windows) Die Berechtigungen für Verzeichnisse werden ignoriert. Bei Dateien wird nur Bit 7 benutzt (das UNIX-Benutzer-Schreiberlaubnisbit). Wenn dieses Bit frei ist, besitzt die Datei nur Lesestatus/ist schreibgeschützt. Ansonsten kann zu der Datei geschrieben werden.

(Macintosh) Alle 9 Erlaubnisbits werden für die Verzeichnisse benutzt. Die Bits, die in UNIX-Systemen jeweils die Lese-, Schreib- und Ausführungserlaubnis kontrollieren, werden verwendet, um die Zugriffsrechte von Dateien lesen, Änderungen vornehmen und Ordner einsehen auf Macintoshcomputern zu kontrollieren. Bei Dateien wird nur Bit 7 (das UNIX-Benutzer-Schreiberlaubnisbit) verwendet. Wenn dieses Bit frei ist, ist die Datei gesperrt. Ansonsten ist die Datei nicht gesperrt.

(UNIX) Die neun Erlaubnisbits korrespondieren genau mit den neun UNIX-Erlaubnisbits, die das Lesen, Schreiben und Ausführen für die Benutzer, Gruppen und Welten regeln. Die folgende Abbildung zeigt die Erlaubnisbits von einem UNIX-System.

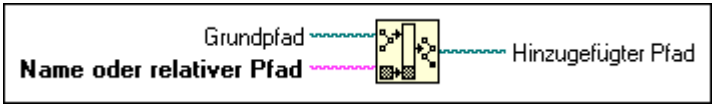


Beschreibungen der Datei I/O- Funktionen und -VIs

Die folgenden Funktionen und -VIs sind von der **Datei I/O**-Palette verfügbar.

Pfad erstellen

Erstellt einen neuen Pfad, indem ein **Name der relativer Pfad** an einen vorhandenen Pfad angehangen wird.



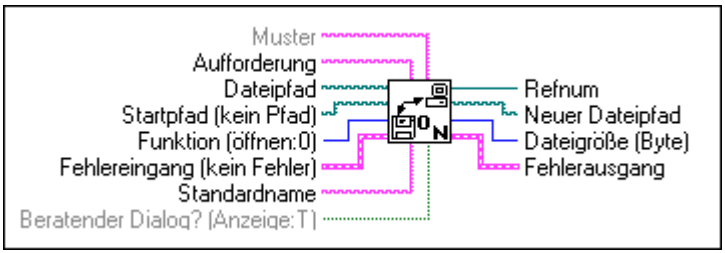
Datei schließen

Schreibt alle durch die **Refnum** gekennzeichneten Puffer der Datei zum Laufwerk, aktualisiert den Dateieintrag im Verzeichnis, schließt die Datei und annulliert die **Refnum** für nachfolgende Dateioperationen.

☞ **Hinweis** ***Das VI Datei schließen handhabt Fehler-I/O anders als andere Dateifunktionen: es läuft ab, wenn sein* Fehlereingang *anzeigt, daß in einer vorangegangenen Funktion ein Fehler aufgetreten ist.***

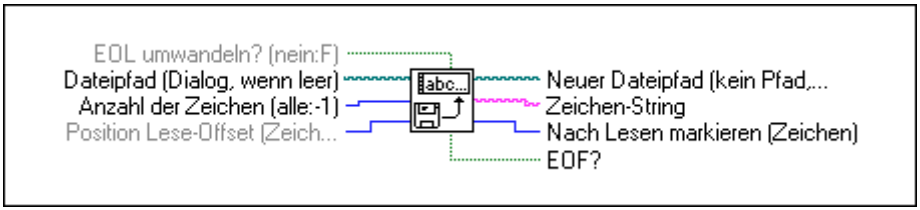
Öffnen/Erstellen/Ersetzen einer Datei

Öffnet eine vorhandene Datei, erstellt eine neue Datei oder ersetzt eine vorhandene Datei, programmgesteuert oder interaktiv über ein Dialogfeld. Sie können wahlweise eine **Aufforderung**, einen **Standardnamen**, einen **Startpfad** oder **Muster** festlegen. Verwenden Sie dieses VI mit den Funktionen Datei schreiben oder Datei lesen.



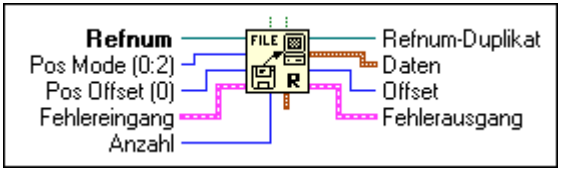
Zeichen aus Datei lesen

Liest mit dem vorgegebenen Offset-Zeichen beginnend aus einer Byte-Stream-Datei eine vorgegebene Anzahl von Zeichen. Das VI öffnet die Datei vor dem Lesen und schließt sie danach.



Datei lesen

Liest aus der durch die **Refnum** vorgegebenen Datei Daten und gibt sie in **Daten** aus. Das Lesen beginnt an der durch **Pos Mode** und **Pos Offset** vorgegebenen Stelle und hängt von dem Format der vorgegebenen Datei ab.



Byte Stream Dateien lesen

Wenn **Refnum** eine Byte-Stream-Datei-Referenznummer ist, liest die Funktion Datei lesen Daten von der durch **Refnum** vorgegebenen Byte-Stream-Datei. Sie können entweder **Zeilenmodus** oder **Byte-Stream-Typ** verbinden, wenn Byte-Stream-Dateien gelesen werden, jedoch nicht beide. Wenn Sie Byte-Stream-Typ nicht verbinden, geht die Funktion Datei lesen davon aus, daß es sich bei den Daten, die an dem festgelegten Byte-Offset anfangen, um einen String aus Zeichen handelt. Wenn Sie **Byte-Stream-Typ** verbinden, interpretiert die Funktion die an dem festgelegten Byte-Offset beginnenden **Daten** als **Anzahl** von Instanzen dieses Typs. Nach den Leseoperationen setzt die Funktion den Dateimarker an das Byte, das dem zuletzt gelesenen Byte folgt. Wenn die Funktion vor dem Lesen aller angeforderten Dateien auf Ende der Datei trifft, gibt sie so viele ganze Instanzen von dem angegebenen **Byte-Stream-Typ**en aus, wie sie vorfindet.

Zeichen lesen

Um Zeichen von einer Byte-Stream-Datei (normalerweise einer Textdatei) lesen zu lassen, verbinden Sie den **Byte-Stream-Typ** nicht. Die folgenden Absätze beschreiben die Art und Weise, mit der die Parameter **Zeilenmodus**, **Anzahl**, **EOL umwandeln** und **Daten** arbeiten, wenn sie von einer Byte-Stream-Datei lesen.

Zeilenmodus in Verbindung mit **Anzahl** bestimmt, wann das Lesen stoppt.

Wenn **Zeilenmodus** TRUE ist und **Anzahl** nicht verbunden oder **Anzahl** gleich 0 ist, liest die Funktion Datei lesen solange, bis sie einen Marker Ende der Zeile findet - einen Wagenrücklauf, einen Zeilenvorschub oder einen von einem Zeilenvorschub gefolgten Wagenrücklauf - oder auf Ende der Datei trifft. Wenn **Zeilenmodus** TRUE und **Anzahl** größer als 0 ist, liest die Funktion Datei lesen, bis sie auf einen Marker Ende der Zeile oder Ende der Datei trifft oder **Anzahl**zeichen liest.

Wenn **Zeilenmodus** FALSE ist, liest die Funktion Datei lesen **Anzahl**zeichen. Wenn in diesem Fall **Anzahl** nicht verbunden ist, gibt sie den Standardwert 0 und **Zeilenmodus** den Standardwert FALSE aus.

EOL umwandeln (F) legt fest, ob die Funktion die Marker Ende der Zeile, die sie liest, in G-Marker Ende der Zeile umwandelt. Unter Windows ist der systemspezifische Marker Ende der Zeile ein von einem Zeilenvorschub gefolgter Wagenrücklauf, unter Macintosh ist er ein Wagenrücklauf und unter UNIX ein Zeilenvorschub. Der G-Marker Ende der Zeile ist ein Zeilenvorschub.

Wenn **EOL umwandeln** TRUE ist, wandelt die Funktion alle Marker Ende der Zeile, auf die sie trifft, in Zeilenvorschübe um. Wenn **EOL umwandeln** FALSE ist, konvertiert die Funktion, die Marker Ende der Zeile, die sie liest, nicht. Der Standardwert von **EOL umwandeln** ist FALSE.

Daten ist der Zeichenstring, der von der Datei gelesen wurde.

Binäre Daten lesen

Um binäre Daten von einer Byte-Stream-Datei zu lesen, muß der Datentyp mit **Byte-Stream-Typ** verbunden sein. In diesem Fall arbeiten die Funktionen **Anzahl** und **Daten** in der in den folgenden Absätzen beschriebenen Art und Weise, und **Zeilenmodus** und **EOL umwandeln** müssen nicht verbunden sein.

Byte-Stream-Typ kann jeder Datentyp sein. Die Funktion Datei lesen interpretiert die mit dem festgelegten Byte-Offset beginnenden Daten als **Anzahl** der Instanzen dieses Typs. Wenn der Typ von variabler Länge ist, d.h., ein Array, ein String oder ein Cluster, geht die Funktion davon aus, daß jede Instanz dieses Typs, die Länge oder Dimensionen von dieser Instanz enthält. Wenn sie diese nicht enthalten, interpretiert diese Funktion die Daten verkehrt. Stellt die Funktion Datei lesen fest, daß die Daten nicht mit dem Typ übereinstimmen, setzt sie den Wert von **Daten** auf dessen Standardwert für diesen Typ und gibt einen Fehler aus.

Anzahl ist die Anzahl der Instanzen vom zu lesenden **Byte-Stream-Typ**. Wenn **Anzahl** nicht verbunden ist, gibt die Funktion eine einzelne Instanz von dem **Byte-Stream-Typ** aus.

Wenn **Anzahl** verbunden wird, kann es eine Skalarzahl sein, wobei die Funktion dann ein 1D Array aus Instanzen des **Byte-Stream-Typ**s ausgibt. Oder es ist ein Cluster aus n Skalarzahlen, wobei die Funktion dann ein n-dimensionales Array aus Instanzen von **Byte-Stream-Typ** ausgibt.

Wenn die verbundene **Anzahl** eine Skalarzahl und **Byte-Stream-Typ** etwas anderes als ein Array ist, gibt diese Funktion diese Anzahl der Instanzen in einem 1D Array aus. Wenn der Typ z.B. eine Single Fließkommazahl und **Anzahl** 3 ist, gibt die Funktion ein Array aus drei Single Fließkommazahlen aus. Wenn jedoch der Typ ein Array ist, gibt die Funktion die Instanzen in einem Clusterarray aus, weil G nicht über Arrays aus Arrays verfügt. Wenn also der Typ ein Array aus Single Fließkommazahlen und **Anzahl** 3 ist, gibt die Funktion ein Clusterarray aus drei Single Fließkommazahlenarrays aus.

Wenn die verbundene **Anzahl** ein Cluster aus n Zahlen ist, gibt die Funktion ein n-dimensionales Array aus Instanzen dieses Typs aus. Die Größe jeder Dimension ist der Wert der korrespondierenden Zahl entsprechend ihrer Anordnung im Cluster. Die Anzahl von in dieser Weise ausgegebenen Instanzen ist das Produkt von den n Zahlen. Sie können somit 20 Single Fließkommazahlen als ein 2D Array aus zwei Spalten und 10 Reihen ausgeben, indem Sie einen Cluster aus zwei Elementen mit Element 0 = 2 und Element 1 = 10 mit **Anzahl** verbinden.

Daten enthält die von der Datei gelesenen Daten. Lesen Sie die vorangegangene Beschreibung zu Anzahl für eine Erklärung der Datenstrukturen, die verfügbar sind.

Datenprotokolldateien lesen

Wenn es sich bei **Refnum** um eine Referenznummer einer Datenprotokolldatei handelt, liest die Funktion Datei lesen Aufzeichnungen von der durch **Refnum** angegebenen Datei. Wenn die Daten in dieser Datei nicht mit dem Datentyp, der mit dieser Datenprotokolldatei assoziiert ist, übereinstimmen, gibt diese Funktion einen Fehler aus.

Die Anzahl der gelesenen Aufzeichnungen kann unter der durch **Anzahl** vorgegebenen liegen, wenn diese Funktion auf den Marker Ende der Datei trifft. Die Funktion setzt den Dateimarker an die Aufzeichnung, die der zuletzt gelesenen Aufzeichnung folgt. (Sie sollten nie eine partielle Aufzeichnung vorfinden; sollte es dennoch geschehen, ist die Datei korrumpiert.)

Verbinden Sie nicht **EOL umwandeln**, **Zeilenmodus** und **Byte-Stream-Typ**. Sie betreffen die Datenprotokolldateien nicht. Die Parameter **Anzahl** und **Daten** arbeiten in der folgenden Art und Weise.

Anzahl ist die Anzahl der zu lesenden Aufzeichnungen und kann verbunden sein oder auch nicht. Wenn **Anzahl** nicht verbunden wird, gibt die Funktion eine einzelne Aufzeichnung von dem Datenprotokolltyp aus, der festgelegt wird, wenn die Datei angelegt oder geöffnet wird. Wenn der Typ z.B. ein 16-Bit-Integer-Wert ist, gibt die Funktion einen 16-Bit-Integer-Wert aus. Wenn der Typ ein Array aus 16-Bit-Integer-Werten ist, gibt die Funktion ein Array aus 16-Bit-Integer-Werten aus. (Normalerweise setzen sich Ihre Aufzeichnungen aus Clustern aus verschiedenen Elementen zusammen, doch die Regeln für die einfachen, in diesen Beispielen verwendeten Typen gelten ebenso für diese.)

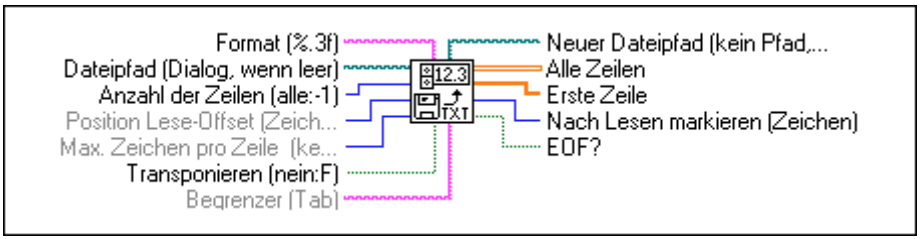
Wenn **Anzahl** verbunden wird, kann es eine Skalarzahl sein, wobei die Funktion dann ein 1D Array aus Aufzeichnungen ausgibt. Oder es kann ein Cluster aus n Skalarzahlen sein, wobei die Funktion dann ein n-dimensionales Array aus Aufzeichnungen ausgibt.

Wenn die verbundene **Anzahl** eine Skalarzahl und der Typ des Datenprotokolls etwas anderes als ein Array ist, gibt die Funktion diese Anzahl der Aufzeichnungen in einem 1D Array aus. Wenn z.B. der Typ eine Single Fließkommazahl und **Anzahl** 3 ist, enthält das Array drei Single Fließkommazahlen. Wenn jedoch der Typ ein Array ist, gibt die Funktion die Aufzeichnungen in einem Clusterarray aus, weil G nicht über Arrays aus Arrays verfügt. Wenn der Typ des Datenprotokolls ein Array aus Single Fließkommazahlen und **Anzahl** 3 ist, gibt die Funktion demnach ein Clusterarray aus drei Single Fließkommazahlenarrays aus.

Wenn die verbundene **Anzahl** ein Cluster aus n Zahlen ist, gibt die Funktion ein n-dimensionales Array aus Aufzeichnungen aus. Die Größe jeder Dimension ist der Wert der korrespondierenden Zahl entsprechend ihrer Anordnung im Cluster. Die Anzahl der Aufzeichnungen, die auf diese Weise ausgegeben werden, ist das Produkt der n Zahlen. Sie können somit 20 Aufzeichnungen als ein 2D Array aus 2 Spalten und 10 Reihen ausgeben, indem Sie einen Zwei-Elemente-Cluster mit Element 0 = 2 und Element 1 = 10 mit **Anzahl** verbinden.

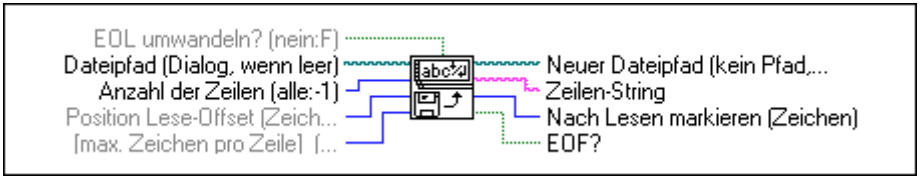
Aus Spreadsheet-Datei lesen (Tabelle)

Liest von einer numerischen Textdatei an einem angegebenen Zeichen-Offset beginnend eine vorgegebene Anzahl von Zeilen oder Reihen und konvertiert die Daten in ein 2D Single Array aus Zahlen. Das Array kann wahlweise transponiert werden. Das VI öffnet die Datei vor dem Lesen und schließt sie danach. Sie können dieses VI zum Lesen einer im Textformat gespeicherten Tabellendatei einsetzen. Dieses VI ruft die Funktion Tabellenstring in Array auf, um die Daten zu konvertieren.



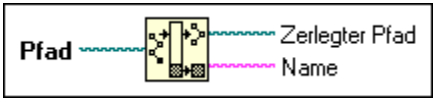
Zeilen aus Datei lesen

Liest aus einer Byte-Stream-Datei mit einem vorgegebenen Zeichen-Offset beginnend eine vorgegebene Anzahl von Zeilen. Das VI öffnet die Datei vor dem Lesen und schließt sie hinterher.



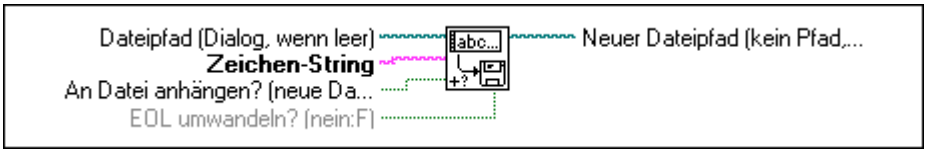
Pfad zerlegen

Gibt **Name** von der letzten Pfadkomponente aus sowie **Zerlegter Pfad**, der zu dieser Komponente führt.



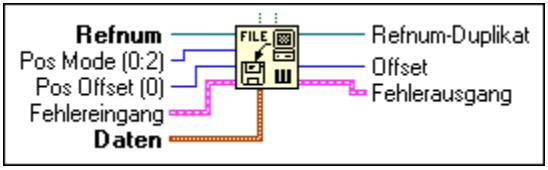
Zeichen in Datei schreiben

Schreibt einen Zeichenstring in eine neue Byte-Stream-Datei, oder hängt den String an eine vorhandene Datei an. Das VI öffnet oder erstellt die Datei vor dem Schreiben und schließt sie hinterher.



Datei schreiben

Schreibt Daten in die durch **Refnum** angegebene Datei. Das Schreiben beginnt bei einer Byte-Stream-Datei an der durch **Pos Mode** oder **Pos Offset** vorgegebenen Stelle und bei einer Datenprotokolldatei am Ende der Datei. **Datum**, **Überschrift** und das Format der vorgegebenen Datei bestimmen die Menge der geschriebenen Daten.



Byte-Stream Dateien schreiben

Wenn **Refnum** die Referenznummer einer Byte-Stream-Datei ist, schreibt die Funktion Datei schreiben an die Stelle in der durch **Refnum** angegebenen Byte-Stream-Datei, die durch **Pos Modus** oder **Pos Offset** vorgegeben ist. Wenn der höchste Datentyp von variabler Länge ist (d.h., ein String oder ein Array), kann die Funktion Datei schreiben eine **Überschrift** in die Datei schreiben, mit der die Größe der Daten angegeben wird. Die Funktion Datei schreiben setzt den Dateimarker an das Byte, das dem zuletzt geschriebenen Byte folgt. Die Funktion **EOL konvertieren** stellt fest, ob es die Marker Ende der Zeile, die sie schreibt, in systemspezifische Marker Ende der Zeile umwandelt. Die Funktion **EOL konvertieren** kann nur verbunden werden, wenn **Daten** ein String ist. Unter Windows ist der systemspezifische Marker Ende der Zeile ein von einem Zeilenvorschub gefolgter Wagenrücklauf, für UNIX ist er ein Zeilenvorschub und für Macintosh ist er ein Wagenrücklauf. Wenn **Überschrift** TRUE ist, wird die Funktion **EOL konvertieren** von der Funktion Datei schreiben ignoriert.

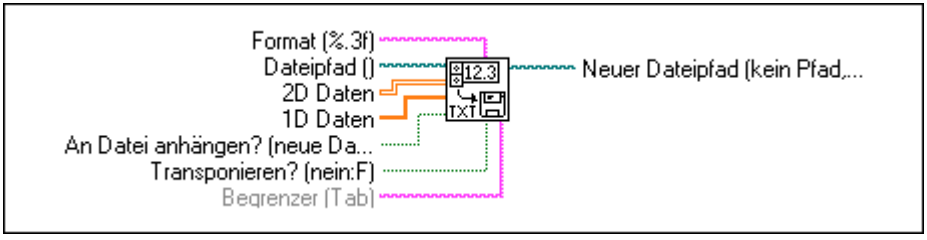
Datenprotokolldateien schreiben

Wenn **Refnum** die Referenznummer einer Datenprotokolldatei ist, schreibt die Funktion Datei schreiben die Daten als Aufzeichnungen in die durch **Refnum** angegebene Datenprotokolldatei. Das Schreiben setzt immer am Ende der Datenprotokolldatei ein (Datenprotokolldateien sind nur zum Anhängen). Die Funktion Datei schreiben setzt den Dateimarker an die Aufzeichnung, die der zuletzt geschriebenen Aufzeichnung folgt. Die Parameter **EOL konvertieren**, **Überschrift**, **Pos Modus** und **Pos Offset** gelten nicht für Datenprotokolldateien und können nicht verbunden werden. Der Parameter **Daten** arbeitet in Datenprotokolldateien in der folgenden Art und Weise.

Daten muß entweder ein Datentyp sein, der mit dem beim Öffnen oder Erstellen der Datei festgelegten Datentyp übereinstimmt, oder ein Array aus diesen Datentypen. Im ersten Fall schreibt diese Funktion **Daten** als eine einzelne Aufzeichnung in die Datenprotokolldatei. Die Repräsentierung numerischer Daten wird gegebenenfalls in die Repräsentierung des Datentyps gezwungen. Im letzteren Fall schreibt diese Funktion jedes Element von **Daten** als eine getrennte Aufzeichnung in zeilenweiser Anordnung in die Datenprotokolldatei.

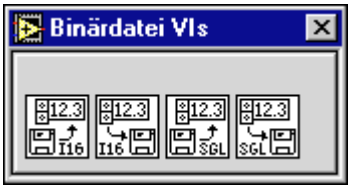
In Spreadsheetdatei schreiben (Tabelle)

Konvertiert ein 2D oder 1D Array aus Single Fließkommazahlen (Sgl) in einen Textstring und schreibt den String in eine neue Byte-Stream-Datei oder hängt den String an eine vorhandene Datei an. Die Daten können wahlweise transponiert werden. Diese VI öffnet oder erstellt die Datei vor dem Schreiben und schließt sie hinterher. Sie können dieses VI einsetzen, um eine von den meisten Tabellenkalkulationsprogrammen lesbare Textdatei zu erstellen. Dieses VI ruft die Funktion Array in Tabellen-String auf, um die Daten zu konvertieren.



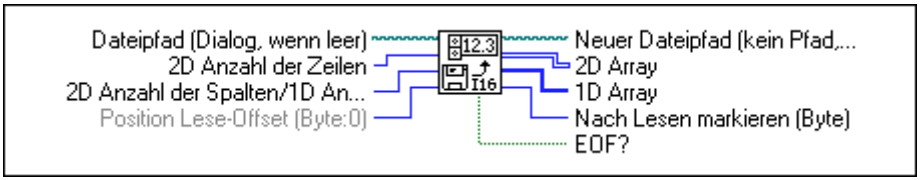
Beschreibungen von Binärdatei-VIs

Die folgenden VIs sind von der Unterpalette **Binärdatei VIs** verfügbar.



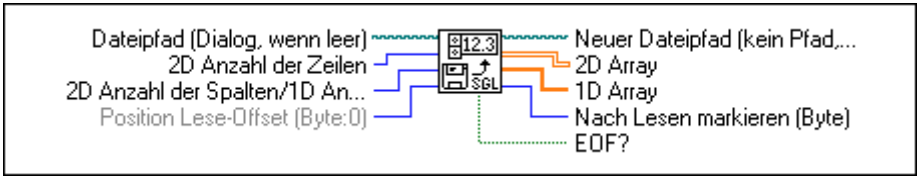
Aus I16-Datei lesen

Liest aus einer Byte-Stream-Datei mit vorzeichenbehafteten Word-Integer-Werten (I16) ein 2D oder 1D Array aus Daten. Das VI öffnet die Datei vor dem Lesen und schließt sie hinterher. Sie können dieses VI zum Lesen von unskalierten oder binären Daten einsetzen, die von Datenerfassungs-VIs erfaßt und von der Funktion In I16-Datei schreiben in eine Datei geschrieben wurden.



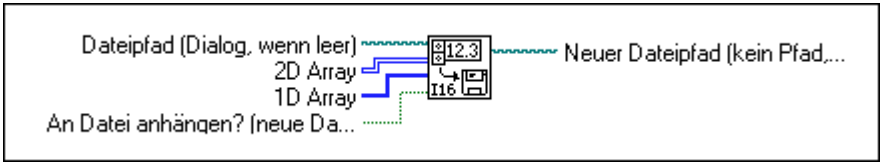
Aus Sgl-Datei lesen

Liest aus einer Byte-Stream-Datei aus Fließkommazahlen (Sgl) ein 2D oder 1D Array aus Daten. Das VI öffnet die Datei vor dem Lesen und schließt sie hinterher. Sie können das VI zum Lesen skalierter Daten einsetzen, die von Datenerfassungs-VIs erfaßt und von der Funktion In Sgl-Datei schreiben in eine Datei geschrieben wurden.



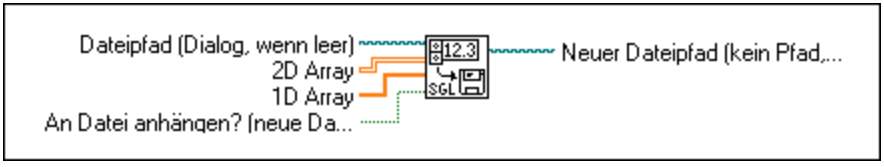
In I16-Datei schreiben

Schreibt ein 2D oder 1D Array aus vorzeichenbehafteten Word-Integer-Werten (I16) in eine neue Byte-Stream-Datei oder hängt die Daten an eine vorhandene Datei an. Das VI öffnet oder erstellt die Datei vor dem Schreiben und schließt sie hinterher. Sie können dieses VI zum Schreiben von unskalierten oder binären Daten von Datenerfassungs-VIs einsetzen.



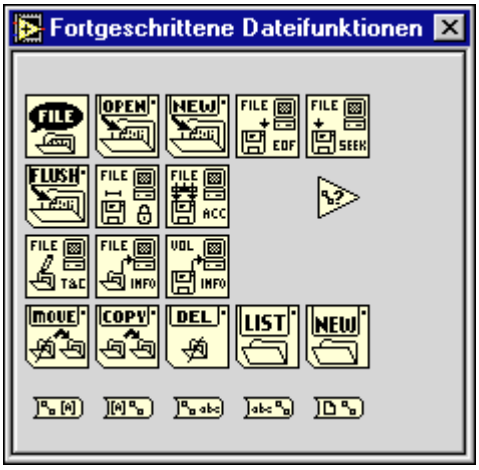
In Sgl-Datei schreiben

Schreibt ein 2D oder 1D Array aus Single Zahlen (Sgl) in eine neue Byte-Stream-Datei oder hängt die Daten an eine vorhandene Datei an. Das VI öffnet oder erstellt die Datei vor dem Schreiben und schließt sie hinterher. Sie können dieses VI zum Schreiben von skalierten Daten von Datenerfassungs-VIs, ohne die Repräsentierung zu ändern, einsetzen.



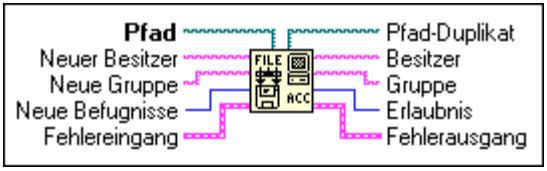
Beschreibungen von fortgeschrittenen Dateifunktionen

Die folgenden Funktionen sind auf der Unterpalette **Fortgeschrittene Dateifunktionen** verfügbar.



Zugriffsrechte

Legt für die in **Pfad** angegebene Datei oder das Verzeichnis den Besitzer, die Gruppe und die Berechtigungen fest und gibt diese aus. Wenn **Neuer Besitzer**, **Neue Gruppe** oder **Neue Berechtigungen** von Ihnen nicht vorgegeben werden, gibt diese Funktion die aktuellen Einstellungen unverändert aus.

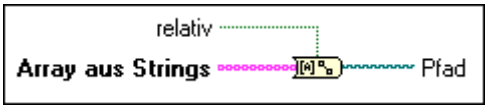


(Windows) Die Funktion Zugriffsrechte ignoriert **Neuer Besitzer** und **Neue Gruppe** und gibt für **Besitzer** und **Gruppe** leere Strings aus, weil Windows Besitzer und Gruppen nicht unterstützt.

(Macintosh) Wenn **Pfad** auf eine Datei verweist, ignoriert die Funktion Zugriffsrechte **Neuer Besitzer** und **Neue Gruppe** und gibt für **Besitzer** und **Gruppe** leere Strings aus, weil Macintosh für Dateien Besitzer und Gruppen nicht unterstützt.

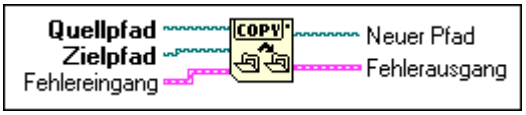
Array aus Strings in Pfad

Konvertiert **Array aus Strings** in einen relativen oder absoluten **Pfad**.



Kopieren

Kopiert die im **Qellpfad** vorgegebene Datei oder das Verzeichnis zu dem in **Zielpfad** vorgegebenen Speicherplatz. Wenn Sie ein Verzeichnis kopieren, kopiert diese Funktion den gesamten Inhalt rekursiv.

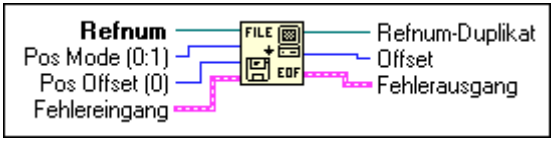


Löschen

Löscht die Datei oder das Verzeichnis, wie in **Pfad** angegeben. Wenn **Pfad** ein Verzeichnis angibt, das nicht leer ist, oder wenn Sie keine Schreibberechtigungen für sowohl die in **Pfad** angegebene Datei als auch das Verzeichnis und das übergeordnete Verzeichnis besitzen, entfernt diese Funktion das Verzeichnis nicht und gibt einen Fehler aus.

EOF (Dateiende)

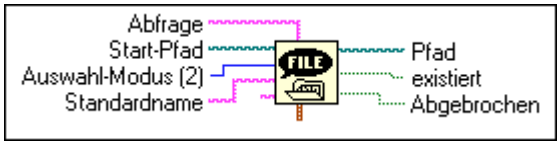
Setzt den logischen Marker EOF (Dateiende) der durch **Refnum** gekennzeichneten Datei, und gibt diesen aus. **Pos Modus** und **Pos Offset** geben die neue Stelle des EOF vor. Wird **Pos Modus** und **Pos Offset** von Ihnen nicht festgelegt, gibt diese Funktion den aktuellen unveränderten EOF aus. Diese Funktion gibt die Stelle des EOF immer auf den Dateianfang bezogen aus.



Das EOF (Dateiende) einer Datenprotokolldatei kann nicht festgelegt werden. Wenn **Refnum** eine Datenprotokolldatei kennzeichnet, können **Pos Modus** und **Pos Offset** nicht verbunden werden. Sie können aber dennoch den EOF-Marker einer Datenprotokolldatei erhalten, der Ihnen mitteilt, wie viele Aufzeichnungen in der Datei vorhanden sind.

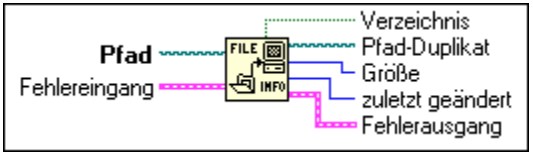
Dateidialog

Zeigt ein Dialogfeld an, mit dem der Pfad zu einer Datei oder einem Verzeichnis angegeben werden kann. Sie können mit Hilfe dieses Dialogfeldes vorhandene Dateien oder Verzeichnisse auswählen oder einen Speicherplatz und einen Namen für eine neue Datei oder ein neues Verzeichnis auswählen.



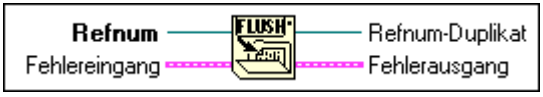
Datei-/Verzeichnis-Informationen

Gibt Informationen über die Datei oder das Verzeichnis, wie im **Pfad** angegeben, aus, einschließlich **Größe**, Tag der letzten Änderung und, ob es sich um ein Verzeichnis handelt.



Datei leeren

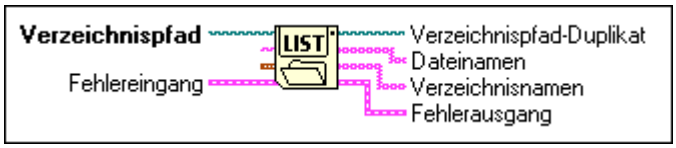
Schreibt den gesamten Puffer der durch **Refnum** angegebenen Datei auf die Festplatte und aktualisiert den Dateieintrag der mit **Refnum** verknüpften Datei im Verzeichnis. Die Datei bleibt geöffnet und **Refnum** gültig.



In eine Datei geschriebene Daten verbleiben oftmals im Puffer, bis der Puffer aufgefüllt ist oder die Datei geschlossen wird. Mit dieser Funktion wird das Betriebssystem gezwungen, jegliche Pufferdaten in die Datei zu schreiben.

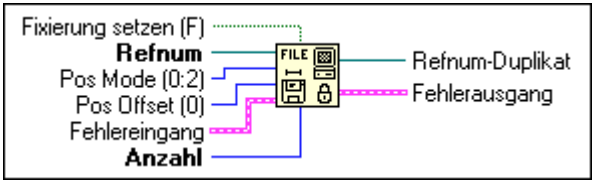
Verzeichnis auflisten

Gibt zwei Arrays aus Strings aus, die die Namen aller im **Verzeichnispfad** vorgefundenen Dateien und Verzeichnisse auflisten, wobei beide Arrays auf der Basis von **Pattern** gefiltert werden und das Array **Dateinamen** auf der Basis von dem vorgegebenen **Datenprotokolltyp** gefiltert wird.



Fixierungsbereich

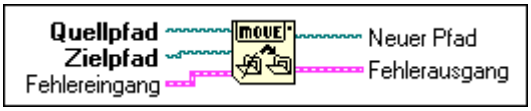
Fixiert einen Bereich einer durch **Refnum** angegebenen Datei oder gibt einen Bereich von dieser frei. Das Fixieren eines Dateibereichs verhindert sowohl das Lesen als auch Schreiben durch andere Nutzer, das Überschreiben von Berechtigungen für die Datei und für Keine Lese-/Schreibzugriffe, die mit **Refnum** assoziiert sind. Lesen Sie für eine vollständige Besprechung der Berechtigungen den Abschnitt *Überblick über die Datei-I/O-VIs und -Funktionen* weiter vorn in diesem Kapitel. Das Freigeben eines Dateibereichs entfernt das durch Fixieren eines Bereichs bedingte Umgehen der Berechtigungen, so daß es von den Berechtigungen der Datei und von Kein Lese-/Schreibzugriffe, die mit **Refnum** assoziiert sind, abhängt, ob andere Nutzer aus dem Dateibereich lesen oder in den Dateibereich schreiben können.



Es ist nicht möglich, einen Bereich einer Datenprotokolldatei zu fixieren.

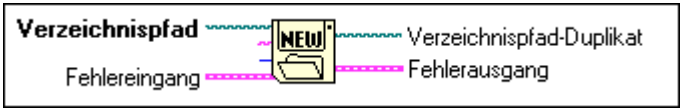
Bewegen

Verschiebt die Datei oder das Verzeichnis, wie in **Quellpfad** angegeben, an den in **Zielpfad** vorgegebenen Speicherplatz.



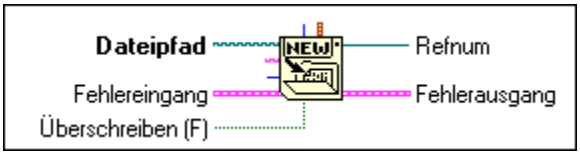
Neues Verzeichnis

Erstellt das in **Verzeichnispfad** vorgegebene Verzeichnis. Wenn eine Datei oder ein Verzeichnis an dem vorgegebenen Speicherplatz bereits vorhanden ist, gibt diese Funktion einen Fehler aus, anstatt die vorhandene Datei oder das vorhandene Verzeichnis zu überschreiben.



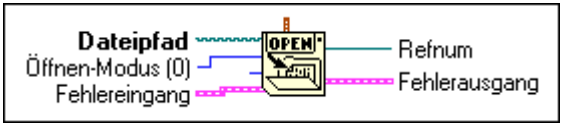
Neue Datei

Erstellt die in **Dateipfad** angegebene Datei und öffnet sie zum Lesen und Schreiben (ohne Berücksichtigung von **Erlaubnis**).



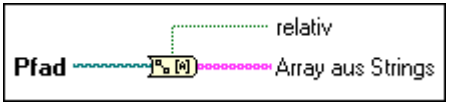
Datei öffnen

Öffnet die in **Dateipfad** angegebene Datei zum Lesen und/oder Schreiben.



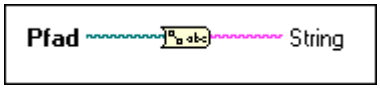
Pfad zu Array aus Strings

Konvertiert einen **Pfad** in ein **Array aus Strings** und zeigt an, ob es sich um einen relativen Pfad handelt.



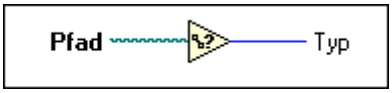
Pfad zu String

Konvertiert **Pfad** in einen String, der einen Pfad in dem Standardformat der Plattform angibt.



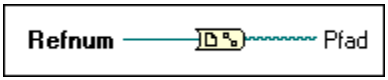
Pfadtyp

Gibt den Typ eines angegebenen Pfads aus und zeigt an, ob es sich um einen absoluten, relativen oder ungültigen Pfad handelt. Diese Funktion überprüft nur das Format eines Pfads, nicht ob der Pfad zu einer vorhandenen Datei oder ein vorhandenes Verzeichnis führt. Daher zeigt diese Funktion einen Pfad nur dann als ungültig an, wenn es sich um Kein Pfad handelt.



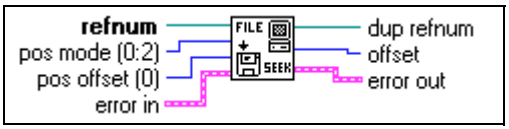
Refnum in Pfad

Gibt den mit **Refnum** assoziierten **Pfad** aus.



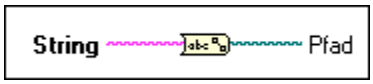
Suchen

Bewegt den aktuellen Dateimarker der durch **Refnum** angegebenen Datei entsprechend dem durch **Pos Modus** gewählten Modus an die durch **Pos Offset** angezeigte Stelle.



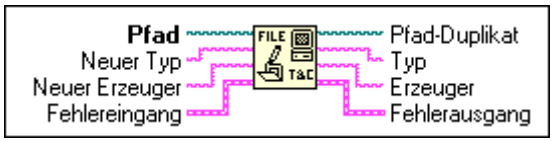
String zu Pfad

Konvertiert **String**, der einen Pfad im Standardformat der aktuellen Plattform angibt, in einen **Pfad**.



Typ und Ersteller

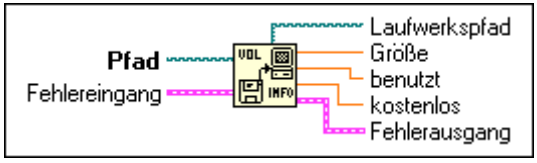
Liest und setzt den Typ und Ersteller der in **Pfad** angegebenen Datei. Dateityp und -ersteller sind Strings aus vier Zeichen. Wenn kein **Neuer Typ** oder **Neuer Erzeuger** festgelegt wird, gibt diese Funktion die aktuellen Einstellungen unverändert aus.



Windows und UNIX unterstützen keine Dateitypen und -ersteller. Der Versuch, den Typ oder Ersteller auf diesen Plattformen zu setzen, führt zu einem Fehler; Sie können jedoch trotzdem den Dateitypen und -ersteller auf diesen Plattformen erhalten. Wenn die angegebene Datei einen Namen trägt, der mit Zeichen endet, die von Typ und Ersteller als den Dateityp angebend erkannt werden (z.B., .vi für den LVIN-Dateityp und `.llb` für den LVAR-Dateityp), gibt diese Funktion diesen Typ in **Typ** und `LBVW` in **Erzeuger** aus. Ansonsten gibt die Funktion *????* sowohl in **Typ** als auch **Erzeuger** aus.

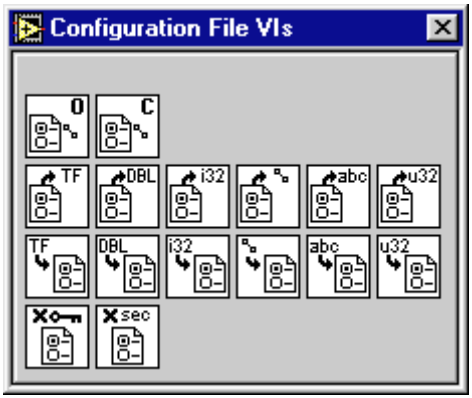
Datenträger-Info

Gibt Informationen über den Datenträger aus, der die Datei oder das Verzeichnis, wie in **Pfad** angegeben, enthält, einschließlich des auf dem Datenträger zur Verfügung gestellten Gesamtspeicherplatzes, der belegten Speichermenge und der freien Speichermenge in Bytes.



Konfigurationsdatei-VIs

Die Konfigurationsdatei-VIs bilden für Sie die Werkzeuge zum Erstellen, Ändern und Lesen einer plattformunabhängigen Konfigurationsdatei. Die folgende Abbildung zeigt die auf der Unterpalette **Konfigurationsdatei-VIs** verfügbaren Optionen.



Die Konfigurationsdatei-VIs arbeiten mit einer plattformunabhängigen Konfigurationsdatei, die dem Format nach dem Standard der Windows-Initialisierungsdateien (`.ini`) ähnelt.

Die Datei ist in Abschnitte eingeteilt, deren Namen in eckigen Klammern stehen. Jeder Abschnitt in einer Datei muß einen einmaligen Namen tragen. Innerhalb jedes Abschnitts befinden sich Schlüssel- und Wertepaare. Jeder Schlüssel innerhalb eines Abschnitts muß einen einmaligen Namen tragen.

Ein Beispiel einer Konfigurationsdatei mit Abschnitten `section 1` und `section 2` ist:

```
[section 1]
key1="string value 1"
key2="string value 2"
key3=53
[section 2]
key1=TRUE
key2=-12.3
key3="/c/temp/data.dat"
```

Die Konfigurationsdatei-VIs unterstützen folgende Datentypen:

- Strings
- Pfade
- Boolesch

- 64-Bit-Fließkommazahlen (Double)
- vorzeichenbehaftete 32-Bit-Integer-Werte (I32)
- vorzeichenlose 32-Bit-Integer-Werte (U32)

Stringdaten müssen in der Datei in doppelten Anführungszeichen stehen. Alle nichtdruckbaren Zeichen in dem String werden in der Datei mit ihren äquivalenten hexadezimalen Steuerzeichen gespeichert (z.B., `\0D` für Wagenrücklauf). Darüber hinaus werden Backslash-Zeichen in der Datei als doppelte Backslashes gespeichert (z.B., \\ für \).

Pfaddaten sind in einem plattformneutralen Format gespeichert. Dieses Format ist das Standard-UNIX-Format für Pfade. Die VIs interpretieren den absoluten Pfad `/c/temp/data.dat` auf den verschiedenen G-Plattformen, wie folgt:

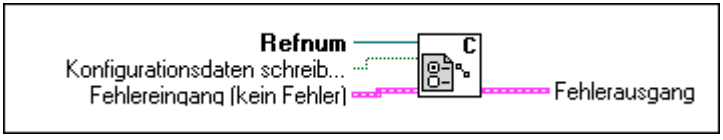
- Windows: `c\temp\data.dat`
- MacOS: `c:temp:data.dat`
- UNIX: `/c/temp/data.dat`

Daneben interpretieren die VIs den relativen Pfad `temp/data.dat`, wie folgt:

- Windows: `temp\data.dat`
- MacOS: `:temp:data.dat`
- UNIX: `temp/data.dat`

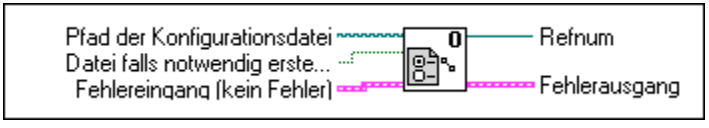
Konfigurationsdaten schließen

Schließt eine durch **Refnum** angegebene Referenz zu Konfigurationsdaten. Wenn **Konfigurationsdaten schreiben?** TRUE ist, schreibt das VI die Daten in die durch **Refnum** angegebene, plattformunabhängige Konfigurationsdatei.



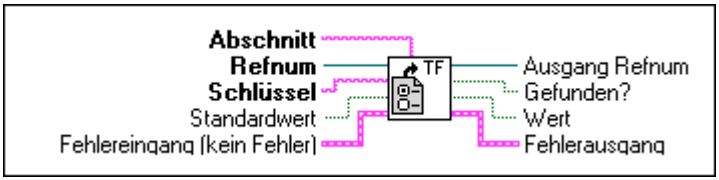
Konfigurationsdaten öffnen

Öffnet eine Referenz zu den in einer plattformunabhängigen Konfigurationsdatei befindlichen Konfigurationsdaten. Wenn die angegebene Datei nicht vorhanden ist und **Datei falls notwendig erstellen?** TRUE ist, erstellt das VI die Konfigurationsdatei auch.



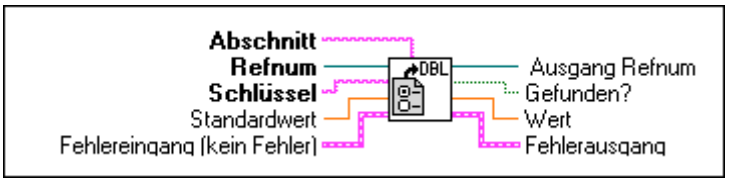
Schlüssel lesen (Boolesch)

Liest einen Booleschen **Wert**, der mit einem **Schlüssel** in einem vorgegebenen **Abschnitt** der durch **Refnum** angegebenen Daten verknüpft ist. Wenn **Schlüssel** nicht vorhanden ist, gibt das VI **Standardwert** aus.



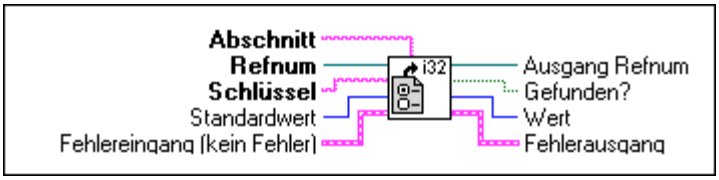
Schlüssel lesen (Double)

Liest einen 64-Bit-Fließkommazahlen-**Wert**, der mit **Schlüssel** verknüpft ist, in einem vorgegebenen **Abschnitt** der durch **Refnum** angegebenen Daten. Wenn **Schlüssel** nicht vorhanden ist, gibt das VI **Standardwert** aus.



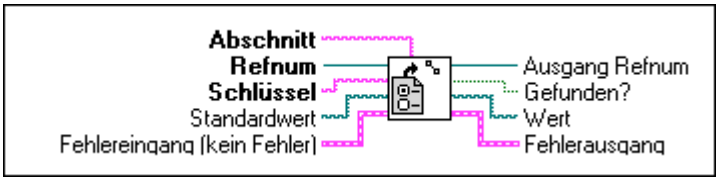
Schlüssel lesen (I32)

Liest einen vorzeichenbehafteten 32-Bit-Integer-**Wert** der mit **Schlüssel** verknüpft ist, in einem vorgegebenen **Abschnitt** von den durch **Refnum** angegebenen Konfigurationsdaten. Wenn der **Schlüssel** nicht vorhanden ist, gibt das VI **Standardwert** aus.



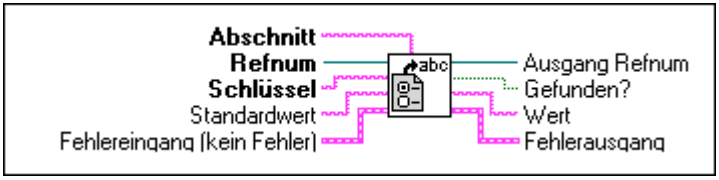
Schlüssel lesen (Pfad)

Liest einen Pfad-**Wert**, der mit einem **Schlüssel** verknüpft ist, in einem vorgegebenen **Abschnitt** der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** nicht vorhanden ist, gibt das VI **Standardwert** aus.



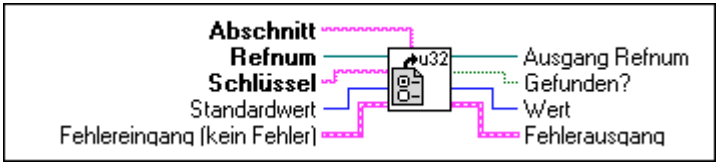
Schlüssel lesen (String)

Liest einen String-**Wert,** der mit einem **Schlüssel** verknüpft ist, in einem vorgegebenen **Abschnitt** der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** nicht vorhanden ist, gibt das VI **Standardwert** aus.



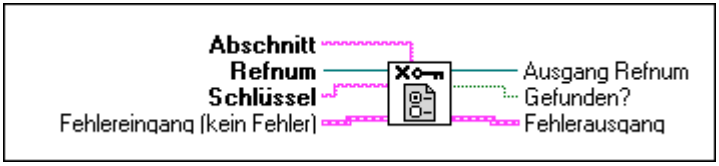
Schlüssel lesen (U32)

Liest einen vorzeichenlosen 32-Bit-**Wert**, der mit einem **Schlüssel** verknüpft ist, in einem vorgegebenen **Abschnitt** der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** nicht vorhanden ist, gibt das VI **Standardwert** aus.



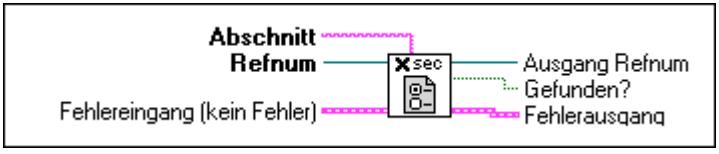
Schlüssel entfernen

Entfernt einen **Schlüssel** in einem vorgegebenen Abschnitt der durch **Refnum** angegebenen Konfigurationsdaten.



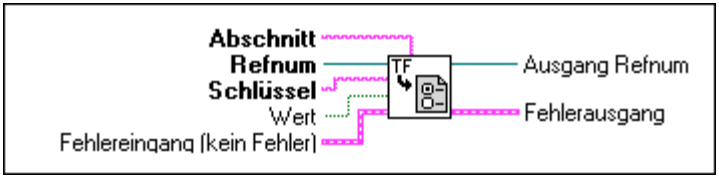
Abschnitt entfernen

Entfernt einen **Abschnitt** von den durch **Refnum** angegebenen Konfigurationsdaten.



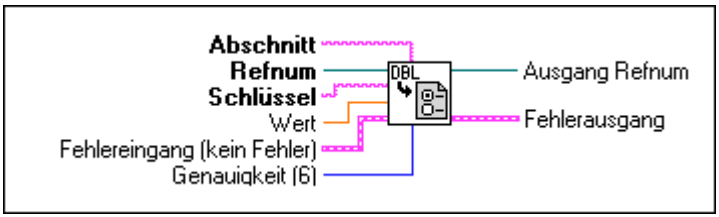
Schlüssel schreiben (Boolesch)

Schreibt einen Booleschen **Wert**, der mit **Schlüssel** verknüpft ist, in einen vorgegebenen Abschnitt der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** vorhanden ist, ersetzt das VI den vorhandenen Wert. Wenn **Schlüssel** nicht vorhanden ist, fügt das VI das Schlüssel/Wertepaar am Ende vom vorgegebenen **Abschnitt** hinzu. Wenn **Abschnitt** nicht vorhanden ist, fügt das VI **Abschnitt** mit dem Schlüssel/Wertepaar am Ende der Konfigurationsdaten hinzu.



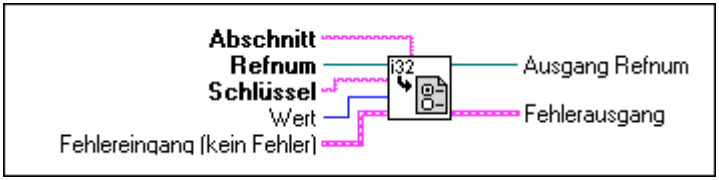
Schlüssel schreiben (Double)

Schreibt einen 64-Bit-Fließkommazahlen-**Wert**, der mit **Schlüssel** verknüpft ist, in einen vorgegebenen Abschnitt der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** vorhanden ist, ersetzt das VI den vorhandenen Wert. Wenn **Schlüssel** nicht vorhanden ist, fügt das VI das Schlüssel/Wertepaar am Ende vom vorgegebenen **Abschnitt** zu. Wenn **Abschnitt** nicht vorhanden ist, fügt das VI den Abschnitt mit dem Schlüssel/Wertepaar am Ende der Konfigurationsdaten hinzu.



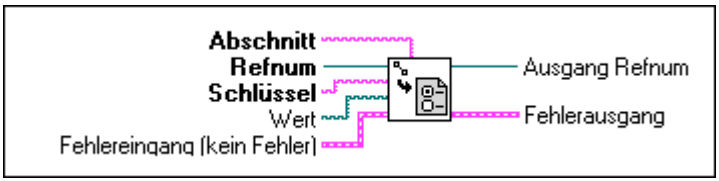
Schlüssel schreiben (I32)

Schreibt einen vorzeichenbehafteten 32-Bit-Integer-**Wert**, der mit **Schlüssel** verknüpft ist, in einen vorgegebenen Abschnitt der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** vorhanden ist, ersetzt das VI den vorhandenen Wert. Wenn **Schlüssel** nicht vorhanden ist, fügt das VI das Werte/Schlüsselpaar am Ende vom vorgegebenen **Abschnitt** hinzu. Wenn **Abschnitt** nicht vorhanden ist, fügt das VI **Abschnitt** mit dem Schlüssel/Wertepaar am Ende der Konfigurationsdaten hinzu.



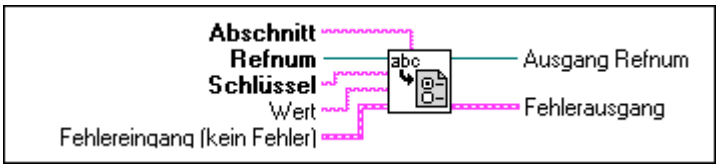
Schlüssel schreiben (Pfad)

Schreibt einen Pfad-**Wert**, der mit **Schlüssel** verknüpft ist, in einen vorgegebenen Abschnitt der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** vorhanden ist, ersetzt das VI den vorhandenen Wert. Wenn **Schlüssel** nicht vorhanden ist, fügt das VI das Schlüssel/Wertepaar an dem Ende vom vorgegebenen **Abschnitt** zu. Wenn **Abschnitt** nicht vorhanden ist, fügt das VI **Abschnitt** mit dem Schlüssel/Wertepaar am Ende der Konfigurationsdaten hinzu.



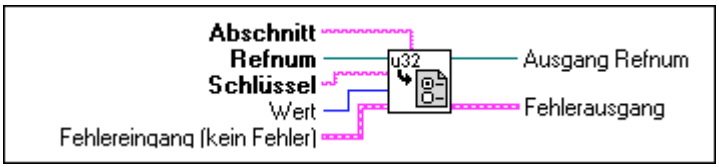
Schlüssel schreiben (String)

Schreibt einen String-**Wert**, der mit **Schlüssel** verknüpft ist, in einen vorgegebenen **Abschnitt** der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** vorhanden ist, ersetzt das VI den vorhandenen Wert. Wenn **Schlüssel** nicht vorhanden ist, fügt das VI das Schlüssel/Wertepaar an dem Ende vom vorgegebenen **Abschnitt** zu. Wenn **Abschnitt** nicht vorhanden ist, fügt das VI **Abschnitt** mit dem Schlüssel/Wertepaar am Ende der Konfigurationsdaten hinzu.



Schlüssel schreiben (U32)

Schreibt einen vorzeichenlosen 32-Bit-Integer-**Wert**, der mit **Schlüssel** verknüpft ist, in einen vorgegebenen **Abschnitt** der durch **Refnum** angegebenen Konfigurationsdaten. Wenn **Schlüssel** vorhanden ist, ersetzt das VI den vorhandenen Wert. Wenn **Schlüssel** nicht vorhanden ist, fügt das VI das Schlüssel/Wertepaar am Ende vom vorgegebenen **Abschnitt** zu. Wenn der Abschnitt nicht vorhanden ist, fügt das VI **Abschnitt** mit dem Schlüssel/Wertepaar am Ende der Konfigurationsdaten hinzu.

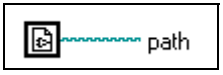


Beschreibungen der Dateikonstanten

Die folgenden Konstanten sind auf der Unterpalette **Dateikonstanten** verfügbar.

Pfadkonstante des aktuellen VIs

Gibt den Pfad zu der Datei aus, die das VI enthält, in dem diese Funktion auftritt. Wenn das VI in eine Anwendung eingebaut ist (unter Verwendung der Anwendungsbilder-Bibliothek), gibt die Funktion den Pfad zu dem VI in der Anwendungsdatei aus und behandelt die Anwendungsdatei als eine VI-Bibliothek.



Konstante Standardverzeichnis

Gibt den Pfad zu Ihrem Standardverzeichnis aus. Das Verzeichnis, das vom Dateidialog am Anfang angezeigt wird, bildet das Standardverzeichnis. Das Verzeichnis wird im Dialogfeld Voreinstellungen (**Bearbeiten»Voreinstellungen**) unter **Pfaden** festgelegt.

Leerer Pfad

Gibt einen leeren Pfad aus.

Kein Pfad

Gibt einen Pfad aus, dessen Wert Kein Pfad ist. Sie können diesen Pfad als eine Ausgabe von Strukturen und SubVIs einsetzen, wenn ein Fehler auftritt.

Keine Refnum

Gibt eine Refnum aus, dessen Wert Kein Refnum ist. Sie können diese Refnum als eine Ausgabe von Strukturen und SubVIs einsetzen, wenn ein Fehler auftritt.

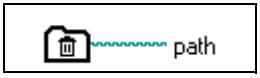
Pfadkonstante

Verwenden Sie diese, um einen konstanten Verzeichnis- oder Dateipfad an das Blockdiagramm zu geben. Setzen Sie diesen Wert, indem Sie mit dem Bedienwerkzeug innerhalb der Konstante klicken und den Wert eingeben. Verwenden Sie die Standardsyntax für Dateipfade der jeweiligen Plattform. Sie können den Wert der Pfadkonstante auf Kein Pfad setzen, indem Sie mit dem Bedienwerkzeug auf das Pfadsymbol klicken und aus dem auftauchenden Menü **Kein Pfad** auswählen. Für weitere Informationen zum Einsatz von dem Wert Kein Pfad lesen Sie bitte im *LabVIEW Benutzerhandbuch* im Kapitel 6, *Strings und Datei I/O*, den Abschnitt *Pfade und Refnum*.

Der Wert dieser Konstante kann solange nicht geändert werden, wie das VI abläuft. Sie können diese Konstante mit einem Label versehen.

Konstante temporäres Verzeichnis

Gibt den Pfad zum temporären Verzeichnis aus. Bei dem temporären Verzeichnis handelt es sich um das Verzeichnis, in dem vorübergehende Informationen, die der Nutzer oder das Betriebssystem normalerweise periodisch löscht, gespeichert sind. Das Verzeichnis wird im Dialogfeld G-Voreinstellungen (**Bearbeiten»Voreinstellungen**) unter **Pfaden** festgelegt.



Konstante VI Bibliothek

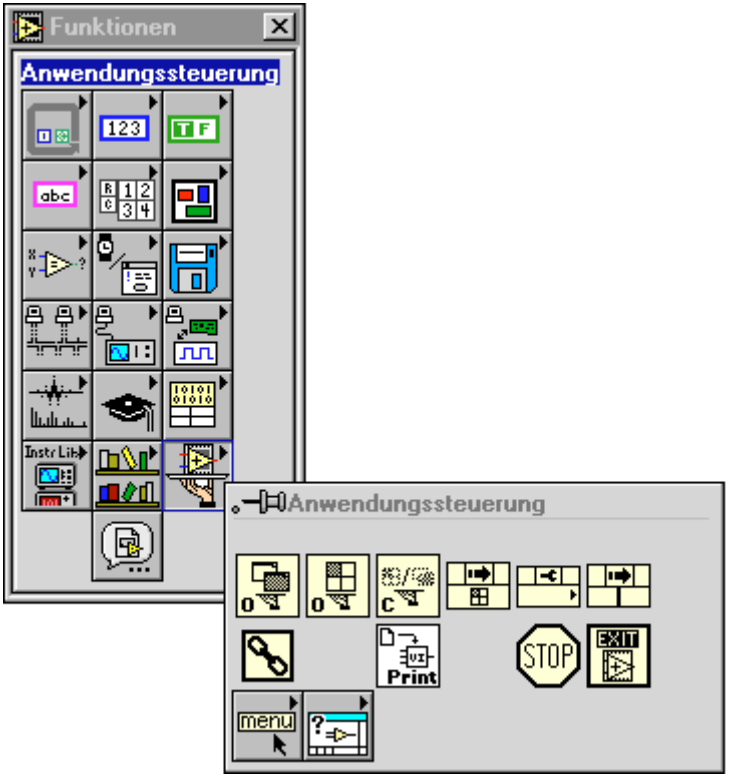
Gibt den Pfad zu dem Verzeichnis der VI-Bibliothek für die aktuelle Entwicklungsbibliothek auf dem aktuellen Computer aus. Das Verzeichnis wird im Dialogfeld Voreinstellungen (**Bearbeiten»Voreinstellungen**) unter **Pfaden** festgelegt. Wenn Sie eine Anwendung unter Verwendung der Anwendungsbilder-Bibliotheken erstellen, ist dieser Pfad der Pfad des Verzeichnisses, das die Anwendung enthält.

12

Funktionen zur Anwendungssteuerung

Dieses Kapitel beschreibt die Anwendungssteuerungs-Funktionen.

Sie können auf die **Anwendungssteuerungs**-Palette über das Menü **Funktionen»Anwendungssteuerung**, wie in der folgenden Abbildung dargestellt, zugreifen.



Die Palette **Anwendungssteuerung** umfaßt die folgenden Unterpaletten:

- Hilfefunktionen
- Menüfunktionen

Anwendungssteuerungsfunktionen

Die folgenden Funktionen zur Anwendungssteuerung sind verfügbar.

Aufruf über Referenz

Die Funktion Aufruf über Referenz ist einem SubVI-Knoten sehr ähnlich: beide können zum Aufruf eines VIs verwendet werden. Es besteht jedoch ein wesentlicher Unterschied. Wenn Sie einen SubVI-Knoten auf dem Diagramm ablegen, legen Sie fest, welches VI aufgerufen wird.

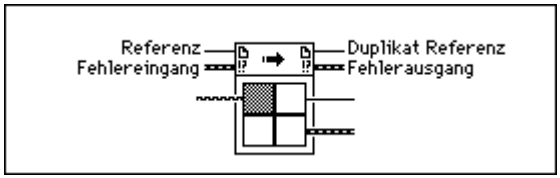
Bei der Funktion Aufruf über Referenz legt der Endbenutzer über die **Referenzeingabe** fest, welches VI während der Ausführungszeit aufgerufen wird. Die Funktion Aufruf über Referenz kann u.U. ein VI auf einem anderen Computer aufrufen.

Auf der Oberseite der Funktion Aufruf über Referenz befinden sich vier Anschlüsse: ein Eingangs-/Ausgangs-Paar von Durchfluß-VI-Referenzanschlüssen und ein Eingangs-/Ausgangs-Paar von Durchfluß-Fehlerclustern. Der VI-Referenzeingang läßt nur Verbindungen mit strikt typisierten VI-Referenzen zu. Unterhalb dieser Anschlüsse befindet sich eine Fläche, in der sich ein Anschlußfeld befindet, das mit dem eines VIs identisch ist und seine Anschlüsse (anstelle seines Icons) zeigt. Das Anschlußfeld der strikt typisierten VI-Referenz legt die Pattern und die Datentypen dieses Anschlußfeldes fest. Verbinden Sie diese Anschlüsse wie die eines normalen SubVIs.

Solange keiner der Anschlüsse auf dem Anschlußfeld mit einer Verbindung versehen ist, paßt sich das Anschlußfeld automatisch an das Anschlußfeld der Eingangs-VI-Referenz an. Wenn allerdings einer der Anschlüsse verbunden ist, paßt sich der Knoten nicht automatisch an und muß ausdrücklich in dem Anschlußfeld geändert werden (möglicherweise durch Auflösen dieser Verbindung), indem das Popup-Menü aufgerufen und Menüpunkt **An Referenzeingang anpassen** ausgewählt wird.

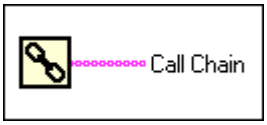
Während der Ausführungszeit tritt beim Aufruf des VIs ein kleiner Verarbeitungsaufwand auf, der beim Aufruf eines normalen SubVIs nicht notwendig ist. Dieser Verarbeitungsaufwand ergibt sich aus dem Bestätigen der VI-Referenz und einiger anderer Einzelheiten. Dieser Verarbeitungsaufwand eines VI-Aufrufs im lokalen LabVIEW sollte jedoch für alle außer den kleinsten SubVIs unerheblich sein. Der Aufruf eines VIs in einer anderen LabVIEW-Anwendung (am anderen Ende des Netzwerks) verursacht

beträchtlichen Verarbeitungsaufwand. Der **Referenz**-Eingang legt das von der Funktion Aufruf über Referenz aufgerufene VI fest.



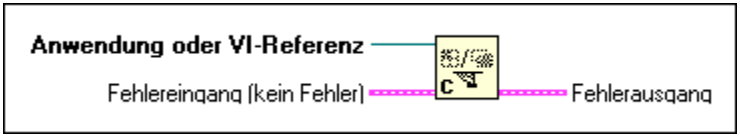
Aufrufkette

Gibt eine Referenz an eine LabVIEW-Anwendung oder ein VI aus.



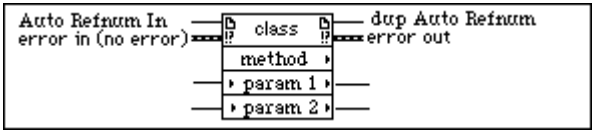
Anwendung oder VI-Referenz schließen

Schließt ein geöffnetes VI oder die TCP-Verbindung zu einer geöffneten Kopie von LabVIEW.



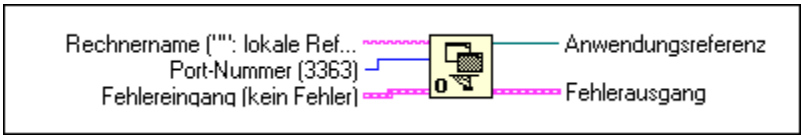
Methodenknoten

Aktiviert eine Methode oder eine Aktion eines VIs. Die Mehrzahl der Methoden sind mit Parametern verknüpft. Die Methode wird über **Methode** im Popup-Menü des Knotens ausgewählt. Nachdem die Methode ausgewählt ist, erscheinen die verknüpften Parameter in der folgenden Abbildung. Sie können die Parameterwerte einstellen und einsehen. Bei den Parametern mit weißem Hintergrund handelt es sich um erforderliche Eingaben und bei denen mit grauem Hintergrund um empfohlene Eingaben.



Anwendungsreferenz öffnen

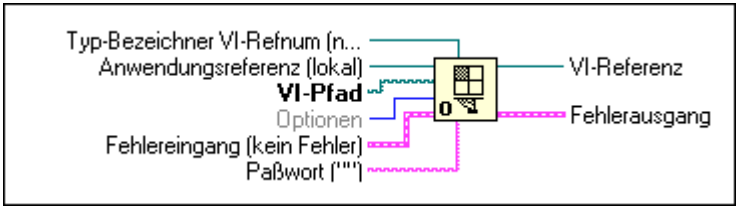
Gibt eine Referenz zu einer auf dem angegebenen Computer laufenden VI-Server-Anwendung aus. Wenn kein Wert für **Rechnername** angegeben wird, gibt diese Funktion eine Referenz zu der lokalen LabVIEW-Anwendung, in der die Funktion abläuft, aus.



Sie können den **Anwendungsreferenz**-Ausgang als Eingang des Eigenschaftsknotens-Knotens und des Methodenknotens verwenden, um Eigenschaften einzusehen oder einzustellen und Methoden auf der Anwendung zu aktivieren. Durch Verwendung von Anwendungsreferenz als Eingang der Funktion Geöffnete VI-Referenz können die Referenzen zu den VIs dieser Anwendung eingesehen werden. Schließen Sie die Referenz mit der Funktion Anwendung schließen oder VI-Referenz. Wird vergessen, diese Referenz zu schließen, wird sie automatisch geschlossen, wenn das mit dieser Funktion verknüpfte Top-Level-VI die Ausführung beendet. Es gehört jedoch zur guten Arbeitsweise, die mit dem Aufrechterhalten der Verbindung benutzten Ressourcen durch Schließen der Referenz zu sparen, sobald Sie dessen Nutzung abgeschlossen haben.

VI-Referenz öffnen

Gibt eine Referenz zu einem VI aus, das durch einen Namenstring oder einen Pfad zum Speicherplatz angegebenen ist.



Weitere Referenzen zu VIs in anderen LabVIEW-Anwendungen können durch Verbinden dieser Funktion mit einer **Anwendungsreferenz** (von der Funktion Anwendungsreferenz öffnen bereitgestellt) erhalten werden. In diesem Fall verweist **Pfadeingabe** zu dem Dateisystem auf dem entfernten LabVIEW-Computer. Wenn eine Referenz mit der lokalen LabVIEW-Anwendung verbunden wird, ergibt sich dasselbe Verhalten, als ob keine Verbindungen mit dem Eingang **Anwendungsreferenz** hergestellt wurden.

Wenn Sie beabsichtigen, Bearbeitungsoperationen an dem referenzierten VI vorzunehmen, und wenn das VI ein paßwortgeschütztes Diagramm hat, können Sie ein **Paßwort** an den Paßwortstringeingang liefern. Wenn Sie das verkehrte Paßwort liefern, gibt die Funktion VI-Referenz öffnen einen Fehler und eine ungültige VI-Referenz aus. Wenn Sie kein Paßwort beim Öffnen eines paßwortgeschützten VIs bereitstellen, können Sie nichtsdestoweniger die Referenz erhalten, doch nur Operationen ausführen, die das VI nicht bearbeiten.

Wenn Sie beabsichtigen, das angegebene VI durch die Funktion Aufruf durch Referenz aufzurufen, müssen Sie eine strikt typisierte VI-Referenz mit dem Eingang **Typ Bezeichner** verbinden. Die Funktion ignoriert den Wert dieses Eingangs. Nur der Typ des Eingangs - die Informationen des Anschlußfelds - wird verwendet. Durch Angeben dieses Typs überprüft die Funktion VI-Referenz öffnen während der Ablaufzeit, ob das Anschlußfeld des angegebenen VIs mit der Eingabe vom **Typ Bezeichner** übereinstimmt.

☞ **Hinweis** ***Es ist möglich, einen Allgemeingültigen VI-Refnum-Typ-Bezeichner mit dem Eingang Typen-Bezeichner zu verbinden. Das Ergebnis davon ist dasselbe Verhalten, als ob die Typ-Bezeichner-Eingabe überhaupt nicht verbunden wurde.***

Wird ein Typ-Bezeichner-Eingang mit einer strikt typisierten VI-Refnum verbunden, muß das VI mehrere Voraussetzungen erfüllen, ehe die VI-Referenz erfolgreich zurückgegeben wird:

- Das VI darf aus keinem Grund geteilt werden.
- Das VI muß als SubVI ausführbar sein, d.h., es darf nicht als Top-Level-VI aktiv sein (es sei denn, das VI ist ablaufinvariant).
- Das Anschlußfeld des VIs muß mit dem vom Typ-Bezeichner übereinstimmen.

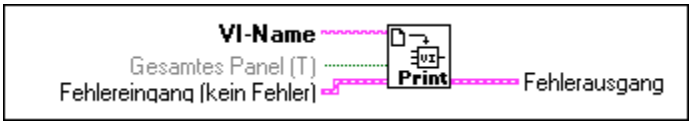
Wird vergessen, diese Referenz zu schließen, wird sie automatisch geschlossen, wenn das mit dieser Funktion verknüpfte Top-Level-VI die Ausführung beendet. Es gehört jedoch zur guten Arbeitsweise, die mit dem Aufrechterhalten der Verbindung benutzten Ressourcen durch Schließen der Referenz zu sparen, sobald Sie dessen Nutzung abgeschlossen haben.

Wenn eine strikt typisierte Referenz mit einem ablaufinvarianten VI verbunden wird, wird dieser Referenz ein dedizierter Speicherplatz zugewiesen. Dieser Speicherplatz wird immer in Verbindung mit der Ausgabe der VI-Referenz verwendet. Das kann zu neuen Verhaltensweisen führen, mit denen Sie u.U. in LabVIEW nicht vertraut sind. Parallelaufrufe (die die Funktion Aufruf über Referenz verwenden) an ein ablaufvariantes VI, das dieselbe VI-Referenz verwendet, laufen nicht parallel ab, sondern seriell, einer nach dem anderen.

Beachten Sie, daß eine VI-Referenz dem in anderen Sprachen bekannten Funktionszeiger ähnelt. In LabVIEW können diese Funktionszeiger auch dazu verwendet werde, VIs über das Netzwerk verteilt aufzurufen.

Panel drucken

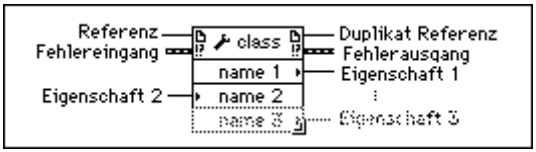
Erzeugt denselben Ausdruck wie der programmierte Druck nach Abschluß, kann aber von anderen VIs und zu anderen Zeitpunkten als dem des Abschlusses aufgerufen werden. Standardmäßig wird das gesamte Panel ausgedruckt, nicht nur der sichtbare Teil des Fensters. Dabei wird angenommen, daß das VI zwar geladen, doch das Fenster nicht notwendigerweise geöffnet ist.



Eigenschaftsknoten

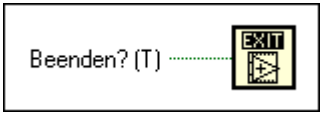
Setzt (schreibt) und erhält (liest) Informationen über VI- und Anwendungseigenschaften. Um das VI oder die Anwendungsklasse auszuwählen, wählen Sie vom Untermenü **VI-Serverklasse auswählen** vom Popup-Menü. Wählen Sie **Anwendung**, um eine Anwendungsklasse auszuwählen. Um eine VI-Klasse auszuwählen, wählen Sie **Virtuelles Instrument** oder verbinden Sie die VI- oder Anwendungs-Refnum mit **Referenz**, wodurch die Knotenauswahlen dementsprechend wechseln.

Um eine spezielle Eigenschaft auszuwählen, rufen Sie eins der Popup-Menüs der Anschlüsse **Name** auf, und wählen Sie **Eigenschaften**. Um Eigenschaftsinformationen einzustellen, wählen Sie **In Schreiben ändern** im Popup-Menü aus, und um Eigenschaftsinformationen zu erhalten, wählen Sie im Popup-Menü **In Lesen ändern** aus. Manche Eigenschaften sind schreibgeschützt; in diesem Fall finden Sie keine Option **In Schreiben ändern** im Popup-Menü. Der Eigenschaftensknoten funktioniert in derselben Art und Weise wie der Attributknoten. Wenn Sie Objekte zu dem Knoten hinzufügen möchten, rufen Sie das Popup-Menü auf, und wählen Sie **Element hinzufügen**, oder klicken Sie auf den Knoten und ziehen Sie diesen, um die Anzahl der Objekte in diesem Knoten zu erweitern. Wenn dieser Knoten abläuft, werden die Eigenschaften in der Reihenfolge von oben nach unten bearbeitet. Wenn bei einer der Eigenschaften ein Fehler auftritt, hält der Knoten an der Eigenschaft an und gibt einen Fehler aus. Es werden keine weiteren Eigenschaften bearbeitet. Der Fehlerstring berichtet, welche Eigenschaft den Fehler verursacht hat. Denken Sie daran, daß Sie die Eigenschaftswerte einstellen, wenn die kleinen Richtungspfeile einer Eigenschaft auf der linken Seite sind. Wenn die kleinen Richtungspfeile auf der rechten Seite sind, erhalten Sie die Eigenschaftswerte. Jeder Eigenschaftenname hat einen kurzen und einen langen Namen, der über den Menüpunkt **Namen ändern** im Popup-Menü geändert werden kann. Kein Name bildet ein weiteres Namenformat, wobei nur der Typ jeder Eigenschaft angezeigt wird.



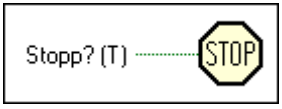
Beenden

Stoppt alle ablaufenden VIs und beendet die aktuelle Sitzung in LabVIEW. Diese Funktion fährt nur LabVIEW herunter und beeinflußt keine anderen Anwendungen. Diese Funktion stoppt alle ablaufenden VIs in derselben Weise wie die Stoppfunktion.



Stopp

Stoppt das VI, in dem es ausgeführt wird, genauso als ob auf die **Stopptaste** in der Symbolleiste geklickt wurde. Wenn der Eingang verbunden ist, tritt ein Stopp nur ein, wenn der Eingabewert TRUE ist. Ist der Eingang nicht verbunden, tritt der Stopp ein, sobald der Knoten, der gerade abläuft, beendet ist.



Wenn es notwendig ist, die Ausführung aller VIs in einer Hierarchie auf dem Blockdiagramm abzubrechen, kann diese Funktion eingesetzt werden; Sie müssen dabei jedoch Vorsicht walten lassen. Stellen Sie vor dem Aufruf der Stoppfunktion mit einer Eingabe von TRUE sicher, daß zunächst alle abschließenden Aufgaben des VIs fertiggestellt werden, z.B. Dateien schließen, Einstellen von sicheren Werten für die gesteuerten Geräte usw. Wenn die Stoppfunktion in einem SubVI plaziert wird, ist sicherzustellen, daß andere Benutzer das Verhalten dieser Funktion verstehen, weil sie deren VI-Hierarchien zur Abbrechung der Ausführung veranlaßt.

Sie sollten diese Funktion im allgemeinen vermeiden, wenn Sie in Ihrem VI ein eingebautes Abschlußprotokoll haben. Zum Beispiel sollten die I/O-Operationen in While-Schleifen ausgeführt werden, so daß das VI die Schleife bei einem I/O-Fehler beenden kann. Daneben sollten Sie auch statt der Stoppfunktion eine Boolesche Stoppsteuerung auf dem Frontpanel zum Beenden von Schleifen auf Antrag des Benutzers in Erwägung ziehen.

Beschreibungen der Hilfefunktionen

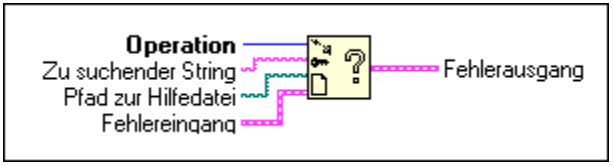
Die folgende Abbildung zeigt die auf der Unterpalette **Hilfe** verfügbaren Optionen.

Hilfefenster kontrollieren

Modifiziert das **Hilfe**fenster durch Einblenden, Ausblenden und Neupositionieren.

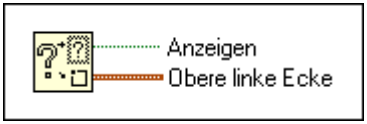
Online-Hilfe kontrollieren

Steuert das Online-Hilfe-System durch das Anzeigen des Inhaltsverzeichnisses einer Hilfedatei, das Anwählen eines bestimmten Themas in der Hilfedatei oder das Schließen des Online-Hilfe-Systems.



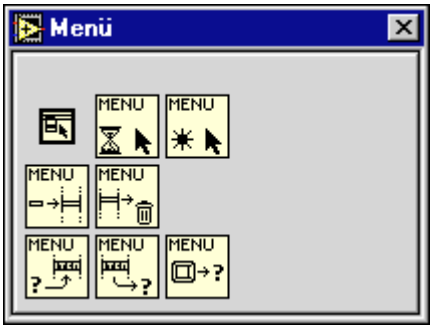
Status der Hilfefenster erhalten

Gibt den Status und die Positionsinformationen des **Hilfefensters** aus.



Menüfunktionen

Die folgende Abbildung zeigt die auf der Unterpalette **Menü** verfügbaren Optionen.

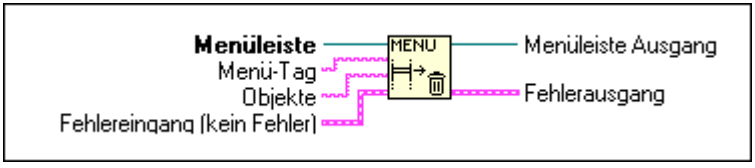


Die Menüfunktionen arbeiten mit Menüs, die durch Refnums gekennzeichnet sind. Eine Refnum des Menüs eines VIs kommt von dem konstanten Aktuelles VI-Menü. Die Einträge sind durch ein Eintrags-Tag (String) und manchmal durch einen Eintragspfad (String) gekennzeichnet, welcher eine Auflistung von Eintrags-Tags vom Menüstamm bis zu dem Eintrag ist, wobei die Tags durch Doppelpunkte voneinander abgeteilt sind.

Die folgenden Menüfunktionen sind verfügbar.

Menüeinträge löschen

Löscht Menüeinträge von der Menüleiste oder einem Untermenü in der Menüleiste.

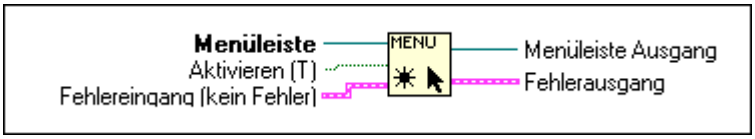


Wenn **Menü-Tag** angegeben ist, werden die Einträge von dem durch **Menü-Tag** angegebenen Untermenü gelöscht, ansonsten werden die Einträge von der Menüleiste gelöscht. Die Funktion gibt einen Fehler aus, wenn **Menü-Tag** oder einer der angegebenen Einträge nicht gefunden werden konnte.

Einträge können ein Tag (String) eines vorhandenen Eintrags sein, ein Array aus Tags von vorhandenen Einträgen, ein Stellenindex (auf Null-basierender Integer-Wert) eines Eintrags in dem Menü oder ein Array aus Stellenindizes der Einträge in dem Menü. Wenn **Einträge** nicht verbunden ist, werden alle Einträge in dem Menü gelöscht. Wenn ein Untermenü in einem der angegebenen Einträge vorhanden ist, wird das Untermenü automatisch samt Inhalt gelöscht. Da Trennzeichen keine eindeutigen Tags tragen, werden sie am besten mit Hilfe ihrer Stellenindizes gelöscht.

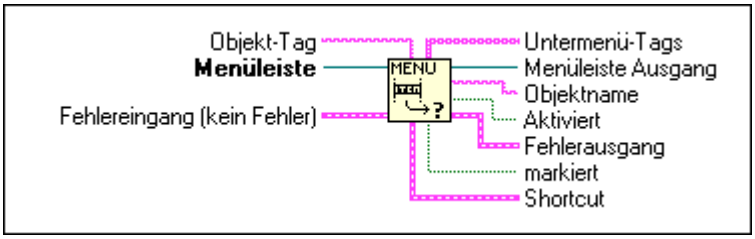
Aktivieren der Menüüberwachung

Aktiviert oder deaktiviert die Überwachung von Menüauswahlen.



Menüpunkt-Info bekommen

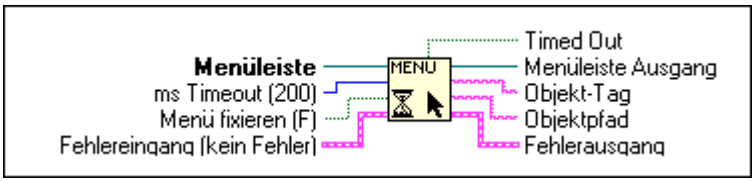
Gibt die Attribute des mit **Objekt-Tag** angegebenen Menüeintrags aus.



Bei den Eintragsattributen handelt es sich um **Objektname** (der String, der in dem Menü angezeigt wird), **Aktiviert** (falls nicht aktiviert, ist dieser Eintrag grau ausgeblendet), **Ausgewählt** (gibt an, ob das Kontrollkästchen neben dem Eintrag markiert ist) und **Shortcut** (Direktzugriffstaste). Wenn der Eintrag über ein Untermenü verfügt, werden dessen Eintrags-Tags in **Untermenü-Tags** als String-Array ausgegeben. Wenn **Objekt-Tag** nicht verbunden ist, werden die Einträge der Menüleiste ausgegeben. Wenn **Objekt-Tag** ungültig ist, wird ein Fehler ausgegeben.

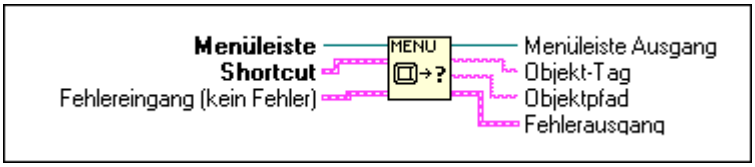
Menüauswahl bekommen

Gibt den **Objekt-Tag** des zuletzt gewählten Menüeintrags aus und wartet hierbei optional `timeout milliseconds` (Zeitbegrenzung in Millisekunden). **Objektpfad** ist ein String aus der Stelle des Eintrags in der Menühierarchie und trägt das Format einer Auflistung von Menü-Tags, die durch Doppelpunkte (:) abgeteilt sind. Wenn das Blockmenü auf `True` eingestellt ist, wird ein Eintrags-Tag nach dem Lesen aus der Menüauswahl herausgenommen.



Menü-Kurzinformation bekommen

Gibt den über einen jeweiligen Shortcut erreichbaren Menüeintrag aus.

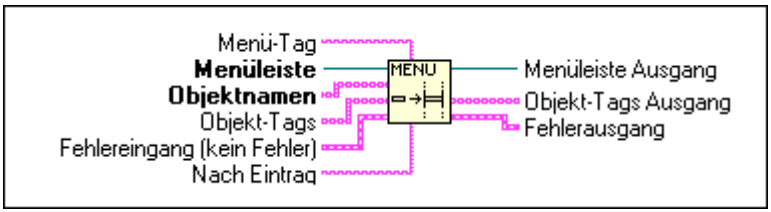


Objektpfad ist ein String aus Menüeintrags-Tags, die durch Doppelpunkte (:) abgeteilt sind.

Shortcut besteht aus einem String (Taste) und einen Booleschen Wert (der angibt, ob die Umschalt-Taste einbezogen ist oder nicht).

Menüeinträge einfügen

Fügt Menüeinträge in eine Menüleiste oder ein Untermenü in der Menüleiste ein.



Menü-Tag gibt an, wo die Einträge eingefügt werden. Wenn **Menü-Tag** nicht angegeben wird, werden die Menüeinträge in die **Menüleiste** eingefügt.

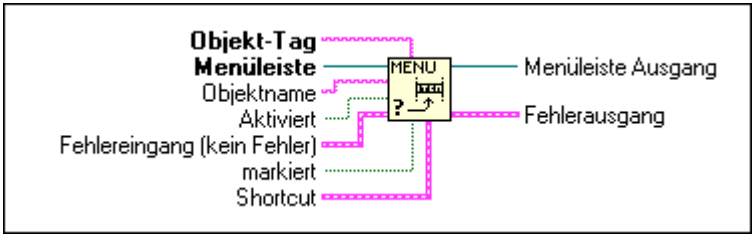
Objektnamen und **Objekt-Tags** kennzeichnen die in das Menü einzufügenden Einträge. Der Typ von **Objektname** und **Objekt-Tag** kann ein String-Array (zum Einfügen von mehreren Einträgen) oder nur ein String (zum Einfügen von einem einzelnen Eintrag) sein. Sie können entweder **Objektnamen** oder **Objekt-Tags** verbinden, wodurch sowohl die Namen und Tags dieselben Werte erhalten. Wenn es für Sie erforderlich ist, daß jeder Eintrag einen anderen Namen und Tag trägt, müssen getrennte Werte für **Objektnamen** und **Objekt-Tags** eingespeist werden.

Nach Eintrag kennzeichnet die Stelle, an der Einträge eingefügt werden. **Nach Eintrag** kann ein Tag (String) eines vorhandenen Eintrags oder ein Stellenindex (auf Null basierender Integer-Wert) in dem Menü sein. Um einen Eintrag am Anfang des Menüs einzufügen, ist eine Zahl kleiner als 0 in **Nach Eintrag** einzugeben. Um einen Eintrag am Ende des Menüs einzufügen, ist eine Zahl größer als die Anzahl der Einträge in dem Menü einzugeben. Sie können mit Hilfe des Anwendungs-Tag APP_SEPARATOR ein Trennzeichen einfügen. Die Funktion stellt immer sicher, daß die Tags aller eingefügten Menüeinträge einmalig sind, indem an die bereitgestellten Einträge gegebenenfalls Zahlen angehängt werden.

Objekt-Tags-Ausgang gibt die tatsächlichen Einträge der eingefügten Einträge aus. Wenn **Menü-Tag** oder **Nach Eintrag** (Eintrag) nicht gefunden wird, gibt die Funktion einen Fehler aus.

Menüpunkt-Info einstellen

Setzt die Attribute eines durch **Menü** und **Objekt-Tag** angegebenen Menüeintrags. Bei den Eintragsattributen handelt es sich um **Objektname** (der in dem Menü angezeigte String), **Aktiviert** (falls nicht aktiviert, grau ausgeblendet), **Ausgewählt** (gibt an, ob das Kontrollkästchen neben dem Eintrag markiert ist) und **Shortcut** (Direktzugriffstaste). Attribute, die nicht verbunden sind, bleiben unverändert. Wenn ein **Objekt-Tag** ungültig ist, wird ein Fehler ausgegeben.

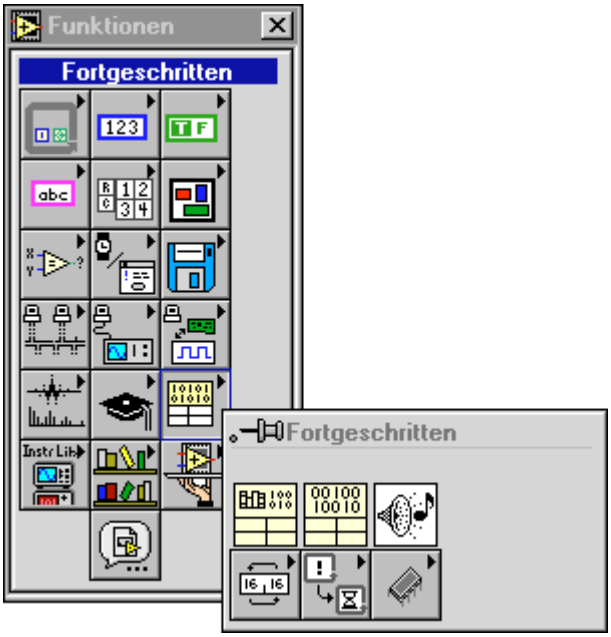


13

Fortgeschrittene Funktionen

Dieses Kapitel beschreibt die Funktionen, die fortgeschrittene Operationen ausführen. Daneben beschreibt dieses Kapitel die Funktionen zur Datenmanipulation und -synchronisation und die VIs VI Bedienelement und Speicher.

Auf die Palette **Fortgeschritten** kann, wie in der folgenden Abbildung gezeigt, über das Menü **Funktionen»Fortgeschritten** zugegriffen werden.



Die Funktionen Fortgeschritten umfassen die folgenden Unterpaletten:

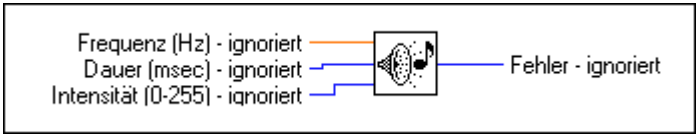
- Datenmanipulation
- Speicher
- Synchronisation

Beschreibungen der Fortgeschrittenen Funktionen

Die folgenden Fortgeschrittenen Funktionen sind verfügbar.

Piepton

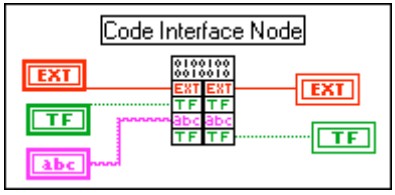
Veranlaßt das System, einen Piepton auszugeben. Die Tonfrequenz kann in Hertz, die Dauer in Millisekunden und die Intensität als ein Wert zwischen 0 und 255, wobei 255 am lautesten ist, vorgegeben werden. Obwohl dieses VI auf allen Plattformen angezeigt wird, funktionieren die Parameter Frequenz, Dauer und Intensität nur auf Macintosh-Computern.



Code Interface Node

Ruft in konventionellen Programmiersprachen geschriebenen Code, wie z.B. in C, direkt vom Blockdiagramm auf. Code Interface Nodes (CINs) ermöglichen es Ihnen, in anderen Sprachen geschriebene Algorithmen oder plattform-spezifische Funktionen oder Hardware, die G nicht direkt unterstützt, zu verwenden.

CINs sind vergrößerbar und zeigen, ähnlich der Bündelfunktion, Datentypen der verbundenen Eingänge und Ausgänge an. Die folgende Abbildung zeigt die CIN-Funktion.



Das LabVIEW-Interface zu externem Code ist sehr leistungsstark. Sie können eine beliebige Anzahl Parameter an externen Code weiterleiten oder von externem Code übernehmen, und jeder Parameter kann hierbei jedem beliebigen Datentypen von G entsprechen. LabVIEW stellt verschiedene Bibliotheken mit Routinen zur Verfügung, die die Arbeit mit G-Datentypen erleichtern. Diese Routinen unterstützen die Speicherzuweisung, Dateimanipulation und Datentypumwandlung.

Wenn Sie ein VI umwandeln, das einen CIN zu einer anderen Plattform enthält, muß der Code für die neue Plattform erneut kompiliert werden, weil CINs einen für eine andere Programmiersprache kompilierten Code verwenden. Sie können für ein CIN einen Quellcode schreiben, womit der CIN plattformunabhängig ist und nur der Quellcode zur Umwandlung

für die andere Plattform erneut kompiliert werden muß. Sie finden Beispiele für CINs in `examples\cins`.

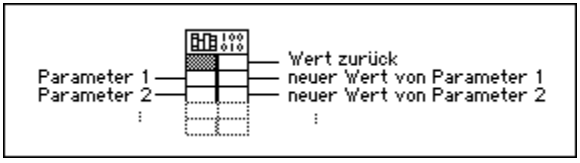
Weitere Informationen über Code Interface Nodes finden Sie nur in dem im Portable Document Format (PDF = übertragbares Dokumentenformat) verfügbaren *LabVIEW Code Interface Reference Manual*.

Aufruf ext. Bibliotheken

Ruft Standardbibliotheken auf, ohne daß ein Code Interface Node (CIN) geschrieben wird. Unter Windows kann eine DLL-Funktion (Dynamic Link Library = dynamische Verknüpfungsbibliothek) direkt aufgerufen werden. Auf Macintosh- und UNIX-Computern kann eine gemeinsam benutzte Bibliotheken-Funktion direkt aufgerufen werden. Auf Macintosh 68 K muß eine CFM-68K-Systemerweiterung installiert sein, damit der Knoten Aufruf ext. Bibliotheken arbeitet.

Dieser Knoten unterstützt eine große Anzahl von Datentypen und Aufrufkonventionen. Er kann zum Aufruf von Funktionen von den meisten Standard- und selbsterstellten Bibliotheken eingesetzt werden.

Der Knoten Aufruf ext. Bibliotheken ähnelt, wie in der folgenden Abbildung gezeigt, dem Knoten Code Interface Node.



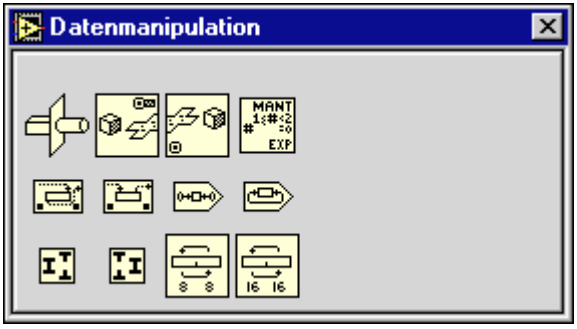
Die Funktion Aufruf ext. Bibliotheken setzt sich aus gepaarten Eingangs-/Ausgangsanschlüssen zusammen; die Eingänge befinden sich auf der linken und die Ausgänge auf der rechten Seite. Sie können eine oder beide verwenden. Der Rückgabewert dieser Funktion wird am rechten Anschluß des obersten Anschlußpaars des Knotens ausgegeben. Wenn es keinen Rückgabewert gibt, wurde dieses Anschlußpaar nicht verwendet. Jedes weitere Anschlußpaar entspricht einem Parameter der Funktionsparameterliste. Ein Wert wird durch Verbinden mit dem linken Anschluß eines Anschlußpaares an die Funktion weitergeleitet. Der Wert eines Parameters kann nach Funktionsaufruf durch Verbinden des rechten Anschlusses eines Anschlußpaares abgelesen werden.

Wenn Sie über das Popup-Menü des Knotens **Konfigurieren...** aufrufen, erhalten Sie ein Dialogfenster, über das der Bibliotheksname oder -pfad, der Funktionsname, Aufrufkonventionen, Parameter und der Rückgabewert für den Knoten festgelegt werden können. Durch Drücken von **OK** vergrößert sich der Knoten automatisch auf die richtige Anzahl von Anschlüssen. Er stellt dann die Anschlüsse auf den richtigen Datentyp ein. Für

weitere Informationen über die Funktion Aufruf ext. Bibliotheken lesen Sie bitte im *Referenzhandbuch zur Programmierung in G* das Kapitel 25, *Aufrufen von Code aus anderen Sprachen*.

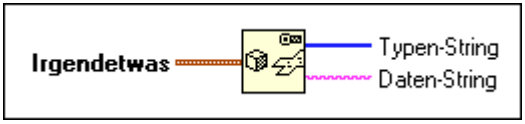
Beschreibungen der Funktionen zu Datenmanipulation

Die folgende Abbildung zeigt die auf der **Datenmanipulation**-Unterpalette verfügbaren Optionen.



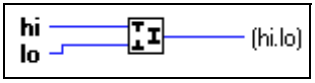
In Binärstring wandeln

Wandelt **Irgendetwas** in einen String aus Binärwerten um. **Typen-String** ist ein Typendeskriptor zur Beschreibung des Datentyps Irgendetwas. **Daten-String** ist eine umgewandelte Form von Irgendetwas. Für weitere Informationen über den Typendeskriptor und umgewandelte Daten lesen Sie bitte im *Referenzhandbuch zur Programmierung in G* den Anhang A, *Datenspeicherformate*.



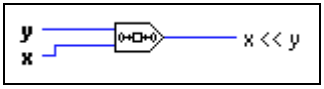
Zahlen vereinigen

Erzeugt aus den Byte- oder Wort-Komponenten eine Zahl.



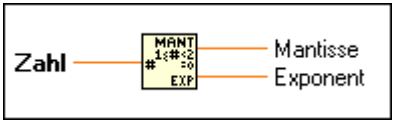
Logisches Schieben

Verschiebt **x** um die durch **y** vorgegebene Bitanzahl.



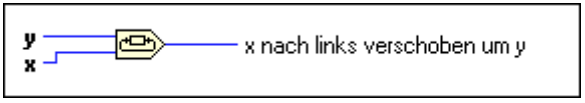
Mantisse & Exponent

Gibt die Mantisse und den Exponenten des numerischen Eingabewerts in der folgenden Form aus: **Zahl** = **Mantisse** * 2 **Exponent**. Wenn **Zahl** 0 ist, sind sowohl **Mantisse** und **Exponent** 0. Ansonsten ist der Wert von **Mantisse** größer als oder gleich 1 und kleiner als 2, und der Wert von **Exponent** ist ein Integer-Wert.



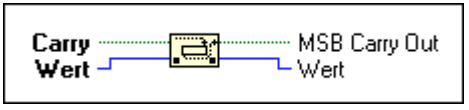
Drehen

Dreht **x** um die mit **y** vorgegebene Bitanzahl.



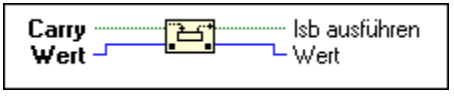
Nach Links rotieren (mit Carry)

Dreht in Eingabe-**Wert** jedes Bit nach links (vom niedrigstwertigen zum höchstwertigen Bit), fügt **Carry** in das niederwertige Bit ein und gibt das höchstwertige Bit aus.



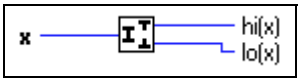
Nach Rechts rotieren (mit Carry)

Dreht in Eingabe-**Wert** jedes Bit nach rechts (vom höchstwertigen zum niedrigstwertigen Bit), fügt **Carry** in das hochwertige Bit ein und gibt das niedrigstwertige Bit aus.

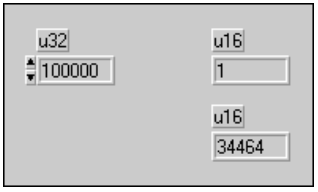


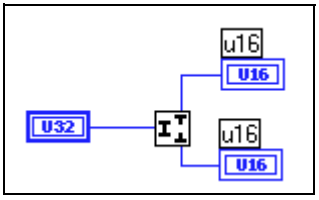
Zahl teilen

Teilt eine Zahl in ihre Byte- oder Wort-Komponenten auf.



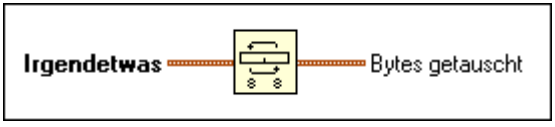
Die folgende Abbildung zeigt ein Beispiel, wie die Funktion Zahl teilen eingesetzt wird. Die Funktion teilt die vorzeichenbehaftete 32-Bit-Zahl 100.000 in die High-Word-Komponente, 1, und die Low-Word-Komponente 34.464 auf.





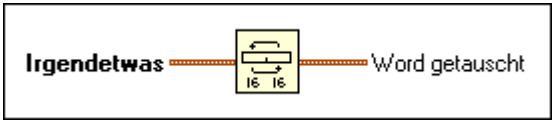
Bytes tauschen

Tauscht in jedem Wort in **Irgendetwas** die hochwertigen 8-Bits und niederwertigen 8-Bits.



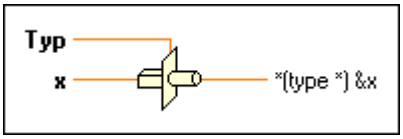
Worte tauschen

Tauscht in jedem Long-Integer-Wert in **Irgendetwas** die hochwertigen 16-Bits und niederwertigen 16-Bits.



Typenformung

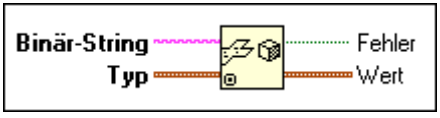
Formt **x** in den Datentyp **Typ**.



Formung von Daten in einen String konvertiert sie in eine plattformunabhängige, big endian Form, d.h., daß die Funktion das höchstwertige Byte oder Wort an die erste Stelle und das niedrigstwertige Byte oder Wort an die letzte Stelle setzt, die Ausrichtung entfernt und Extended-Zahlen in 16-Byte-Zahlen umwandelt. Formung eines Strings in ein 1D-Array konvertiert den String aus einer plattformunabhängigen Form in eine für diese Plattform systemspezifische Form.

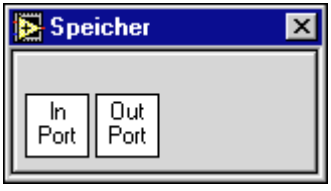
Von Binärstring wandeln

Konvertiert **Binärstring** zu dem mit **Typ** verbundenen Typ. Diese Funktion führt die Umkehrung der Funktion In Binärstring umwandeln aus. **Binärstring** sollte umgewandelte Daten von dem mit **Typ** verbundenen Typ enthalten. Für weitere Informationen lesen Sie bitte im *Referenzhandbuch zur Programmierung in G* den Abschnitt *Geglättete Daten* im Anhang A, *Datenspeicherformate*.



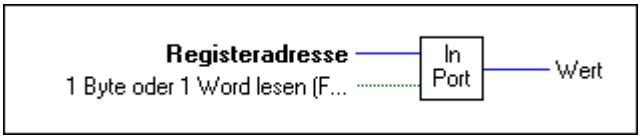
Beschreibungen der Speicher-VIs

Die folgende Abbildung zeigt die auf der **Speicher**-Unterpalette verfügbaren Optionen.



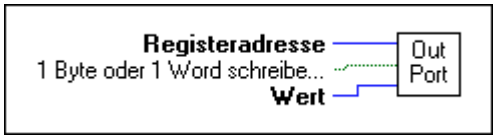
Eingangsport (Windows 3.1 und Windows 95)

Liest einen Byte- oder Wort-Integer von einer bestimmten **Registeradresse**. Da dieses VI nicht auf allen Plattformen verfügbar ist, sind VIs, die dieses SubVI einsetzen, nicht übertragbar.



Ausgangsport (Windows 3.1 und Windows 95)

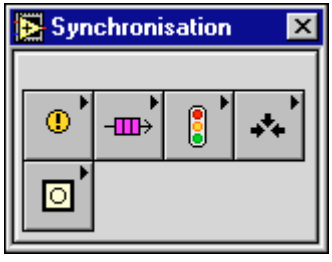
Schreibt einen Byte- oder Wort-Integer an eine bestimmte **Registeradresse**. Da dieses VI nicht auf allen Plattformen verfügbar ist, sind VIs, die dieses SubVI einsetzen, nicht übertragbar.



Synchronisations-VIs

Mit den Synchronisations-VIs können parallel ablaufende Aufträge synchronisiert werden. Daneben können die Synchronisations-VIs dazu verwendet werden, Daten zwischen parallel ablaufenden Aufträgen (Tasks) zu transportieren. Sie können über das Menü **Funktionen»Fortgeschritten»Synchronisation** auf die **Synchronisation**s-Palette zugreifen.

Die folgende Abbildung zeigt die auf der **Synchronisations**-Palette verfügbaren Optionen.



Die **Synchronisation**s-Palette setzt sich aus fünf Unterpaletten zusammen:

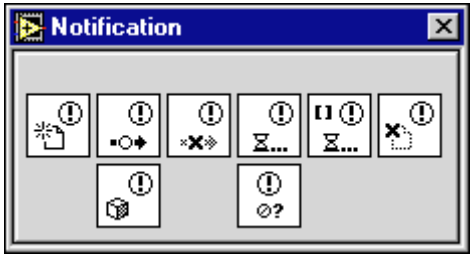
- Meldungs-VIs
- Queue-VIs
- Rendezvous-VIs
- Semaphor-VIs
- Occurrence-Funktionen

Meldungs-VIs

Mit den Meldungs-VIs können Daten von einem Auftrag zu einem anderen oder mehreren getrennten, parallelen Aufträgen geleitet werden. Insbesondere werden diese VIs eingesetzt, wenn ein oder mehrere VIs oder Teile von Blockdiagrammen warten sollen, bis ein anderes VI oder ein anderer Teil eines Blockdiagramms ihnen gewisse Daten sendet.

Die Meldungs-VIs unterscheiden sich von den Queue-VIs dadurch, daß die Datenübertragung nicht gepuffert erfolgt. Das heißt, wenn eine Meldung in dem Moment ihrer Übertragung nicht erwartet wird, gehen die Daten “verloren”, wenn eine andere Meldung gesendet wird. Außerdem können mehr als ein VI Auf Meldung warten dieselben Daten empfangen.

Auf die Meldungs-VIs kann über das Menü **Funktionen»Fortgeschritten» Synchronisation»Notification** zugegriffen werden.

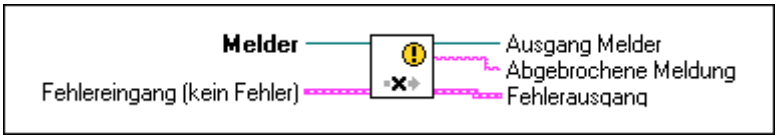


Die Meldungs-VIs verwenden das Bedienelement **Melder-Refnum** aus der Palette **Bedienelemente»Pfad & Refnum**.

Die Melder-Refnum können mit den folgenden VIs eingesetzt werden.

Meldung abbrechen

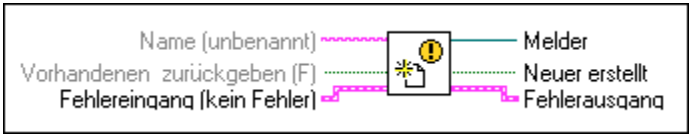
Mit diesem VI kann eine früher gesendete Meldung abgebrochen und zurückgegeben werden.



Dieses VI verhindert einen Aufruf zum VI Auf Meldung warten, während **Vorhergehendes ignorieren** auf FALSE gesetzt ist, damit die vorangegangene Meldung eingesehen werden kann.

Melder erstellen

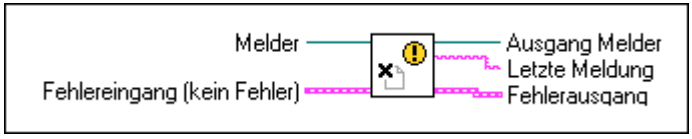
Sucht einen vorhandenen **Melder** auf oder erstellt einen neuen **Melder** und gibt eine Refnum aus, mit der andere Meldungs-VIs aufgerufen werden können.



Wenn **Name** angegeben ist, sucht das VI zuerst nach einem vorhandenen **Melder** unter demselben Namen und gibt dessen Refnum aus, wenn er existiert. Wenn ein benannter Melder unter diesem Namen noch nicht vorhanden ist und der Eingang **Vorhandenen zurückgeben** auf FALSE steht, erstellt das VI einen neuen **Melder** und gibt dessen Refnum aus. Der Ausgang **Neuer erstellt** gibt TRUE aus, wenn das VI einen neuen Melder erstellt.

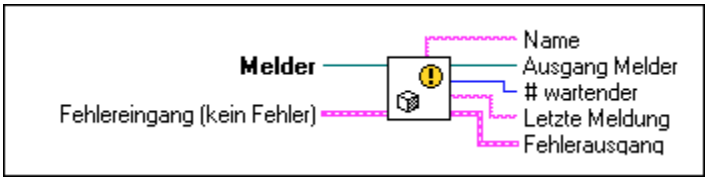
Melder zerstören

Löst den angegebenen **Melder** auf und gibt die **Letzte Meldung**, die gesendet wurde, aus. Alle VIs Auf Meldung warten, die gegenwärtig auf diesem Melder warten, schalten sofort ab und geben einen Fehler aus.



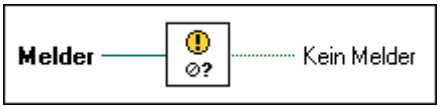
Melderstatus bekommen

Gibt die aktuelle Statusinformation von **Melder** aus.



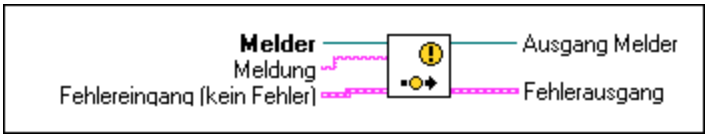
Kein Melder

Gibt TRUE aus, wenn **Melder** keine gültige Melder-Refnum ist.



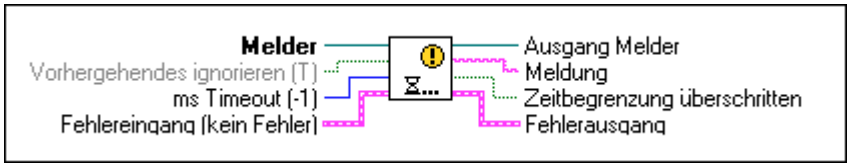
Meldung senden

Sendet **Meldung** an den angegebenen **Melder**. Alle VIs Auf Meldung warten, die gegenwärtig auf diesen **Melder** warten, beenden das Warten und geben die vorgegebene **Meldung** aus.



Auf Meldung warten

Wartet darauf, daß das VI Meldung senden an den angegebenen Melder **Meldung** sendet.

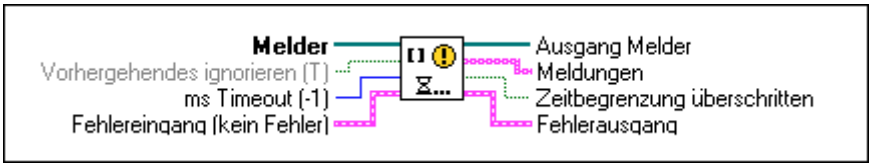


Wenn **Vorhergehendes ignorieren** FALSE ist und eine Meldung seit dem letzten Aufruf dieses VIs gesendet wurde, gibt das VI sofort den Wert der alten Meldung und **Zeitbegrenzung überschritten** als FALSE aus. Wenn der Eingang **Vorhergehendes ignorieren** TRUE ist, wartet das VI eine Wartedauer in Millisekunden (Standardeinstellung –1 oder ständig), ehe die Zeit abgelaufen ist und es abschaltet. Wenn eine Meldung gesendet wird, gibt **Zeitbegrenzung überschritten** FALSE aus. Wenn keine

Meldung gesendet wurde oder **Melder** nicht gültigist, gibt **Zeitbegrenzung überschritten** TRUE aus.

Auf Meldung von mehreren warten

Wartet darauf, daß das VI Meldung senden eine Meldung zu einem der angegebenen Melder sendet.



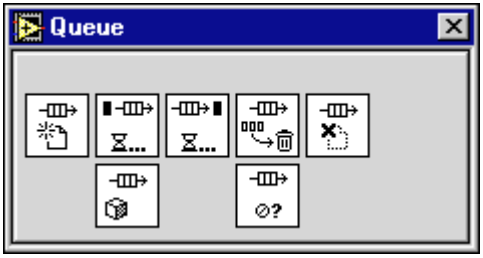
Wenn **Vorhergehendes ignorieren** auf FALSE steht und eine Meldung seit dem letzten Aufruf dieses VIs an einen der angegebenen Melder gesendet wurde, gibt das VI sofort den/die Wert(e) der letzten Meldung(en) und **Zeitbegrenzung überschritten** als FALSE aus. Wenn der Eingang **Vorhergehendes ignorieren** TRUE ist, wartet das VI die Anzahl von **ms timeout** Millisekunden (Standardeinstellung –1 oder ständig) vor dem Auslaufen der Zeit. Wenn wenigstens eine Meldung gesendet wurde, gibt **Zeitbegrenzung überschritten** FALSE aus. Wenn keine Meldung gesendet wurde, gibt **Zeitbegrenzung überschritten** TRUE aus.

Queue-VIs

Mit den Queue-VIs können geordnete Sequenzen von Datenelementen von einem Auftrag an einen anderen, getrennten, parallelen Auftrag geleitet werden. Diese VIs werden insbesondere eingesetzt, wenn ein Auftrag warten soll, bis ein anderer Auftrag ihn mit bestimmten Daten versorgt. Daneben werden diese VIs auch eingesetzt, wenn ein Auftrag solange warten soll, bis ein anderer Auftrag die vom ersten Auftrag gelieferten Daten verarbeitet hat.

Die Queue-VIs unterscheiden sich von den Meldungs-VIs dadurch, daß die Datenübertragung gepuffert erfolgt. Das heißt, wenn in dem Moment des Einreihens eines Elements in ein Queue nicht auf Lesen gewartet wird, verbleibt das Element solange im Queue, bis es ausdrücklich entfernt wird. Außerdem gibt es den Unterschied, daß nur ein VI Daten erhalten kann, wenn Daten in das Queue eingereiht werden und zwei VIs darauf warten, sie aus dem Queue zu entnehmen.

Auf die Queue-VIs kann über das Menü **Funktionen»Fortgeschritten»Synchronisation» Queue** zugegriffen werden.

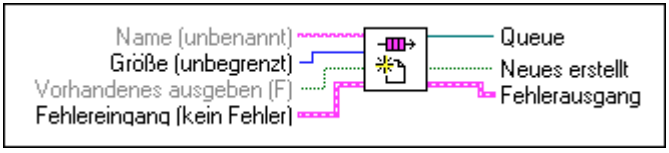


Die Queue-VIs verwenden das Bedienelement **Queue-RefNum** von der Palette **Bedienelement»Pfad & Refnum**.

Queue-RefNum kann mit den folgenden VIs verwendet werden.

Queue erstellen

Sucht ein vorhandenes Queue auf oder erstellt ein neues Queue und gibt eine Refnum aus, die beim Aufruf anderer Queue-VIs verwendet werden kann.

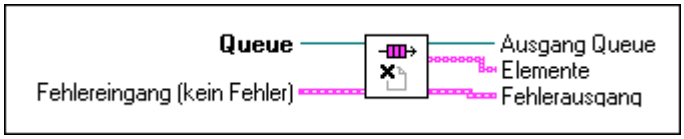


Wenn Sie die Größe mit > 0 festlegen, begrenzen Sie die Queue-Größe auf diese Anzahl von Elementen. Wenn das VI Queue-Element einfügen versucht, ein Element in ein volles Queue einzufügen, muß es warten, bis mit dem VI Queue-Element entfernen ein Element entfernt wurde. Die Standardeinstellung für ein unbegrenztes Queue ist -1.

Wenn ein Name angegeben wurde, sucht das VI zunächst nach einem vorhandenen Queue unter demselben Namen und gibt dessen Refnum aus, wenn es existiert. Wenn ein benanntes Queue unter demselben Namen noch nicht existiert und der Eingang **Vorhandenes ausgeben** FALSE ist, erstellt das VI ein neues Queue und gibt dessen Refnum aus. Der Ausgang **Neues erstellt** gibt TRUE aus, wenn das VI ein neues Queue erstellt.

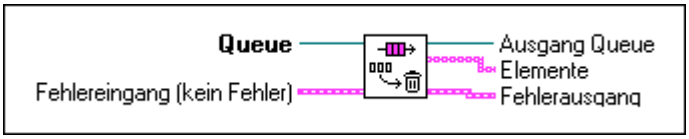
Queue zerstören

Löst das angegebene Queue auf und gibt jegliche, in dem Queue befindliche Elemente aus. Alle VIs Queue-Element einfügen und Queue-Element entfernen, die gegenwärtig an diesem Queue warten, laufen sofort ab und geben einen Fehler aus.



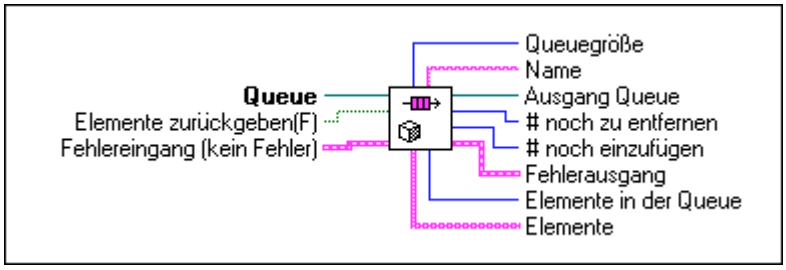
Queue leeren

Entfernt alle **Elemente** aus dem **Queue**.



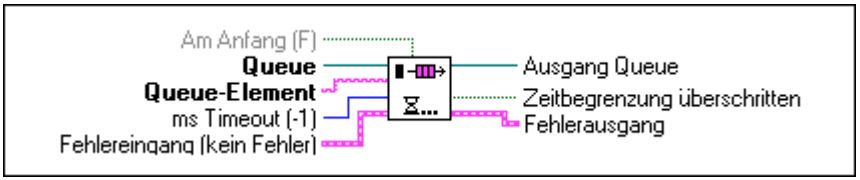
Queue-Status bekommen

Gibt die aktuellen Statusinformationen von **Queue** aus.



Queue-Element einfügen

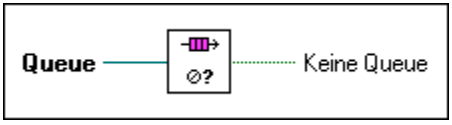
Fügt ein Element in ein Queue ein.



Der Parameter **Am Anfang** legt fest, ob das Element am Ende (Standardeinstellung) oder am Anfang des Queues eingefügt wird. Wenn das Queue voll ist, wartet das VI die Anzahl von **Timeout** Millisekunden (Standardeinstellung -1 oder ständig), ehe es ausläuft. Wenn während des Wartens Platz frei wird, wird das Element eingefügt und **Timeout** gibt FALSE aus. Verbleibt das Queue voll oder ist das Queue ungültig, gibt **Timeout** TRUE aus.

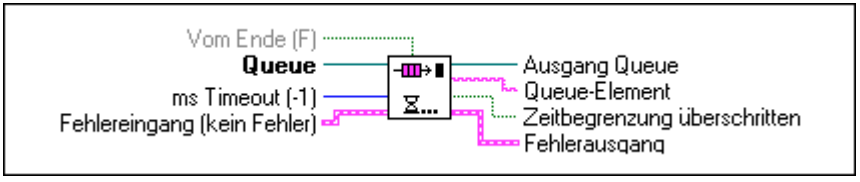
Kein Queue

Gibt TRUE aus, wenn **Queue** keine gültige Queue-Refnum ist.



Queue-Element entfernen

Entfernt ein Element von einem Queue.

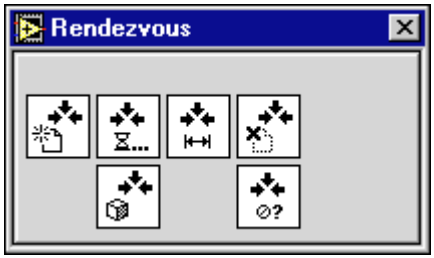


Der Parameter **Vom Ende** legt fest, ob das ausgegebene Element vom Anfang (Standardeinstellung) oder vom Ende des Queues genommen wird. Wenn Queue leer ist, wartet das VI die Anzahl von **Timeout** Millisekunden (Standardeinstellung -1 oder ständig), ehe es ausläuft. Wenn ein Element während des Wartens verfügbar wird, wird es ausgegeben und **Zeitbegrenzung überschritten** gibt FALSE aus. Wenn kein Element verfügbar wird oder das Queue nicht gültig ist, gibt **Zeitbegrenzung überschritten** TRUE aus.

Rendezvous-VIs

Mit den Rendezvous-VIs können zwei oder mehr getrennte, parallele Aufträge an bestimmten Punkten in der Ausführung synchronisiert werden. Jeder Auftrag, der den Rendezvous-Treffpunkt erreicht, wartet, bis die vorgegebene Anzahl von Aufträgen wartet; daraufhin setzen dann alle Aufträge die Ausführung fort.

Auf die Rendezvous-VIs kann über das Menü **Funktionen»Fortgeschritten»Synchronisation»Rendezvous** zugegriffen werden.

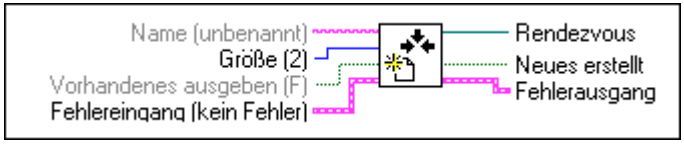


Die Rendezvous-VIs verwenden das Bedienelement **Rendezvous-RefNum** von der Palette **Bedienelement»Pfad & Refnum**.

Die Rendezvous-Refnum kann mit den folgenden VIs verwendet werden.

Rendezvous erstellen

Sucht ein vorhandenes **Rendezvous** oder erstellt ein neues **Rendezvous** und gibt eine Refnum aus, die beim Aufruf anderer Rendezvous-VIs verwendet werden kann.



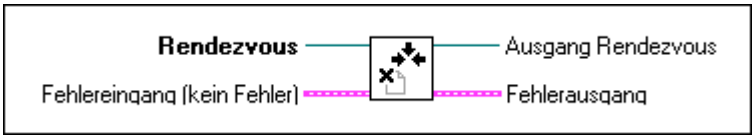
Größe legt fest, wie viele Aufträge sich zum **Rendezvous** treffen müssen, damit die Ausführung fortgesetzt wird. Die Standardeinstellung beträgt 2.

Wenn **Name** festgelegt wird, sucht das VI zunächst nach einem vorhandenen Rendezvous und gibt dessen Refnum aus, wenn es existiert. Wenn kein Rendezvous unter diesem Namen vorhanden ist und der Eingang **Vorhandenes ausgeben** FALSE ist, erstellt das VI ein neues

Rendezvous und gibt dessen Refnum aus. Der Ausgang **Neues erstellt** gibt TRUE aus, wenn das VI ein neues Rendezvous erstellt.

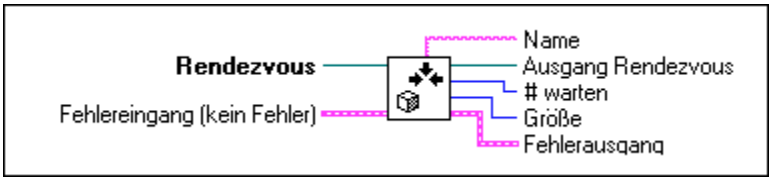
Rendezvous zerstören

Löst das angegebene Rendezvous auf. Alle VIs Am Rendezvous warten, die gegenwärtig an diesem Rendezvous warten, laufen sofort aus und geben einen Fehler aus.



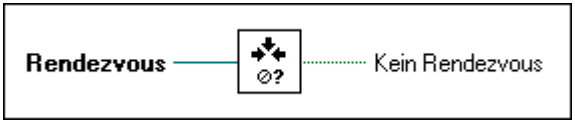
Rendezvous-Status bekommen

Gibt die aktuellen Statusinformationen von einem **Rendezvous** aus.



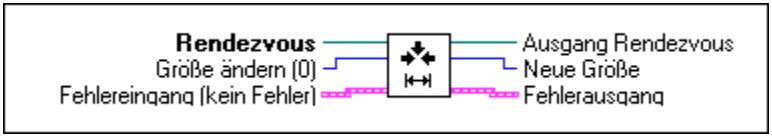
Kein Rendezvous

Gibt TRUE aus, wenn **Rendezvous** keine gültige Rendezvous-Refnum ist.



Rendezvous-Größe ändern

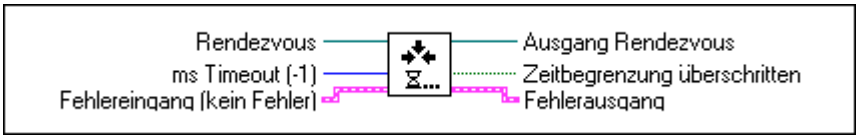
Verändert mit Hilfe von **Größe ändern** die Größe von **Rendezvous** und gibt **Neue Größe** aus.



Wenn die Anzahl der gegenwärtig am **Rendezvous** wartenden Aufträge kleiner als oder gleich **Neue Größe** ist, hören die Aufträge der ersten Größe zu warten auf und setzen die Ausführung fort.

Auf Rendezvous warten

Wartet, bis eine ausreichende Anzahl von Aufträgen an dem Rendezvous eingetroffen ist.



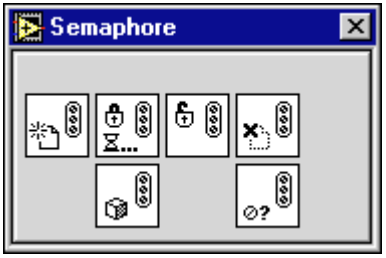
Wenn die Anzahl von Aufträgen, einschließlich des neuen, die am **Rendezvous** warten, kleiner als die Rendezvous-Größe ist, wartet das VI die Anzahl von **Timeout** Millisekunden (Standardeinstellung -1 oder ständig), ehe es ausläuft. Wenn während des Wartens genügend Aufträge am Rendezvous eintreffen, gibt **Zeitbegrenzung überschritten** FALSE aus. Wenn ungenügend Aufträge eintreffen oder das Rendezvous ungültig ist, gibt **Zeitbegrenzung überschritten** TRUE aus.

Semaphor-VIs

Semaphore, auch als Mutexe bekannt, werden zur Begrenzung der Anzahl von Aufträgen, die gleichzeitig auf einer gemeinsam benutzten (geschützten) Ressource operieren, eingesetzt. Eine geschützte Ressource oder ein kritischer Abschnitt kann u.U. das Schreiben zu globalen Variablen oder die Kommunikation mit externen Instrumenten beinhalten.

Semaphor-VIs können zur Synchronisation von zwei oder mehreren getrennten, parallelen Aufträgen eingesetzt werden, so daß zu einem Zeitpunkt jeweils nur ein Auftrag einen kritischen Code-Abschnitt ausführt, der von einem gemeinsamen Semaphor geschützt ist. Diese VIs werden insbesondere dann eingesetzt, wenn andere VIs oder Teile des Blockdiagramms warten sollen, bis ein anderes VI oder ein anderer Teil des Blockdiagramms die Ausführung des kritischen Abschnitts beendet hat.

Auf die Semaphor-VIs kann über das Menü **Funktionen»Fortschgeschrittene» Synchronisation»Semaphor** zugegriffen werden.

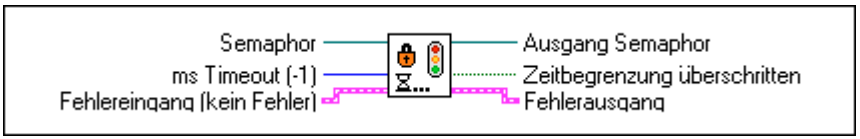


Die Semaphor-VIs verwenden das Bedienelement **Semaphor-Refnum** von der Palette **Bedienelemente»Pfad & Refnum.**

Die Semaphor-Refnum kann mit den folgenden VIs verwendet werden.

Semaphor erfassen

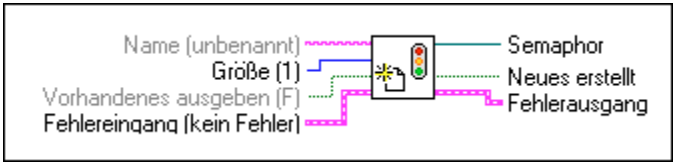
Erwirbt Zugriff zu einem Semaphor.



Wenn der Semaphor bereits von der maximalen Auftragsanzahl beansprucht wird, wartet das VI die Anzahl von **Timeout** Millisekunden (Standardeinstellung -1 oder ständig), ehe es ausläuft. Wenn der Semaphor in dieser Zeit verfügbar wird, gibt **Zeitbegrenzung überschritten** FALSE aus. Wird der Semaphor jedoch in dieser Zeit nicht verfügbar oder der Semaphor ist ungültig, gibt **Zeitbegrenzung überschritten** TRUE aus.

Semaphor erstellen

Sucht einen vorhandenen Semaphor auf oder erstellt einen neuen Semaphor und gibt eine Refnum aus, die beim Aufruf von anderen Semaphor-VIs verwendet werden kann.

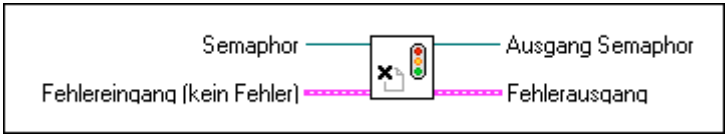


Größe legt fest, wie viele Aufträge den Semaphor gleichzeitig beanspruchen dürfen. Die Standardgröße ist 1.

Wenn ein Name angegeben wird, sucht das VI unter diesem Namen zunächst nach einem vorhandenen Semaphor und gibt dessen Refnum aus, wenn es existiert. Wenn ein benannter Semaphor unter diesem Namen noch nicht vorhanden ist und der Eingang **Vorhandenes ausgeben** FALSE ist, erstellt das VI einen neuen Semaphor und gibt dessen Refnum aus. Der Ausgang **Neues erstellt** gibt TRUE aus, wenn das VI einen neuen Semaphor erstellt.

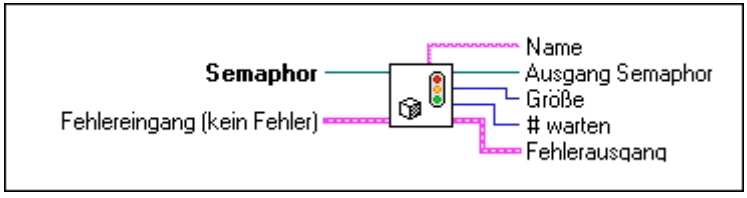
Semaphor zerstören

Löst den angegebenen Semaphor auf. Alle VIs Semaphor beanspruchen, die gegenwärtig an diesem Semaphor warten, laufen sofort aus und geben einen Fehler aus.



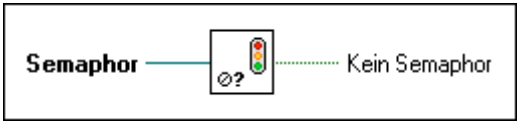
Semaphor-Status bekommen

Gibt die aktuellen Statusinformationen eines Semaphors aus.



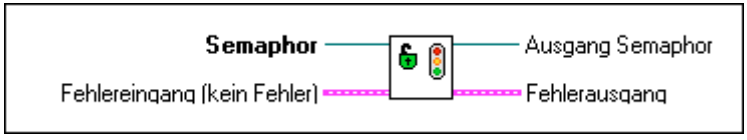
Kein Semaphor

Gibt TRUE aus, wenn **Semaphor** keine gültige Semaphor-Refnum ist.



Semaphor freigeben

Gibt Zugriff auf einen Semaphor frei.



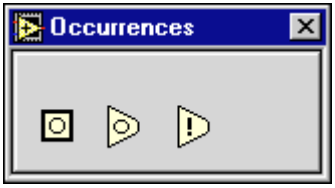
Wenn ein VI Semaphor beanspruchen auf diesen Semaphor wartet, hört es zu warten auf und setzt die Ausführung fort. Wenn Sie das VI Semaphor freigeben eines Semaphors, der nicht beansprucht wurde, aufrufen, setzen Sie die Semaphor-Größe im Grunde um ein Inkrement herauf.

Beschreibungen der Occurrence-Funktionen

Mit den Occurrence-Funktionen können getrennte, synchrone Aktivitäten gesteuert werden. Diese Funktionen können insbesondere dann eingesetzt werden, wenn ein VI oder Teil eines Blockdiagramms warten soll, bis ein anderes VI oder ein anderer Teil eines Blockdiagramms einen Auftrag beendet, ohne daß LabVIEW dabei zur Abfrage gezwungen wird.

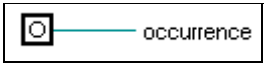
Derselbe Auftrag kann mit globalen Variablen ausgeführt werden, indem eine Schleife den Wert der globalen Variable abfragt, bis sich deren Wert verändert. Globale Variablen verursachen jedoch Verarbeitungsaufwand, weil die die globale Variable abfragende Schleife Ausführungszeit beansprucht. Bei Occurrences ist die Abfrageschleife mit einer Funktion Wartet auf Occurrence ersetzt, und somit wird keine Prozessorzeit benötigt. Wenn irgendein Diagramm die Occurrence setzt, aktiviert LabVIEW in jedem Blockdiagramm alle Wartet-auf-Occurrence-Funktionen, die auf die angegebene Occurrence warten.

Die folgende Abbildung zeigt die auf der **Occurrence**-Unterpalette verfügbaren Optionen.



Occurrence erzeugen

Erzeugt eine **Occurrence**, die an die Funktionen Wartet auf Occurrence und Occurrence setzen weitergegeben werden kann.



Im allgemeinen ist nur ein Occurrence-erzeugen-Knoten mit einem Satz Wartet-auf-Occurrence- und Occurrence-setzen-Funktionen verbunden. Eine Occurrence-setzen-Funktion kann mit einer beliebigen Anzahl Wartet-auf-Occurrence- und Occurrence-setzen-Funktionen verbunden werden. Es ist keine gleiche Anzahl von Wartet-auf-Occurrence- und Occurrence-setzen-Funktionen notwendig.

Anders als andere Synchronisations-VIs stellt jede Occurrence-erzeugen-Funktion auf einem Blockdiagramm eine einzelne, einzigartige Occurrence dar. So gesehen ist eine Occurrence-erzeugen-Funktion eine Konstante. Wenn jedes Mal, wenn eine Occurrence-erzeugen-Funktion abläuft, ein VI arbeitet, produziert der Knoten denselben Wert. Wenn Sie z.B. eine Occurrence-erzeugen-Funktion in eine Schleife plazieren, ist der von der Occurrence-erzeugen-Funktion erzeugte Wert für jede Iteration der Schleife derselbe. Wenn Sie eine Occurrence-erzeugen-Funktion auf dem Blockdiagramm eines ablaufinvarianten VIs

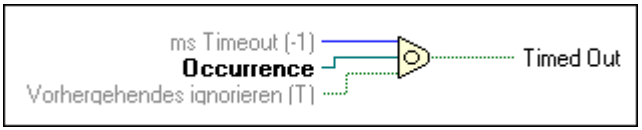
plazieren, erzeugt die Funktion Occurrence-erzeugen für jeden Aufrufer einen unterschiedlichen Wert.

Occurrence setzen

Triggert die vorgegebene **Occurrence**. Alle auf diese Occurrence wartenden Blockdiagramme hören mit dem Warten auf.

Auf Occurrence warten

Wartet darauf, daß die Funktion Occurrence erzeugen eine bestimmte **Occurrence** setzt oder triggert.



Teil II

Datenerfassungs-VIs

Teil II, *Datenerfassungs-VIs*, stellt die vorhandenen VIs vor, die mit Ihren Datenerfassungs-Hardware-Geräten (DAQ) arbeiten. In diesem Teil sind folgende Kapitel enthalten:

- Kapitel 14, *Einführung zu den LabVIEW Datenerfassungs-VIs*, enthält grundlegende Informationen über die Datenerfassungs-VIs (DAQ) und zeigt Ihnen, wo Sie diese in LabVIEW finden können.
- Kapitel 15, *Einfache Analogeingangs-VIs*, beschreibt die Einfachen Analogeingangs-VIs, die einfache Analogeingangs-Funktionen ausführen.
- Kapitel 16, *Mittlere Analogeingangs-VIs*, beschreibt die Mittleren Analogeingangs-VIs.
- Kapitel 17, *Analogeingangs-Utility-VIs*, beschreibt die Analogeingangs-Utility-VIs. Diese VIs - AI einen Scan lesen, AI Signalverlauf abtasten und AI kontinuierlich abtasten sind Einzel-VI-Lösungen für häufig vorkommende Analogeingangs-Probleme. Die Analogeingangs-Utility-VIs sind Mittel-Level-VIs, die sich auf die VIs von Fortgeschrittenem Level stützen.
- Kapitel 18, *Fortgeschrittene Analogeingangs-VIs*, enthält Referenzbeschreibungen der Fortgeschrittenen Analogeingangs-VIs. Diese VIs stellen die Schnittstelle zur NI-DAQ-Software dar und bilden die Grundlage der Einfachen,- Utility- und Mittleren Analogeingangs-VIs.
- Kapitel 19, *Einfache Analogausgangs-VIs*, beschreibt die Einfachen Analogausgangs-VIs in LabVIEW, die einfache Analogausgangs-Operationen durchführen.
- Kapitel 20, *Mittlere Analogausgangs-VIs*, beschreibt die Mittlere Analogausgangs-VIs. Diese VIs - AO ein Update schreiben, AO Signalverlauf erzeugen und AO kontinuierlich erzeugen - sind Einzel-VI-Lösungen für häufig vorkommende Analogausgangsprobleme.

- Kapitel 21, *Analogausgang-Utility-VIs*, beschreibt die Analogausgangs-Utility-VIs. Die VIs - AO kontinuierlich erzeugen, AO Signalverlauf erzeugen und AO ein Update schreiben - sind Einzel-VI-Lösungen für häufig vorkommende Analogausgangsprobleme. Die Analogausgangs-Utility-VIs sind Mittel-Level-VIs.
- Kapitel 22, *Fortgeschrittene Analogausgangs-VIs*, enthält Referenzbeschreibungen der Fortgeschrittenen Analogausgangs-VIs. Diese VIs stellen die Schnittfläche zur NI-DAQ-Software dar und bilden die Grundlage für die Einfachen,- Utility- und Mittleren Analogausgangs-VIs.
- Kapitel 23, *Einfache Digital-I/O-VIs*, beschreibt die Einfachen Digital-I/O-VIs, die einfache Digital-I/O-Operationen durchführen.
- Kapitel 24, *Mittlere Digital-I/O-VIs*, beschreibt die Mittlere Digital-I/O-VIs. Diese VIs sind Einzel-VI-Lösungen für häufig vorkommende digitale Probleme.
- Kapitel 25, *Fortgeschrittene Digital-I/O-VIs*, beschreibt die Fortgeschrittenen Digital-I/O-VIs, zu denen die Digitalanschluß- und Digitalgruppen-VIs gehören. Die Digitalanschluß-VIs werden für ein sofortiges Lesen und Schreiben mit Digitalleitungen und -anschlüssen verwendet. Die Digitalgruppen-VIs werden für sofortige, getaktete I/Os oder für solche im Handshake-Betrieb mit mehreren Anschlüssen verwendet. Diese VIs stellen die Schnittfläche zur NI-DAQ-Software dar und bilden die Grundlage für die Einfachen und Mittleren Digital-I/O-VIs.
- Kapitel 26, *Einfache-Counter-VIs*, beschreibt die Einfachen Counter-VIs, die einfache Zähloperationen durchführen.
- Kapitel 27, *Mittlere Counter-VIs*, beschreibt Mittlere Counter-VIs, die Sie zur Programmierung an MIO-, TIO- und anderen Geräten mit DAQ-STC- oder Am9513-Counter-Chips verwenden können. Diese VIs rufen die Fortgeschrittenen Counter-VIs auf, um die Counter (Zähler) für übliche Operationen zu konfigurieren und um die Counter zu starten, abzulesen und zu stoppen. Sie können diese VIs so konfigurieren, daß diese Einzelimpulse und fortlaufende Impulsfolgen erzeugen, Events oder abgelaufene Zeit zählen, ein Signal herunterdividieren und die Impulsbreite oder -periode messen. Die Einfachen Counter-VIs rufen die Mittleren Counter-VIs für verschiedene Erzeugungs-, Zähl- und Meßoperationen auf.

- Kapitel 28, *Fortgeschrittene Counter-VIs*, beschreibt die VIs, die Hardware-Counter konfigurieren und steuern. Sie können diese VIs zur Erzeugung variabler Tastverhältnis-Rechteckschwingungen, zur Zählung von Events und zur Messung von Perioden und Frequenzen verwenden.
- Kapitel 29, *Kalibrierungs- und Konfigurations-VIs*, beschreibt die VIs, die spezifische Geräte kalibrieren und Konfigurationsinformationen einstellen und zurückgeben.
- Kapitel 30, *Signalkonditionierungs-VIs*, beschreibt die Datenerfassungs-Signalkonditionierungs-VIs, die Sie benutzen, um Analogeingangs-Spannungsablesungen von RTD (Resistance Temperature Detector; Widerstandstemperaturfühler), Ordnungsmeßstreifen oder Thermoelemente in Dehnungs- oder Temperatureinheiten umzuwandeln.

14

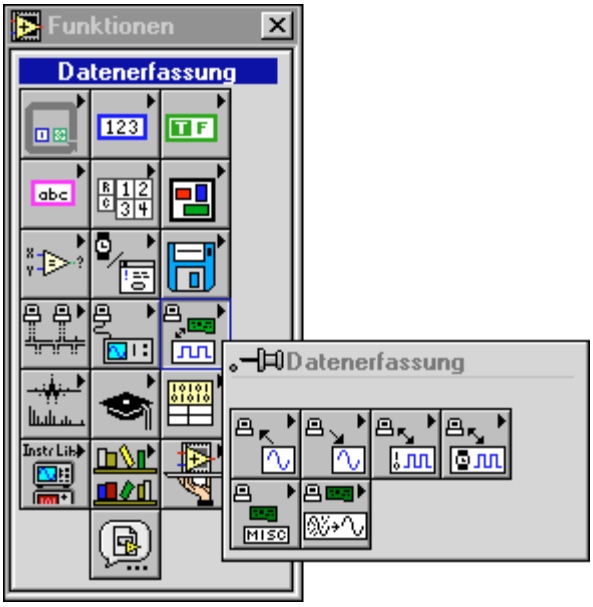
Einführung zu den LabVIEW Datenerfassungs-VIs

In diesem Kapitel sind grundlegende Informationen über Datenerfassungs-VIs enthalten; es werden außerdem die jeweiligen Fundstellen in LabVIEW angegeben. Die Beschreibungen dieser VIs umfassen Kapitel 14 bis 29.

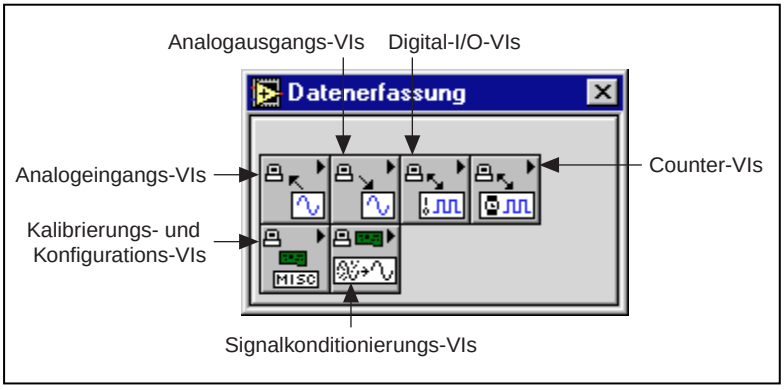
LabVIEW beinhaltet eine Anzahl von VIs, die mit Ihren Datenerfassungs-Hardwaregeräten funktionieren. Mit den Datenerfassungs-VIs von LabVIEW können Sie Erfassungs- und Steueranwendungen entwickeln.

Sie können die Datenerfassungs-VIs in der Palette **Funktionen** im jeweiligen LabVIEW-Blockdiagramm finden. Die Datenerfassungs-VIs befinden sich im unteren Teil der Palette **Funktionen**.

Um auf die Palette **Datenerfassung** zugreifen zu können, wählen Sie, wie in nachstehender Abbildung dargestellt, **Funktionen**»**Datenerfassung** aus.



Die Palette **Datenerfassung** enthält sechs Unterpaletten-Icons, über die Sie zu den verschiedenen Klassen von Datenerfassungs-VIs gelangen. In der folgenden Abbildung wird die Bedeutung der einzelnen Icons in der Palette **Datenerfassung** gezeigt.



Dieser Teil des Handbuchs ist nach der Reihenfolge, in der die Datenerfassungs-VI-Icons in der Palette **Datenerfassung** von links nach rechts erscheinen, gegliedert. In diesem Abschnitt werden die Kapitel Analogeingangs-VI von den Kapiteln Analogausgangs-VI gefolgt, die dann wiederum von den Kapiteln Digital-I/O-VI gefolgt werden, usw. Meistens sind einer Klasse Datenerfassungs-VIs in der Palette mehrere

Kapitel gewidmet, da viele der VI-Kapitel auch mehrere Unterpaletten enthalten.

Online-Hilfe für die Datenerfassungs-VIs finden

Durch Benutzung des Hilfefensters (**Hilfe»Hilfe anzeigen**) von LabVIEW können Sie online wertvolle Informationen zu einzelnen VIs erhalten. Wenn Sie den Cursor auf ein VI-Icon plazieren, werden das Verbindungsdiagramm und die Parameterbezeichnungen für dieses VI im Hilfefenster angezeigt. Sie können auch Informationen für Bedien- oder Anzeigeelemente im Frontpanel finden, indem Sie den Cursor bei geöffnetem Hilfefenster über das Bedien- oder Anzeigeelement plazieren. Für weitere Information zum LabVIEW-Hilfefenster wird auf den Abschnitt *So erhalten Sie Hilfe* in Kapitel 1, *Einführung in die Programmierung in G* des *Referenzhandbuchs zur Programmierung in G* verwiesen.

Zusätzlich zum Hilfefenster sind in LabVIEW weitere, umfassendere Online-Informationen verfügbar. Wählen Sie **Hilfe»Online-Referenz** aus, um auf diese Informationen zuzugreifen. Für die meisten Blockdiagrammobjekte können Sie **Online-Referenz** vom Popup-Menü des Objekts auswählen, um auf die Online-Beschreibung zugreifen zu können. Sie können auf diese Information auch durch Drücken der links abgebildeten Taste zugreifen, die sich unten am Hilfefenster von LabVIEW befindet. Informationen zur Erstellung Ihrer eigenen Online-Referenzdateien finden Sie im Abschnitt *Erstellen Ihrer eigenen Hilfe-Dateien* in Kapitel 5, *Drucken und Dokumentieren von VIs* des *Referenzhandbuchs zur Programmierung in G*.

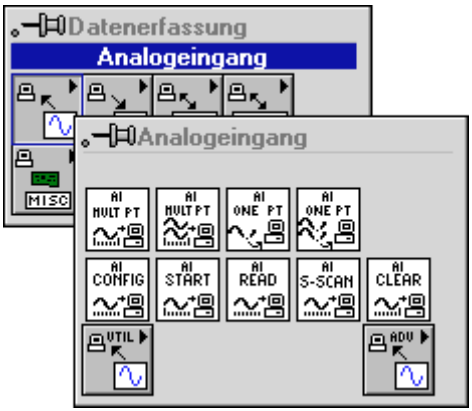
☞ **Hinweis** ***Benutzen Sie für jedes VI nur die benötigten Eingänge. LabVIEW setzt alle nicht verbundenen Eingänge auf die jeweiligen Standardwerte. Viele der Datenerfassungs-Funktionseingänge sind optional und werden im Fenster* Einfache Diagrammhilfe *nicht angezeigt. Diese Eingänge beziehen sich üblicherweise auf selten benutzte Optionen. Wenn eine Eingabe erforderlich ist, bleibt Ihre VI-Verbindung "unterbrochen", bis diese Eingabe mit einem Wert verbunden wird. Erforderliche Eingaben werden im Fenster* Hilfe *in Fettdruck angezeigt, empfohlene Eingaben in Normalschrift und optionale Eingaben in grauer Schrift. Die Standardwerte für Eingänge werden neben dem Eingangsnamen im Fenster* Hilfe *in Klammern angezeigt*.

☞ **Hinweis** ***Einige Datenerfassungs-VIs benutzen einen Enum-Datentyp als Bedien- oder Anzeigeterminal. Wenn Sie einen numerischen Wert mit einem Enum-Anzeigeelement verbinden, konvertiert LabVIEW die Zahl in das nächste Enum-Objekt. Wenn Sie ein Enum-Bedienelement mit einem Zahlenwert verbinden, ist der Wert der Enum-Index.***

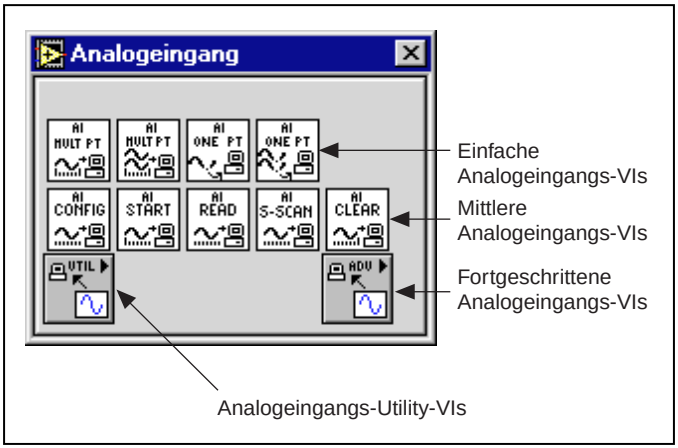
Die Analogeingangs-VIs

Divises führen Analogeingangsfunktionen aus.

Sie finden die Analogeingangs-VIs durch Auswahl von **Funktionen» Datenerfassung»Analogeingang**. Durch Klicken auf das Icon Analogeingabe in der Palette **Datenerfassung** wird, wie in nachstehender Abbildung dargestellt, die Palette **Analogeingang** angezeigt.



In der Palette **Analogeingang** sind vier Klassen von Analogeingangs-VIs vorhanden: die Einfachen Analogeingangs-VIs, die Mittleren Analogeingangs-VIs, die Analogeingangs-Utility-VIs und die Fortgeschrittenen Analogeingangs-VIs. Nachstehend werden diese VI-Klassen beschrieben.



Einfache Analogeingangs-VIs

Die Einfachen Analogeingangs-VIs führen einfache Analogeingangsfunktionen durch. Sie können diese VIs vom Frontpanel aus ausführen oder sie als SubVIs in Grundanwendungen benutzen.

Sie können jedes VI separat als analoge Grundoperation einsetzen. Im Gegensatz zu den VIs der Mittleren und der Fortgeschrittenen Stufe, werden Sie von den Einfachen Analogeingangs-VIs mittels Dialogfelder auf Fehler aufmerksam gemacht. Sie werden aufgefordert, die Ausführung zu stoppen oder den Fehler zu ignorieren.

Die Einfachen Analogeingangs-VIs sorgen für eine benutzerfreundliche Grundschnittfläche mit lediglich den am häufigsten benutzten Eingängen und Ausgängen. Es wird empfohlen, für komplexere Anwendungen die Mittleren Analogeingangs-VIs oder die Fortgeschrittenen Analog-Eingangs-VIs zu benutzen, um eine größere Funktionalität und Leistung zu erreichen.

In Kapitel 15, *Einfache Analogeingangs-VIs* finden Sie spezifische VI-Informationen.

Mittlere Analogeingangs-VIs

Sie können Mittlere Eingangs-VIs an zwei verschiedenen Stellen in der Palette **Analogeingang** finden. Sie können die Mittleren Analogeingangs-VIs in der zweiten Reihe der Palette **Analogeingang** finden. Die anderen Mittleren VIs befinden sich in der Palette **Analogeingangs-Utility**, die später besprochen wird. Die Mittleren Analogeingangs-VIs - AI konfigurieren, AI starten, AI lesen, AI einzelner Scan und AI zurücksetzen - bauen wiederum auf der fundamentalen Bausteinschicht, Fortgeschrittene Analogeingangs-VIs genannt, auf. Diese VIs bieten fast ebensoviel Leistungskraft wie die VIs fortgeschrittener Stufe; sie gruppieren die VIs fortgeschrittener Stufe auf benutzerfreundliche Weise in eine praktische, logische Reihe.

In Kapitel 16, *Mittlere Analogeingangs-VIs* finden Sie spezifische VI-Informationen.

Analogeingangs-Utility-VIs

Analogeingangs-Utility-Icon

Sie können auf die Palette **Analogeingangs-Utility** durch Auswahl des Icons Analogeingangs-Utility in der Palette **Analogeingang** zugreifen. Die Analogeingangs-Utility-VIs - AI einen Scan lesen, AI Signalverlauf abtasten und AI kontinuierlich abtasten- sind Einzel-VI-Lösungen für häufig vorkommende Analogeingangsprobleme. Diese VIs sind benutzerfreundlich, wobei es ihnen jedoch an Flexibilität mangelt. Diese drei VIs werden aus den Mittleren Analogeingangs-VIs in der Palette **Analogeingang** erstellt.

In Kapitel 17, *Analogeingangs-Utility-VIs* finden Sie spezifische VI-Informationen.

Fortgeschrittene Analogeingangs-VIs

Fortgeschrittenes Analogeingangs-Icon

Sie können auf die Palette **Fortgeschrittener Analogeingang** durch Auswahl des Icons Fortgeschrittener Analogeingang in der Palette **Analogeingang** zugreifen. Diese VIs bilden die Schnittfläche zur NI-DAQ-Software und sind die Grundlage der Einfachen,- Utility- und Mittleren Analogeingangs-VIs.

In Kapitel 18, *Fortgeschrittene Analogeingangs-VIs* finden Sie spezifische VI-Informationen.

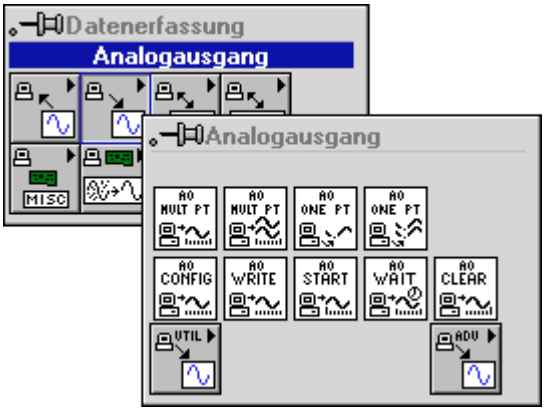
Auffinden eines Analogeingangs-VIs

Beispiele zur Benutzung von Analogeingangs-VIs finden Sie unter `examples\daq\anlogin\anlogin.llb`.

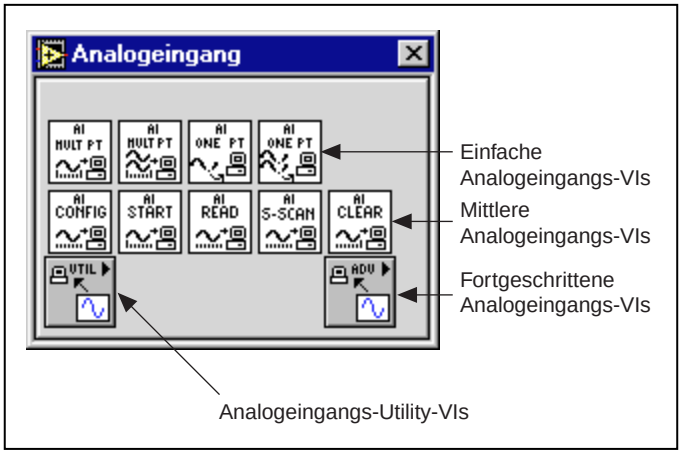
Analogausgangs-VIs

Diese VIs führen Analogausgangs-Operationen durch.

Die Analogausgangs-VIs können Sie durch Auswahl von **Funktionen» Datenerfassung»Analogausgang** finden. Wenn Sie auf das Icon **Analogausgang** in der Palette **Datenerfassung** klicken, wird, wie in nachstehender Abbildung gezeigt, die Palette **Analogausgang** angezeigt.



In der Palette **Analogausgang** sind vier Klassen von Analogausgangs-VIs vorhanden: die Einfachen Analogausgangs-VIs, die Mittleren Analogeingangs-VIs, die Analogausgangs-Utility-VIs und die Fortgeschrittenen Analogausgangs-VIs. Nachstehend werden diese VI-Klassen beschrieben.



Einfache Analogausgangs-VIs

Die Einfachen Analogausgangs-VIs führen einfache Analogausgangsfunktionen durch. Sie können diese VIs vom Frontpanel aus bedienen oder sie als SubVIs in Grundanwendungen benutzen.

Sie können jedes VI separat als analoge Grundausgangsoperation einsetzen. Im Gegensatz zu den VIs der Mittleren und der Fortgeschrittenen Stufe, werden Sie von den Einfachen Analogausgangs-VIs mittels

Dialogfelder auf Fehler aufmerksam gemacht. Sie werden aufgefordert, die Ausführung zu stoppen oder den Fehler zu ignorieren.

Die Einfachen Analogausgangs-VIs sorgen für eine benutzerfreundliche Grundschnittfläche mit lediglich den am häufigsten benutzten Eingängen und Ausgängen. Es wird empfohlen, für komplexere Anwendungen die Mittleren Analogausgangs-VIs oder die Fortgeschrittenen Analogausgangs-VIs zu benutzen, um eine größere Funktionalität und Leistung zu erreichen.

In Kapitel 19, *Einfache Analogausgangs-VIs*, finden Sie spezifische VI-Informationen.

Mittlere Analogausgangs-VIs

Sie können Mittlere Ausgangs-VIs an zwei verschiedenen Stellen in der Palette **Analogausgang** finden. Sie können die Mittleren Analogausgangs-VIs in der zweiten Reihe der Palette **Analogausgang** finden. Die anderen VIs mittlerer Stufe befinden sich in der Palette **Analogausgangs-Utilities**, die später besprochen wird. Die Mittleren Analogausgangs-VIs - AO konfigurieren, AO schreiben, AO starten, AO warten und AO zurücksetzen - bauen wiederum auf der fundamentalen Bausteinschicht, Fortgeschrittene Analogeingangs-VIs genannt, auf. Diese VIs bieten fast ebensoviel Leistungskraft wie die VIs Fortgeschrittener Stufe; sie gruppieren die VIs Fortgeschrittener Stufe auf benutzerfreundliche Weise in eine praktische, logische Reihe.

In Kapitel 20, *Mittlere Analogausgangs-VIs*, finden Sie spezifische VI-Informationen.

Analogausgangs-Utility-VIs

Analogausgangs-Utility-Icon

Sie können auf die Palette **Analogausgangs-Utility** durch Auswahl des Icons Analogausgangs-Utility in der Palette **Analogausgang** zugreifen. Die Analogausgangs-Utility-VIs - AI einen Scan lesen, AI Signalverlauf abtasten und AI kontinuierlich abtasten - sind Einzel-VI-Lösungen für häufig vorkommende Analogausgangsprobleme. Diese VIs sind benutzerfreundlich, wobei es ihnen jedoch an Flexibilität mangelt. Diese drei VIs werden aus den Mittleren Analogausgangs-VIs in der Palette **Analogausgang** erstellt.

In Kapitel 21, *Analogausgang-Utility-VIs*, finden Sie spezifische VI-Informationen.

Fortgeschrittene Analogausgangs-VIs

Fortgeschrittenes Analogausgangs-Icon

Sie können auf die Palette **Fortgeschrittene Analogausgangs-VIs** durch Auswahl des Icons Fortgeschrittener Analogausgang in der Palette **Analogausgang** zugreifen. Diese VIs bilden die Schnittfläche zur NI-DAQ-Software und sind die Grundlage der Einfachen,- Utility- und Mittleren Analogausgangs-VIs.

Da alle diese VIs auf die VIs Fortgeschrittener Stufe gestützt sind, können Sie in Kapitel 22, *Fortgeschrittene Analogausgangs-VIs* weitere Information zu den Ein- und Ausgängen und deren Funktion finden.

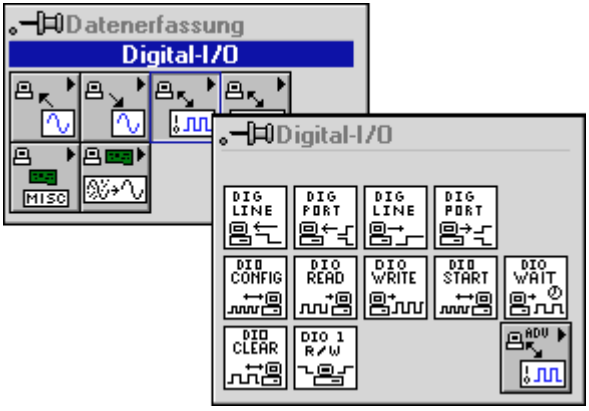
Auffinden eines Analogausgangs-VIs

Beispiele zur Benutzung von Analogausgangs-VIs finden Sie unter `examples\daq\anlogout\anlogout.llb`.

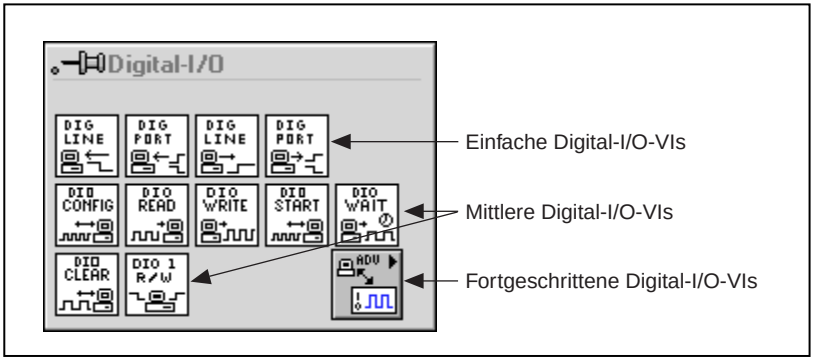
VIs mit digitaler Operation

Diese VIs führen digitale Operationen aus.

Sie können Digital-I/O-VIs durch Auswahl von **Funktionen» Datenerfassung»Digital-I/O** finden. Wenn Sie den Mauszeiger auf das Icon Digital-I/O in der Palette **Datenerfassung** bewegen, wird, wie in nachstehender Abbildung gezeigt, die Palette **Digital-I/O** angezeigt.



In der Palette **Digital-I/O** sind drei Klassen von Digital-I/O-VIs vorhanden. Die Einfachen I/O-VIs, die Mittleren I/O-VIs und die Fortgeschrittenen I/O-VIs. Nachstehend werden diese VI-Klassen beschrieben.



Einfache Digital-I/O-VIs

Die einfachen Digital-I/O-VIs führen einfache digitale Operationen durch. Sie können diese VIs vom Frontpanel aus ausführen oder sie als Sub-VIs in Grundanwendungen benutzen.

Sie können jedes VI separat als digitale Grundoperation einsetzen. Im Gegensatz zu den VIs der Mittleren und der Fortgeschrittenen Stufe, werden Sie von den Einfachen Digital-I/O-VIs mittels Dialogfelder auf Fehler aufmerksam gemacht. Sie werden aufgefordert, die Ausführung zu stoppen oder den Fehler zu ignorieren.

Die Einfachen Digital-I/O-VIs bestehen tatsächlich aus Mittleren Digital-I/O-VIs, die wiederum aus Fortgeschrittenen Digital-I/O-VIs bestehen. Die Einfachen Digital-I/O-VIs sorgen für eine benutzerfreundliche Grundschnittfläche mit lediglich den am häufigsten benutzten Eingängen und Ausgängen. Es wird empfohlen, für komplexere Anwendungen die Mittleren oder Fortgeschrittenen VI-Stufen zu benutzen, um eine größere Funktionalität und Leistung zu erreichen.

In Kapitel 23, *Einfache Digital-I/O-VIs*, finden Sie spezifische VI-Information.

Mittlere Digital- I/O-VIs

Sie können Mittlere Digital- I/O-VIs in den Reihen zwei und drei der Palette **Digital-I/O** finden. Die Mittleren Digital-I/O-VIs bauen wiederum auf der fundamentalen Bausteinschicht, Fortgeschrittene Digital-I/O- VIs,

auf. Diese VIs bieten fast ebensoviel Leistungskraft wie die VIs fortgeschrittener Stufe; sie gruppieren die VIs fortgeschrittener Stufe auf benutzerfreundliche Weise in eine praktische, logische Reihe.

In Kapitel 24, *Mittlere Digital-I/O-VIs*, finden Sie spezifische VI-Informationen.

Fortgeschrittene Digital- I/O-VIs

Fortgeschrittenes Digital-I/O-Icon

Sie können auf die Palette **Fortgeschrittene Digital-I/O** durch Auswahl des Icons **Fortgeschrittene Digital-I/O** in der Palette **Digital-I/O** zugreifen. Diese VIs bilden die Schnittstelle der NI-DAQ-Software und sind die Grundlage der Einfachen,- Utility- und Mittleren Digital-I/O-VIs.

Da alle diese VIs auf die VIs fortgeschrittener Stufe gestützt sind, können Sie in Kapitel 25, *Fortgeschrittene Digital-I/O-VIs* weitere Information zu den Ein- und Ausgängen und deren Funktion finden.

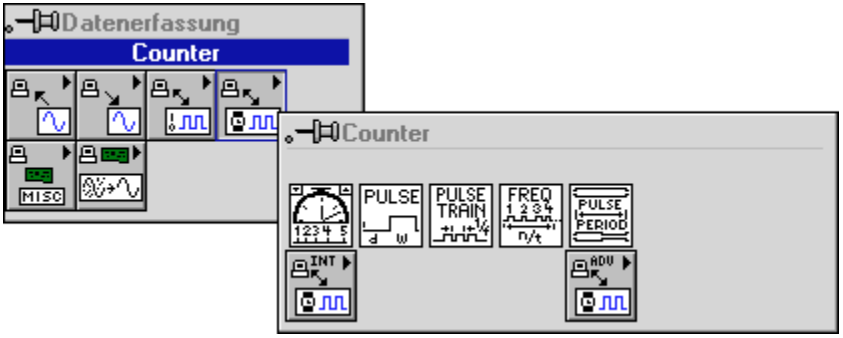
Beispiele zum Auffinden eines Digital-I/O-VIs

Beispiele zur Benutzung von Digital-I/O-VIs finden Sie unter `examples\daq\digital\digio.llb`.

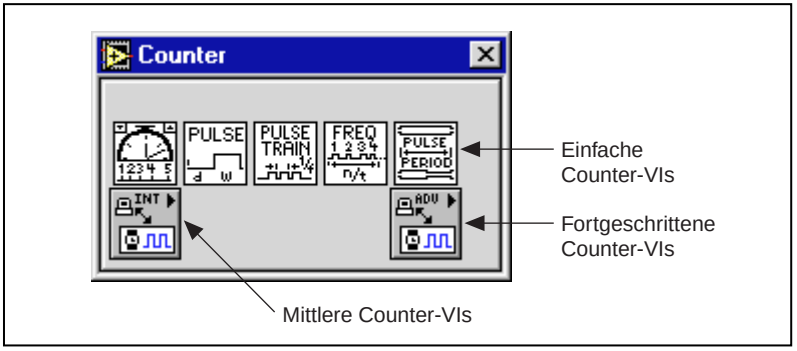
Counter-VIs

Diese VIs führen Zählfunktionen durch.

Sie können Counter-VIs durch Auswahl von **Funktionen» Datenerfassung»Counter** finden. Wenn Sie auf das Icon Counter in der Palette **Datenerfassung** klicken, wird die Palette **Counter,** wie in nachstehender Abbildung dargestellt, angezeigt.



Die Palette **Counter** enthält drei Klassen Counter-VIs: Die Einfachen, Mittleren und Fortgeschrittenen Counter-VIs. Nachstehend werden diese VI-Klassen beschrieben.



Einfache Counter-VIs

Die Einfachen Counter-VIs führen einfache Zähloperationen durch. Sie können diese VIs vom Frontpanel aus bedienen oder sie als SubVIs in Grundanwendungen benutzen.

Sie können jedes VI separat als digitale Grundoperation ausführen. Im Gegensatz zu den VIs der mittleren und der fortgeschrittenen Stufe, werden Sie von den Einfachen Counter-VIs mittels Dialogfelder auf Fehler aufmerksam gemacht. Sie werden aufgefordert, die Ausführung zu stoppen oder den Fehler zu ignorieren.

Die Einfachen Counter-VIs bestehen tatsächlich aus Mittleren Counter-VIs, die wiederum aus Fortgeschrittenen Counter-VIs bestehen. Die Einfachen Counter-VIs sorgen für eine benutzerfreundliche Grundschnittfläche mit lediglich den am häufigsten benutzten Eingängen und Ausgängen. Es wird empfohlen, für komplexere Anwendungen die mittleren oder fortgeschrittenen VI-Stufen zu benutzen, um eine größere Funktionalität und Leistung zu erreichen.

In Kapitel 26, *Einfache-Counter-VIs*, finden Sie spezifische VI-Informationen.

Mittlere Counter-Eingangs-VIs

Mittleres Counter-VI-Icon

Sie können Mittlere Counter-VIs in der zweiten Reihe der Palette **Counter** finden. Die Mittleren Counter-VIs bauen wiederum auf der fundamentalen Bausteinschicht, Fortgeschrittene Counter-VIs auf. Diese VIs, bieten fast ebensoviel Leistungskraft wie die VIs fortgeschrittener Stufe; sie gruppieren die VIs fortgeschrittener Stufe auf benutzerfreundliche Weise in eine praktische, logische Reihe.

In Kapitel 27, *Mittlere Counter-VIs*, finden Sie spezifische VI-Informationen.

Fortgeschrittene Counter-VIs

Fortgeschrittenes Counter-VI-Icon

Sie können auf die Palette **Fortgeschrittener Counter** durch Auswahl des Icons **Fortgeschrittener Counter** in der Palette **Counter** zugreifen. Diese VIs bilden die Schnittstelle der NI-DAQ-Software und sind die Grundlage der Einfachen und Mittleren Counter-VIs.

Da alle diese VIs auf die VIs fortgeschrittener Stufe gestützt sind, können Sie in Kapitel 28, *Fortgeschrittene Counter-VIs*, weitere Information zu den Ein- und Ausgängen und deren Funktion finden.

Auffinden eines Counter-VIs

Beispiele zur Benutzung von Counter-VIs finden Sie, indem Sie die Beispielbibliotheken folgendermaßen öffnen `examples\daq\counter\DAQ-STC.llb`, `examples\daq\counter\am9513.llb` und `examples\daq\counter\8253.llb`.

Kalibrierungs- und Konfigurations-VIs

Diese VIs kalibrieren spezifische Geräte und stellen Konfigurationsinformationen ein und geben diese zurück.

In Kapitel 29, *Kalibrierungs- und Konfigurations-VIs*, finden Sie Informationen zum Auffinden dieser VIs sowie Beispiele.

Signalkonditionierungs-VIs

Diese VIs konvertieren die von den Widerstandstemperaturfühlern, Dehnungsmessern oder Thermoelementen abgelesenen Analogeingangsspannungen in Dehnungs- oder Temperatureinheiten.

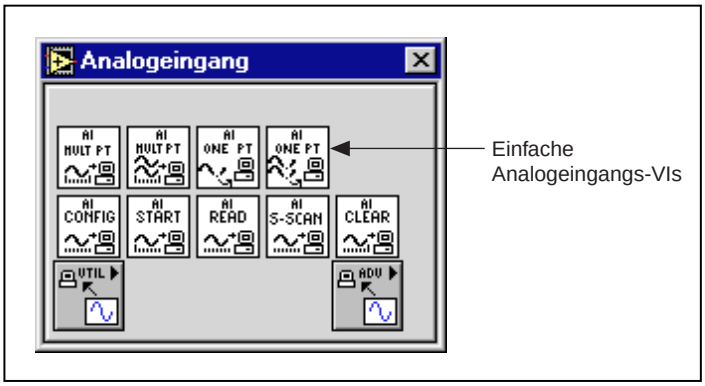
In Kapitel 30, *Signalkonditionierungs-VIs*, finden Sie Informationen zum Auffinden dieser VIs sowie Beispiele.

15

Einfache Analogeingangs-VIs

In diesem Kapitel werden Einfache Analogeingangs-VIs beschrieben, die einfache Analogeingangsoperationen durchführen. Sie können diese VIs vom Frontpanel aus ausführen oder sie als SubVIs in Grundanwendungen einsetzen.

Sie können auf die Einfachen Analogeingangs-VIs durch Auswahl von **Funktionen»Datenerfassung»Analogeingang** zugreifen. Bei den Einfachen Analogeingangs-VIs handelt es sich, wie nachstehend abgebildet, um die VIs in der obersten Reihe der Palette **Analogeingang**.

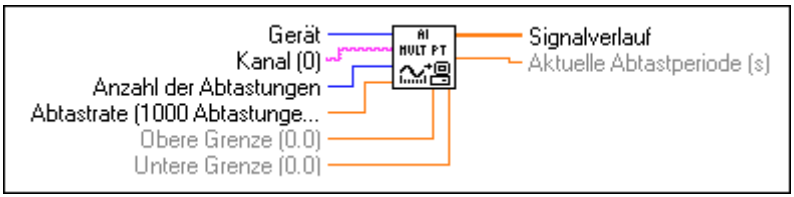


Beschreibungen der Einfachen Analogeingangs-VIs

Folgende Einfache Analogeingangs-VIs sind verfügbar.

AI Signalverlauf erfassen

Erfaßt eine bestimmte Anzahl von Abtastwerten mit einer bestimmten Abtastfrequenz von einem Einzel-Eingangskanal und gibt die erfaßten Daten zurück.

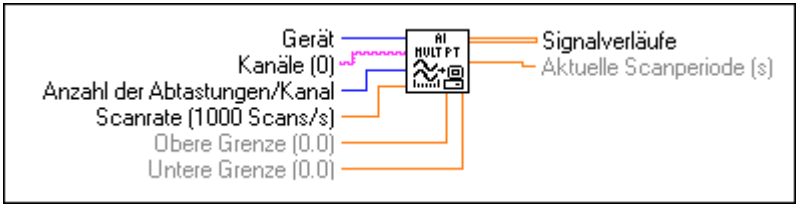


Das VI AI Signalverlauf erfassen führt eine hardwaregetaktete Messung einer Kurvenform (Mehrfach-Spannungsablesung bei festgelegter Abtastrate) an einem Einzel-Analogeingangskanal durch. Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, das VI zu stoppen oder fortzusetzen.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalnummern und Eingabegrenzen.

AI Signalverläufe erfassen

Erfaßt Daten von den angegebenen Kanälen und tastet die Kanäle mit der angegebenen Scanrate ab.

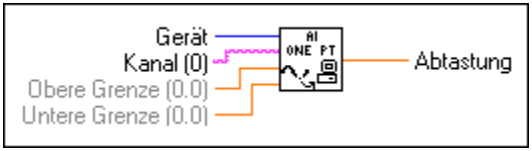


Das AI Kurvenformerfassungs-VI führt eine getaktete Messung verschiedener Kurvenformen an den angegebenen Analogeingangskanälen durch. Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, die Operation abzubrechen oder die Ausführung fortzusetzen.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalnummern und Eingabegrenzen.

AI Abtastkanal

Mißt das mit dem angegebenen Kanal verbundene Signal und gibt den gemessenen Wert zurück.

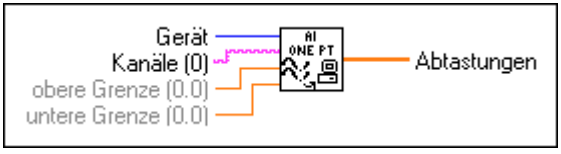


Das AI Abtastkanal-VI führt eine einzelne, nicht getaktete Messung eines Kanals durch. Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, das VI zu stoppen oder fortzusetzen.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalnummern und Eingabegrenzen.

AI Abtastkanäle

Führt eine einzelne Ablesung von jedem angegebenen Kanal durch.



Das AI Abtastkanal-VI mißt von jedem der angegebenen Analogeingangskanäle einen einzelnen Wert. Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, das VI zu stoppen oder fortzusetzen.

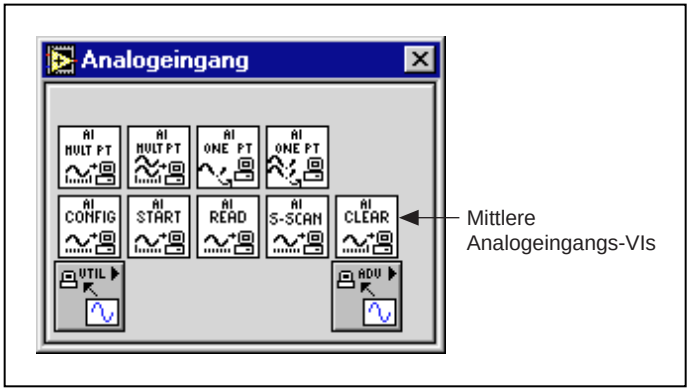
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalnummern und Eingabegrenzen.

16

Mittlere Analogeingangs-VIs

In diesem Kapitel werden die Mittleren Analogeingangs-VIs beschrieben. Diese VIs sind benutzerfreundlich, es mangelt ihnen jedoch an Flexibilität.

Sie können auf die Mittleren Analogeingangs-VIs durch Auswahl von **Funktionen»Datenerfassung»Analogeingang** zugreifen. Die Mittleren Analogeingangs-VIs sind, wie nachstehend dargestellt, die VIs in der zweiten Reihe der Palette **Analogeingang.**



Fehlerbehandlung

LabVIEW macht die Fehlerbehandlung mit den Mittleren Analogeingangs-VIs einfach. Jedes Mittlere VI hat ein Eingangscluster **Fehlereingang** und ein Ausgangscluster **Fehlerausgang**. Die Cluster enthalten einen Booleschen Wert, wodurch angezeigt wird, ob ein Fehler vorgekommen ist, sowie den Fehlercode für den Fehler und den Namen des VIs, das den Fehler zurückgegeben hat. Wenn **Fehlereingang** einen Fehler anzeigt, gibt das VI die Fehlerinformation an **Fehlerausgang** zurück und stellt die Ausführung ein.

☞ **Hinweis** ***Das VI AI zurücksetzen stellt eine Ausnahme zu dieser Regel dar - dieses VI beendet die Erfassung immer, und zwar unabhängig davon, ob Fehlereingang einen Fehler anzeigt.***

Wenn Sie eines der Mittleren Analogeingangs-VIs in einer While-Schleife benutzen, sollten Sie die Schleife stoppen, wenn der **Status** im Cluster **Fehlerausgang** TRUE anzeigt. Wenn Sie den Fehlercluster mit dem Allgemeinen Error-Handler-VI verbinden, entziffert das VI die Fehlerinformation und beschreibt den Fehler.

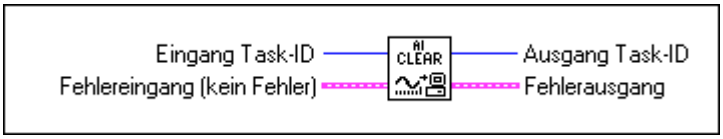
Das Allgemeine Error-Handler-VI befindet sich in **Funktionen**»**Zeit und Dialog** in LabVIEW.

Beschreibungen der Mittleren Analogeingangs-VIs

Folgende Mittlere Analogeingangs-VIs sind verfügbar.

AI zurücksetzen

Setzt alle mit **Task-ID** in Zusammenhang stehenden Eingangstasks zurück.



Das VI AI zurücksetzen stoppt eine mit **Task-ID** in Zusammenhang stehende Erfassung und gibt damit in Zusammenhang stehende interne Quellen, einschließlich Puffer, frei. Ehe Sie eine neue Erfassung beginnen, müssen Sie das VI AI konfigurieren aufrufen. In Kapitel 18, *Fortgeschrittene Analogeingangs-VIs*, finden Sie Beschreibungen zum VI AI Steuerung.

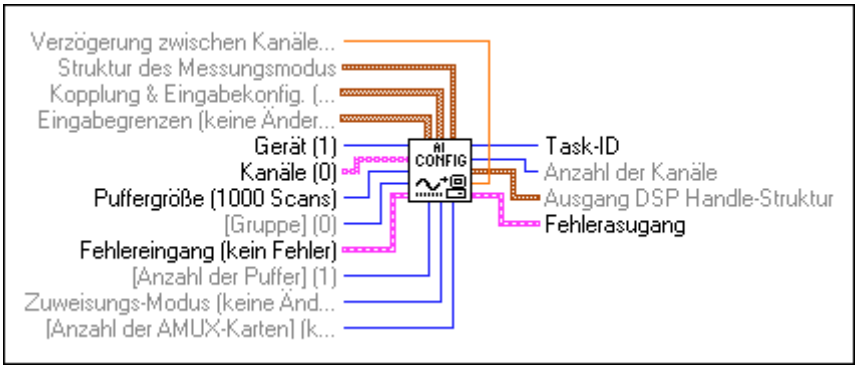
☞ **Hinweis** ***Das VI AI zurücksetzen beendet die Erfassung immer, und zwar unabhängig davon, ob in* Fehlereingang *ein Fehler angezeigt wird.***

Wenn Sie eines der Mittleren Analogeingangs-VIs in einer While-Schleife benutzen, sollten Sie die Schleife stoppen, wenn der **Status** im Cluster **Fehlerausgang** TRUE anzeigt. Wenn Sie den Fehlercluster mit dem Allgemeinen Error-Handler-VI verbinden, entziffert das VI die Fehlerinformation und beschreibt den Fehler.

Das Allgemeine Error-Handler-VI befindet sich in **Funktionen»Zeit und Dialog** in LabVIEW. Für weitere Informationen sehen Sie bitte in Kapitel 10, *Zeit-, Dialog- und Fehlerfunktionen* nach.

AI konfigurieren

Konfiguriert eine Analogeingangsoperation für einen bezeichneten Kanalsatz. Dieses VI konfiguriert die Hardware und weist für eine gepufferte Analogeingangsoperation einen Puffer zu.



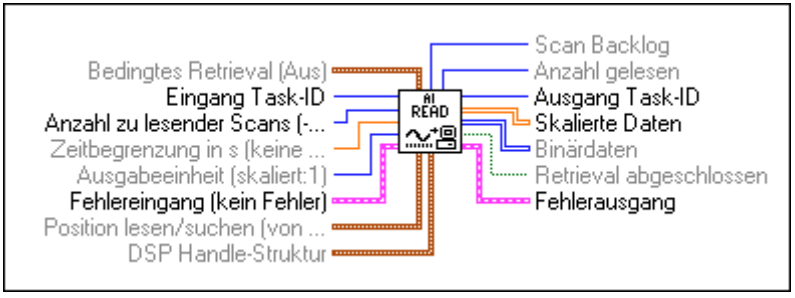
Sie können nur mit den folgenden Geräten mehr als einen Puffer zuweisen.

- **(Macintosh)** NB-A2000, NB-A2100 und NB-A2150

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie Kanalbereiche, Eingabegrenzen und Abtastreihenfolgen, die Sie mit Ihrem DAQ-Gerät von National Instruments benutzen können.

AI lesen

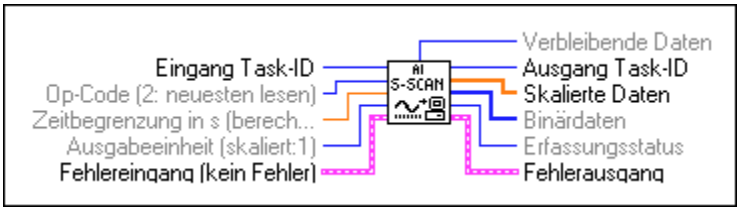
Liest Daten von der gepufferten Datenerfassung.



Das VI AI lesen ruft das VI AI Puffer lesen zur Ablesung von Daten von einer gepufferten Analogeingangserfassung auf.

AI einzelner Scan

Gibt eine Datenabtastung von einer vorher konfigurierten Kanalgruppe zurück.



Wenn Sie mit dem VI AI starten bereits eine Erfassung begonnen haben, liest dieses VI eine Abtastung von den Erfassungspufferdaten oder dem Onboard-FIFO, wenn die Erfassung ungepuffert ist. Wenn Sie keine Erfassung begonnen haben, beginnt dieses VI mit der Erfassung, fragt alle Datenabtastungen ab und schließt dann die Erfassung ab. Die Kanäle der VI-Abtastungen werden durch die Gruppenkonfiguration bestimmt.

Wenn Sie das VI AI starten nicht aufrufen, löst dieses VI eine einzelne Abtastung aus und benutzt dafür die schnellste zulässige Kanaltaktfrequenz. Sie können die Kanaltaktfrequenz mit dem VI AI konfigurieren ändern.

Wenn Sie das VI AI starten ausführen, werden die Abtastungen durch ein Taktsignal ausgelöst.

Sie müssen das VI AI starten benutzen, um den Taktgeber für extern taktgesteuerte Umwandlungen auf extern einzustellen.

Wenn es sich um interne Taktgeber handelt und Sie keinen Speicher zuweisen, beginnt eine getaktete, nicht gepufferte Erfassung, wenn Sie das VI AI starten ausführen. Diese Art der Erfassung wird für die Taktung von Analogeingängen und -ausgängen in einer punktweisen Bedienelement-Anwendung benutzt. Folgende Geräte unterstützen getaktete, nicht gepufferte Erfassungen nicht.

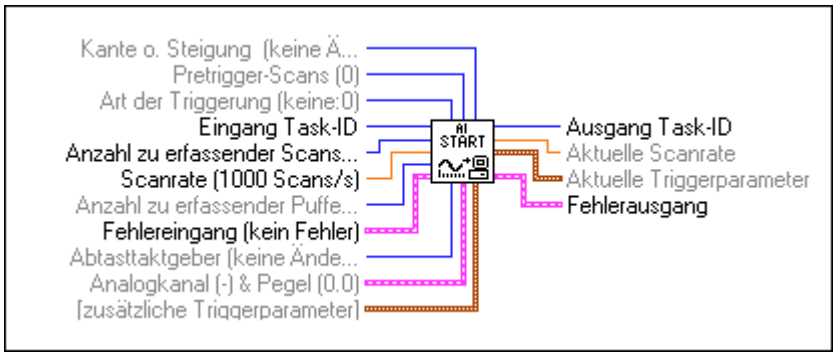
- **(Macintosh)** NB-A2000, NB-A2100 und NB-A2150

☞ **Hinweis** ***Im Falle eines FIFO-Überlaufs während einer getakteten, nicht gepufferten Erfassung startet LabVIEW das Gerät neu.***

Wenn Sie **Op-Code** für eine nicht gepufferte Erfassung auf 1 setzen, liest das VI eine Abtastung vom FIFO ab und gibt die Daten zurück. Wenn **Op-Code** auf 2 gesetzt ist, nimmt das VI Ablesungen vom FIFO vor, bis es leer ist, und es gibt die letzte gelesene Abtastung zurück.

AI starten

Startet eine gepufferte Analogeingangsoperation. Dieses VI setzt die Scanrate, die Anzahl der zu erfassenden Scans und die Triggerbedingungen. Dann beginnt das VI mit der Erfassung.



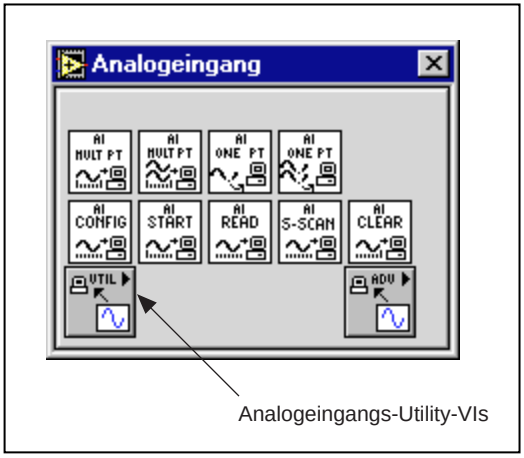
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie Kanalbereiche, Eingabegrenzen, Abtastreihenfolgen, Trigger und Takte, die Sie mit Ihrem DAQ-Gerät von National Instrument benutzen können.

17

Analogeingangs-Utility-VIs

Dieses Kapital beschreibt die Analogeingangs-Utility-VIs. Die VIs - AI einen Scan lesen, AI Signalverlauf abtasten und AI kontinuierlich abtasten - sind Einzel-VI-Lösungen für häufig vorkommende Analogeingangsprobleme. Die Analogeingangs-Utility-VIs sind Mittlere VIs, die somit auf den VIs Fortgeschrittener Stufe beruhen. In Kapitel 18, *Fortgeschrittene Analogeingangs-VIs* finden Sie zusätzliche Informationen zu Ein- und Ausgängen und deren Funktionen.

Sie können auf die Palette **Analogeingangs-Utilities** durch Auswahl von **Funktionen**»**Datenerfassung**»**Analogeingang**»**Analogeingangs-Utilities** zugreifen. Das Icon, das Sie auswählen müssen, um auf die Analogeingangs-Utility-VIs zugreifen zu können, befindet sich, wie nachstehend dargestellt, in der untersten Reihe der Palette **Analogeingang.**



Fehlerbehandlung

LabVIEW macht die Fehlerbehandlung durch die Mittleren Analogeingangs-Utility-VIs einfach. Jedes Mittlere VI hat ein Eingangscluster **Fehlereingang** und ein Ausgangscluster **Fehlerausgang**. Die Cluster enthalten einen Booleschen Wert, wodurch angezeigt wird, ob ein Fehler vorgekommen ist, sowie den Fehlercode für den Fehler und den Namen des VIs, das den Fehler zurückgegeben hat. Wenn **Fehlereingang** einen Fehler anzeigt, gibt das VI die Fehlerinformation an **Fehlerausgang** aus und stellt seine Ausführung ein.

Wenn Sie einen der Mittleren Analogeingangs-Utility-VIs in einer While-Schleife benutzen, sollten Sie die Schleife stoppen, wenn der **Status** im Cluster **Fehlerausgang** TRUE anzeigt. Wenn Sie den Fehlercluster mit dem Allgemeinen Error-Handler-VI verbinden, entziffert das VI die Fehlerinformation und beschreibt den Fehler.

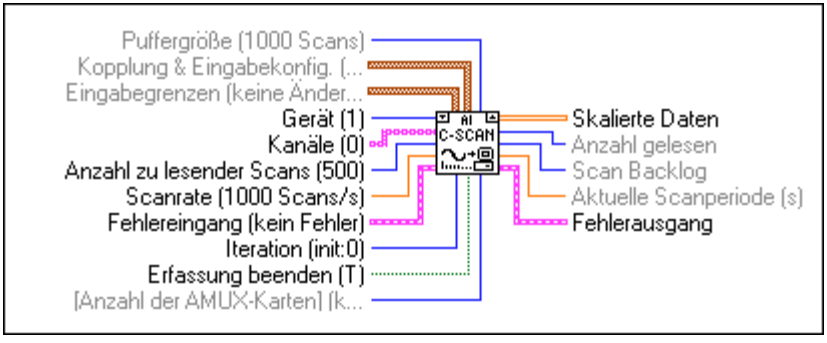
Das Allgemeine Error-Handler-VI befindet sich in LabVIEW in **Funktionen»Zeit und Dialog**. Weitere Informationen zu diesem VI finden Sie in Kapitel 10, *Zeit-, Dialog- und Fehlerfunktionen*.

Beschreibungen des Analogeingangs-Utility-VIs

Folgende VIs sind durch die Analogeingangs-Utility-Unterpalette verfügbar.

AI kontinuierlich abtasten

Führt laufende Zeitabtastwert-Messungen einer Kanalgruppe durch, speichert die Daten in einem Umlaufpuffer und gibt bei jedem Aufruf eine vorgegebene Anzahl von Abtastmessungen zurück.



Iterations terminal

Das VI AI kontinuierlich abtasten tastet eine Kanalgruppe unentwegt ab, wie z.B. für Datenprotokollierungs-Anwendungen. Plazieren Sie das VI in eine While-Schleife, und verbinden Sie das Iterationsterminal der Schleife mit dem VI **Iteration**seingang.

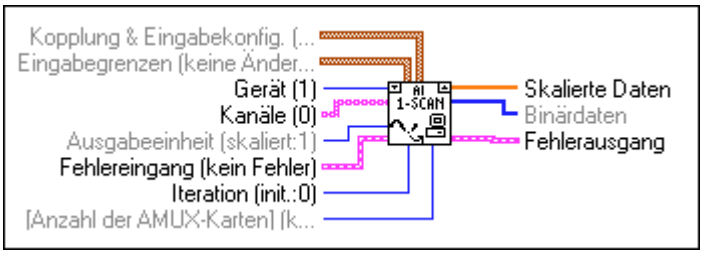
Verbinden Sie auch die Bedingung, die die Schleife abschließt, mit dem Eingang **Erfassung beenden,** invertieren Sie, wenn nötig, das Signal, so daß es bei der letzten Iteration TRUE anzeigt. Bei der Iteration 0 ruft das VI das VI AI konfigurieren auf, um die Kanalgruppe und Hardware zu konfigurieren und einen Datenpuffer zuzuweisen; das VI ruft das VI AI starten auf, um die Scanrate zu setzen und die Erfassung zu beginnen. Bei jeder Iteration ruft das VI das VI AI lesen auf, um die Anzahl der Messungen abzufragen, die durch die **Anzahl der zu lesenden Scans** bestimmt sind, tastet diese ab und gibt die Daten als Array abgetasteter Werte zurück. Bei der letzten Iteration (wenn **Erfassung beendender** ist) oder bei Vorkommen eines Fehlers, ruft das VI das VI AI zurücksetzen auf, um alle momentan durchgeführten Erfassungen zu beenden. Der Aufruf des VIs AI kontinuierlich abtasten außerhalb einer Schleife sollte nicht notwendig sein; wenn Sie dies aber dennoch tun, können Sie die **Iteration** verlassen und brauchen die Eingänge von Erfassung beenden nicht zu verbinden.

Wenn Sie das VI AI kontinuierlich abtasten in einer Schleife aufrufen, um Daten aus der laufenden Erfassung schnell zu lesen, müssen Sie die Daten schnell genug lesen, so daß diese nicht mit neu erfaßten Daten überschrieben werden. Der Ausgang **Scan Backlog** zeigt an, wieviel Daten vom VI erfaßt, aber nicht gelesen wurden. Wenn das Backlog ständig anwächst, überschreiben Ihre neuen Daten irgendwann Ihre alten Daten. Fragen Sie Daten öfters ab oder passen Sie zur Lösung dieses Problems die **Puffergröße**, die **Scanrate** oder die **Anzahl** der **zu lesenden Scans** an.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie Informationen zu Kanalbereichen, Eingabegrenzen und Abtastreihenfolgen, die Sie mit dem DAQ-Gerät von National Instruments benutzen können.

AI einen Scan lesen

Mißt die Signale an den angegebenen Kanälen und gibt die Messungen in einem Array skalierter oder binärer Werte zurück.



Iteration terminal

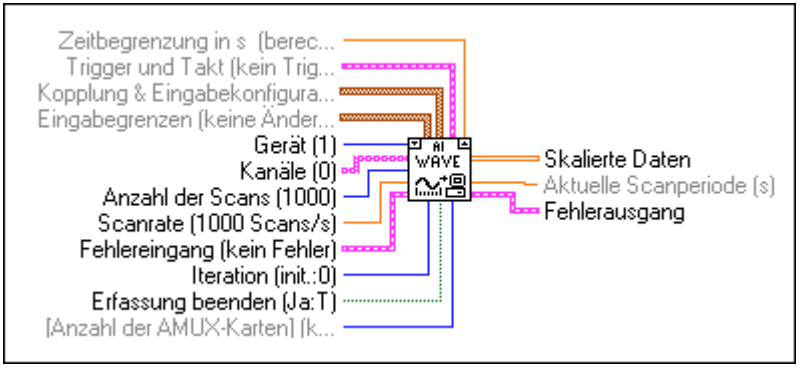
Das VI AI einen Scan lesen führt die sofortige Messung einer Gruppe von einem oder mehreren Kanälen durch. Wenn Sie das VI zur Erlangung mehrfacher Messungen von einer Kanalgruppe in eine Schleife geben, müssen Sie das Schleifen-Iterationsterminal mit dem VI **Iterationsparameter** verbinden.

Bei Iteration 0 ruft dieses VI zur Konfiguration der Kanalgruppe und der Hardware das VI AI konfigurieren auf; dann ruft es das VI AI einzelner Scan auf, um die Ergebnisse zu messen und zu berichten. Bei nachfolgenden Iterationen vermeidet das VI unnötige Konfigurationen und ruft nur das VI AI einzelner Scan auf. Wenn Sie das VI AI einen Scan lesen zur Vornahme einer einzelnen Messung von einer Kanalgruppe aufrufen, muß der **Iterations**parameter nicht verbunden werden.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalbereiche, Eingabegrenzen und Abtastreihenfolgen.

AI Signalverlauf abtasten

Erfaßt die angegebene Anzahl von Scans mit der angegebenen Scanrate und gibt die erfaßten Daten zurück. Sie können die Erfassung triggern.



Iteration terminal

Das VI AI Signalverlauf abtasten erfaßt die genannte Anzahl von Scans von einer Kanalgruppe mit der angegebenen Scanrate. Wenn Sie das VI zur Erlangung mehrfacher Messungen derselben Kanalgruppe in eine Schleife plazieren, müssen Sie das Iterationsterminal mit dem VI **Iterationseingang** verbinden.

Verbinden Sie auch die Bedingung, die die Schleife abschließt, mit dem VI **Eingang** Erfassung beenden; invertieren Sie, wenn nötig, das Signal, so daß die Ablesung bei der letzten Iteration TRUE ist. Bei Iteration 0 ruft dieses VI zur Konfiguration der Kanalgruppe und der Hardware und zur Zuweisung eines Datenpuffers das VI AI konfigurieren auf. Bei jeder Iteration ruft dieses VI die VIs AI starten und AI lesen auf. Das VI AI starten legt die Scanrate und die Triggerbedingungen fest und startet die Erfassung. Das VI speichert die

Messungen während der Erfassung im Puffer; das VI AI Read liest die Messungen vom Puffer ab, skaliert sie und gibt alle Daten als ein Array skalierter Werte zurück. Bei der letzten Iteration (wenn **Erfassung beenden** TRUE ist) oder bei Vorkommen eines Fehlers ruft das VI zur Löschung der derzeit durchgeführten Erfassung auch das VI AI zurücksetzen auf. Wenn Sie das VI AI Signalverlauf abtasten nur einmal aufrufen, müssen Sie **Iteration** und **Erfassung beenden** nicht verbinden.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie Kanalbereiche, Eingabegrenzen, Abtastreihenfolgen, Triggers und Takte, die Sie mit Ihrem DAQ-Gerät von National Instruments DAQ benutzen können.

☞ **Hinweis** ***Diese VIs benutzen ein nicht-initialisiertes Schieberegister als lokalen Speicher, um sich zwischen den VI-Aufrufen die TaskID für die Kanalgruppe zu merken. Üblicherweise benutzen Sie ein VI an einer Stelle Ihres Diagramms; wenn Sie es jedoch mehr als einmal benutzen, teilen sich die Mehrfachinstanzen des VIs dieselbe Task-ID. Alle Aufrufe an eines dieser VIs konfigurieren oder lesen Daten von der Erfassung oder beenden dieselbe Erfassung. Gelegentlich möchten Sie eventuell jedes VI an verschiedenen Stellen benutzen, wobei sich dann jedes Vorkommnis auf eine andere Task-ID beziehen soll (wenn Sie z.B. zwei Geräte gleichzeitig messen). Speichern Sie eine Kopie des VIs unter einem anderen Namen (z.B. AI Signalverlauf abtasten R), und machen Sie Ihr neues VI ablaufinvariant.***

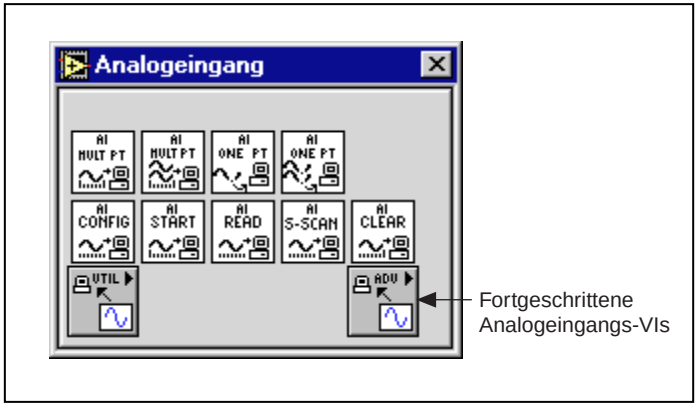
☞ **Hinweis** ***Betrifft alle Analogeingangs-Utility-VIs: Wenn Ihr Programm mehr als*** 2^{31-1} ***Mal iteriert, dürfen Sie den*** **Iteration***seingang nicht mit dem Schleifen-Iterationsterminal verbinden. Statt dessen setzen Sie die* **Iteration** *bei der ersten Schleife auf* 0 *und dann auf jeden beliebigen positiven Wert für alle anderen Iterationen. Das VI rekonfiguriert die Iteration und startet sie neu* ≤ 0.

18

Fortgeschrittene Analogeingangs-VIs

Dieses Kapitel enthält Referenzbeschreibungen der Fortgeschrittenen Analogeingangs-VIs. Diese VIs bilden die Schnittstelle der NI-DAQ-Software und sind die Grundlage für Einfache,- Utility- und Mittlere Analogeingangs-VIs.

Sie können auf die Palette **Fortgeschrittener Analogeingang** durch Auswahl von **Funktionen»Datenerfassung»Analogeingang» Fortgeschrittener Analogeingang** zugreifen. Das Icon, das Sie für den Zugriff auf die Fortgeschrittenen Analogeingangs-VIs auswählen müssen, befindet sich, wie nachstehend dargestellt, in der untersten Reihe der Palette **Analogeingang.**

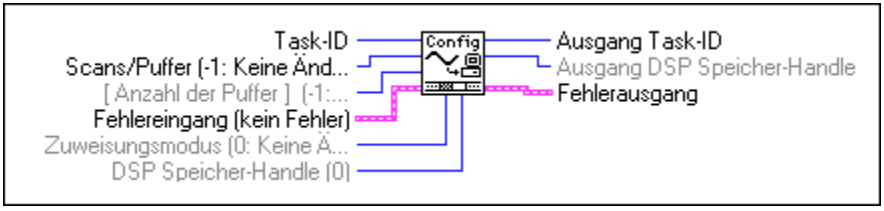


Beschreibungen der Analogeingangs-VIs

Folgende Fortgeschrittene Analogeingangs-VIs sind verfügbar.

AI Puffer-Konfiguration

Weist LabVIEW Speicherplatz zur Speicherung von Analogeingangsdaten zu, bis das VI AI Puffer lesen die Daten an Sie liefern kann. LabVIEW bezieht sich auf die (den) vom VI AI Puffer-Konfiguration zugewiesenen Puffer als internen Puffer, weil Sie nicht direkt darauf zugreifen können.



☞ **Hinweis** ***Wenn Sie das VI AI Steuerung mit einem auf*** `4` ***(löschen) eingestellten Steuercode ausführen, führt das VI eine Funktion aus, die der Ausführung von VI AI Puffer-Konfiguration mit einem auf*** `1` ***eingestellten Zuweisungsmodus entspricht. Dies bedeutet, daß beide VIs die Zuweisung für die internen Analogeingangs-Datenpuffer aufheben. Erfassungen, bei denen ein DSP- oder Erweiterungskartenspeicher benutzt wird, bilden jedoch eine Ausnahme. Das VI AI Steuerung hebt die Zuweisung des DSP-Speichers bei der Löschung einer Erfassung nicht auf. Zur Aufhebung der Zuweisung von DSP-Erfassungspuffer müssen Sie das VI AI Puffer-Konfiguration ausdrücklich aufrufen.***

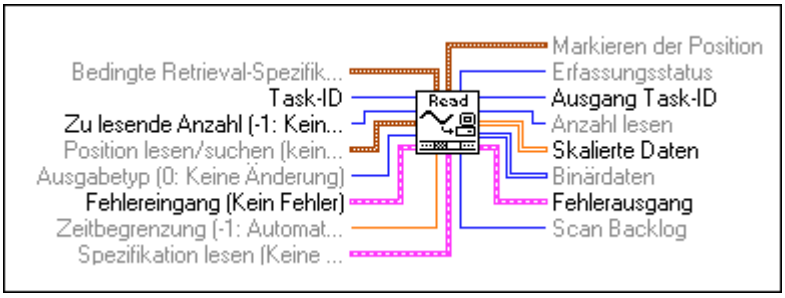
In Tabelle 18-1 sind die Standardeinstellungen und -bereiche für das VI AI Puffer-Konfiguration aufgeführt. In der ersten Reihe sind die Werte für die meisten Geräte aufgeführt, während in den anderen Reihen die Werte für die Geräte aufgeführt sind, die die Ausnahme zur Regel darstellen. In der ersten Reihe sind die Werte für die meisten Geräte und für die Regel aufgeführt.

Tabelle 18-1. AI Puffer-Konfigurations-VI gerätespezifische Einstellungen und Bereiche

Gerät	Scans pro Puffer		Anzahl der Puffer		Zuweisungsmodus	
	Standardeinstellung	Bereich	Standardeinstellung	Bereich	Standardeinstellung	Bereich
Die meisten Geräte	100	0, $n \geq 3$	1	0; 1	2	1; 2
Lab-NB Lab-LC	100	$n \geq 3$	1	0; 1	2	1; 2
NB-A2000 NB-A2100 NB-A2150	100	$n \geq 0$	1	$n \geq 0$	2	1; 2
5102 Geräte	100	$n \geq 3$	1	1	2	1; 2

AI Puffer lesen

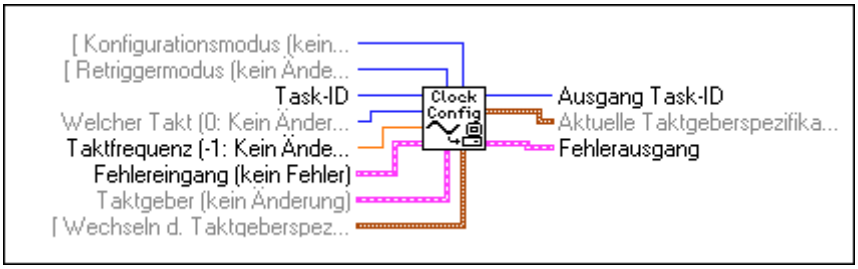
Gibt Analogeingangsdaten von den (vom) internen Datenpuffer(n) zurück.



☞ **Hinweis** ***Wenn das VI Ablesungen von der Triggermarke vornimmt, gibt es solange keine Daten zurück, bis die Erfassung für den Puffer, in dem der Trigger enthalten ist, abgeschlossen ist.***

AI Takt-Konfiguration

Stellt den Kanal und die Abtasttaktraten ein.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Takte.

Sie können für Geräte, die nur einen Kanaltakt haben (Lab-LC, Lab-NB, NB-MIO-16, Lab-PC+, PCI-1200, PC-LPM-16, DAQCard-500, DAQCard-700 und DAQCard-1200), nicht einen unabhängigen Kanal und unabhängige Taktraten einstellen. Die Einstellung eines Werts setzt den anderen zurück, weil die Kanalrate der Scanrate-Anzahl der abzutastenden Kanäle entspricht.

Bei Geräten, die keinen Kanaltakt haben (NB-A2000, NB-A2100 und NB-A2150), wird durch die Einstellung des Kanaltakts ein Fehler erzeugt.

Wenn Sie für die Abtasttaktrate einen Wert von 0 angeben, wird die Intervallabtastung abgeschaltet und die Kanalabtastung (oder Abtastung im zyklischen Warteschlangenbetrieb) wird mit der Kanaltaktrate durchgeführt. Diese Option ist nur für Geräte mit unabhängigen Kanälen und Abtasttakten von Bedeutung.

Die Taktrate ist die Rate, mit der LabVIEW Daten abtastet oder Abtastungen erfaßt. Sie können die Taktrate auf drei verschiedene Arten zum Ausdruck bringen - durch **Taktfrequenz**, durch **Taktperiode** oder durch **Zeitbasis-Quelle**, **Zeitbasis-Signal** und **Zeitbasis-Divisor**. Das VI führt die Suche in diesen Parametern in dieser Reihenfolge durch und stellt die Taktrate mit dem ersten Wert, der ungleich `-1` ist, ein.

Tabelle 18-2 führt Standardeinstellungen und -Bereiche für die Steuerungen des VIs AI Takt-Konfiguration auf.

Tabelle 18-2. Gerätespezifische Einstellungen und Bereiche für Kontrollen im VI AI Takt-Konfiguration

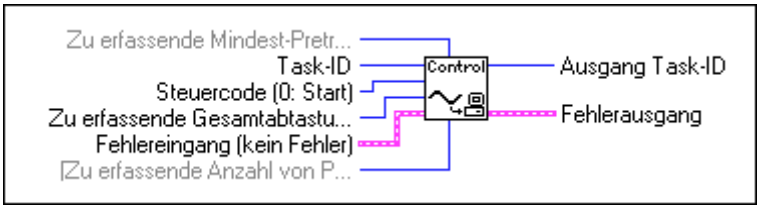
Gerät	Konfigurationsmodus		Retriggermodus	Welcher Takt		Taktgeber	
	Standardeinstellung	Bereich	Standardeinstellung	Standardeinstellung	Bereich	Standardeinstellung	Bereich
AT-MIO-16E1 AT-MIO-16E-2 AT-MIO-64E-1 NEC-MIO-16E-4 PCI-MIO-16E-1 PCI-MIO-16E-4 PCI-MIO-16XE-10	1	1; 3	nicht unterstützt	1	1; 2	1	1; 2 $4 \le n \le 11$
PCI-6110E PCI-6111E	1	1; 3	nicht unterstützt	1	1	1	1; 2 $4 \le n \le 11$
AT-MIO-16E-10 AT-MIO-16DE-10 AT-MIO-16XE-50 PCI-MIO-16XE-50	1	1; 3	nicht unterstützt	1	1; 2	1	1; 2 $4 \le n \le 9$
NB-A2150 NB-A2100 NB-A2000	1	1; 3	nicht unterstützt	1	1	1	$1 \le n \le 3$
DSA-Geräte	1	1; 3	nicht unterstützt	1	1	1	1
PC-LPM-16 DAQCard-500 DAQCard-516 DAQCard-700 Lab-PC	1	1; 3	nicht unterstützt	1	1; 2	1	1; 2
Lab-LC Lab-NB NB-MIO-16	1	1; 3	nicht unterstützt	1	2	1	1; 2
5102	1	1; 3	nicht unterstütztt	1	1	1	1; 6

Tabelle 18-2. Gerätespezifische Einstellungen und Bereiche für Kontrollen im VI AI Takt-Konfiguration (Fortsetzung)

Gerät	Konfigurationsmodus		Retriggermodus	Welcher Takt		Taktgeber	
	Standardeinstellung	Bereich	Standardeinstellung	Standardeinstellung	Bereich	Standardeinstellung	Bereich
5911, 5912	1	1; 3	nicht unterstützt	1	1; 2	1	$1 \le n \le 3$
Alle anderen Geräte	1	1; 3	nicht unterstützt	1	1; 2	1	$1 \le n \le 3$

AI Steuerung

Steuert die Analogeingabetasks und spezifiziert die zu erfassende Datenmenge.



☞ **Hinweis** ***Sie können dieses VI nicht benutzen, um eine Erfassung zu starten, wenn Sie ein PC-LPM-16, DAQCard-500 oder ein DAQCard-700-Gerät zum Abtasten mehrfacher SCXI-Kanäle im Multiplexmodus benutzen. Für diesen Sonderfall müssen Sie zur Datenerfassung das VI AI einzelner Scan benutzen. (Weitere Information über das VI AI einzelner Scan finden Sie in der entsprechenden Beschreibung in diesem Kapitel.) Sie können für ein Lab-Gerät und ein Gerät der Serie 1200, PC-LPM-16, DAQCard-500 oder DAQCard-700-Gerät jedoch das VI AI Steuerung benutzen, wenn Sie SCXI-Kanäle im Parallelmodus abtasten oder einen Einzel-SCXI-Kanal im Multiplexmodus abtasten. Sie können dieses VI für ein MIO-Gerät zur Abtastung von SCXI-Kanälen in jedem der beiden Modi benutzen.***

☞ **Hinweis** ***Nichtgepufferte Erfassungen werden für folgende Geräte nicht unterstützt.***

- **(Macintosh)** ***NB-A2000***
- **(Macintosh)** ***NB-A2100***
- **(Macintosh)** ***NB-A2150***

Tabelle 18-3 führt Standardeinstellungen und -bereiche für die Steuerungen des VIs AI Takt-Konfiguration auf.

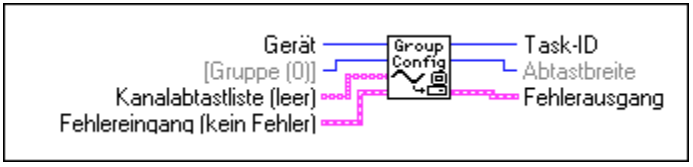
Tabelle 18-3. Gerätespezifische Einstellungen und Bereiche für das VI AI Steuerung

Gerät	Steuercode		Zu erfassende Gesamtabtast-ungen		Zu erfassende Mindest-Pretrig-ger-Abtastungen		Zu erfassende Anzahl von Puffern	
	SE*	B*	SE*	B*	SE*	B*	SE*	B*
NB-A2000 NB-A2150	0	0, 1, 4	0	0, *n*≥0	0	0, *n*≥3	1	*n*≥0
PC-LPM-16 DAQCard-500 DAQCard-700	0	0, 1, 4	0	0, *n*≥3	0	nicht unter-stützt	1	1
MIO-E-Serie	0	0, 1, 4	0	0, *n*≥3	0	0, *n*≥3	1	1
Gerät 5102	0	0, 1, 4	0	*n*≥0	0	*n*≥0	1	1
5911, 5912	0	0,4	0	*n*≥1	0	*n*≥0	1	1
Alle anderen Geräte	0	0, 1, 4	0	0, *n*≥3	0	*n*≥0	1	1

* SE= Standardeinstellung; B= Bereich

AI Gruppen-Konfiguration

Definiert, welche Kanäle zu einer Gruppe gehören und weist diese zu.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalbereiche und Abtastreihenfolgen.

Tabelle 18-4 führt Standardeinstellungen und -bereiche für die Steuerungen des VIs AI Grupenkonfiguration auf. In der ersten Reihe der Tabelle sind die Werte für die meisten Geräte angegeben, während in den anderen Reihen die Werte für die Geräte aufgeführt sind, die die Ausnahme zur Regel darstellen.

Tabelle 18-4. Gerätespezifische Einstellungen und Bereiche für das VI AI Gruppen-Konfiguration

Gerät	Gruppe		Kanalabtastliste	
	Standard-einstellung	Bereich	Standardein-stellung	Bereich
Die meisten Windows-Geräte	0	0≤*n*≤15	alle Kanäle	0≤*n*≤15
Die meisten Macintosh-Geräte	0	0≤*n*≤15	alle Kanäle	0≤*n*≤15
AT-MIO-64F-5 AT-MIO-64E-1*	0	0≤*n*≤15	alle Kanäle	0≤*n*≤63
Lab-PC+, PCI-1200, DAQCard-1200	0	0≤*n*≤15	alle Kanäle	0≤*n*≤7
Lab-LC, Lab-NB	0	0≤*n*≤15	alle Kanäle	0≤*n*≤7
NB-A2000, NB-A2150	0	0≤*n*≤15	alle Kanäle	0≤*n*≤3
NB-A2100	0	0≤*n*≤15	alle Kanäle	0, 1
Geräte 5102	0	0≤*n*≤15	alle Kanäle	0, 1
PCI-4452, PCI-4451	0	0≤*n*≤15	alle Kanäle	0, 1
PCI-4452, PCI-4552	0	0≤*n*≤15	alle Kanäle	0≤*n*≤3

* Die gültigen Kanäle für AT-MIO-64E-1 in Differentialmodus sind 0-7, 16-23, 34-39, 48-55.

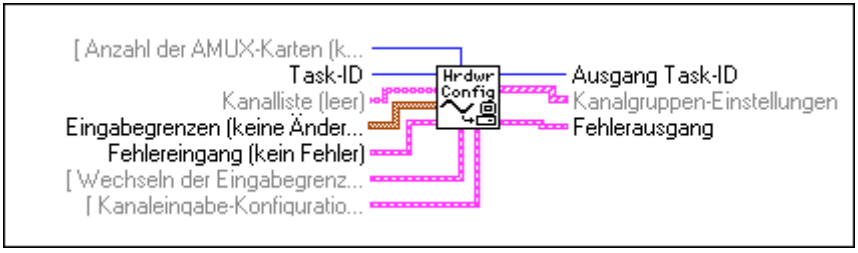
☞ **Hinweis** ***Lab-LC, Lab-NB, Lab-PC+, PCI-1200, PC-LPM-16, DAQCard-500, DAQCard-700 und DAQCard-1200 müssen Kanallisten, die Mehrfachkanäle enthalten, von Kanal n (n ≥ 0) bis Kanal 0 in sequentieller Folge abtasten, einschließlich aller Kanäle zwischen n und 0. NB-A2000, NB-A2150, EISA-A2000 und AT-A2150 erlauben nur folgende Abtastlisten: (0), (1), (2), (3), (0, 1), (2, 3) und (0, 1, 2, 3). NB-A2100 erlaubt folgende Abtastlisten: (0), (1), (0, 1) und (1, 0).***

Der oben angeführte Kanalabtastlistenbereich ist für den single-ended Modus. In Anhang B, DAQ-Hardware-Leistungsfähigkeit, finden Sie Angaben zur Festlegung der gültigen Bereiche für Kanäle im Differentialmodus.

SCXI-Module im Multiplexmodus müssen Kanäle in ansteigender fortlaufender Reihenfolge abtasten, wobei von jedem Kanal des Moduls aus gestartet werden kann. Die von Ihnen angegebene Modulreihenfolge kann arbiträr sein. SCXI-Module im Parallelmodus müssen die DAQ-Gerätebeschränkungen hinsichtlich der Reihenfolge der Kanalabtastlisten befolgen. Im Abschnitt *Kanal-, Port-, und Counter-Adressierung* in Kapitel 3, *Grundlegende Konzepte zur Datenerfassung*, im *Grundlagen der Datenerfassung mit LabVIEW* finden Sie Information über den SCXI-Kanalstringsyntax.

AI Hardware-Konfiguration

Konfigurieren Sie entweder die oberen oder unteren Eingabegrenzen, Polarität oder Verstärkung. Das VI AI Hardware-Konfiguration konfiguriert auch die Kopplung, den Eingabemodus und die Anzahl der AMUX-64T-Geräte. Das Konfigurations-Utility bestimmt die Standardeinstellung für die Parameter dieses VIs.



Sie können dieses VI zum Abruf derzeitiger Einstellungen benutzen, indem Sie es nur mit **Task-ID** oder mit **Task-ID** und **Kanalliste** verbinden. Wenn **Kanalliste** leer ist, konfiguriert das VI Kanäle auf einer *Basis pro Gruppe.* Dies bedeutet, daß die Konfiguration auf alle Kanäle der Gruppe zutrifft. Wenn Sie in der **Kanalliste** einen oder mehrere Kanäle angeben, konfiguriert das VI Kanäle auf *der Basis pro Kanal*. Dies bedeutet, daß die Konfiguration nur auf die von Ihnen angegebenen Kanäle zutrifft. Dieses VI gibt immer die derzeitigen Einstellungen für die gesamte Gruppe zurück.

Wenn die Konfiguration auf der Basis pro Kanal durchgeführt wird, kann die **Kanalliste** einen oder mehrere Kanäle enthalten. Die Kanäle in der **Kanalliste** müssen zu der von **TaskID** benannten Gruppe gehören. Die Kanäle werden auf dieselbe Art und Weise benannt wie für das VI AI Gruppen-Konfiguration. Wenn Sie für einen Kanal mehrere Abtastungen innerhalb einer Abtastung vornehmen und Sie die Hardware-Konfiguration für diesen Kanal bei jeder Abtastung ändern möchten, müssen Sie für jeden Fall die Einstellungen des Kanals innerhalb der Abtastung angeben. Wenn ein Element der **Kanalliste** mehr als einen Kanal angibt, trifft das entsprechende Element der anderen Arrays auf alle diese Kanäle zu.

Das VI verwendet die in den Konfigurationsarrays zuerst angegebenen Werte (**Obere Eingabewerte, untere Eingabewerte**, **Kopplung**, **Bereich**, **Polarität**, **Verstärkung** und **Modus**) auf den ersten Kanal in der Gruppe (wenn Sie die Konfiguration auf einer Basis pro Gruppe durchgeführt haben) oder auf die Kanäle in der **Kanalliste** (wenn Sie die Konfiguration auf einer Basis pro Kanal durchgeführt haben) folgendermaßen. Das VI verwendet die in den Arrays (bei Index 0) zuerst aufgeführten Werte für den ersten Kanal der Gruppe oder den(die) in Index 0 aufgeführten Kanal(Kanäle) der **Kanalliste**. Das VI verwendet die in den Konfigurationsarrays (in Index 1) an zweiter Stelle angegebenen Werte auf den zweiten Kanal der Gruppe oder auf den(die) in Index aufgeführten Kanal(Kanäle) der **Kanalliste**. Das VI verwendet die Werte auf diese Art, bis die Arrays erschöpft sind. Wenn Kanäle der Gruppe oder **Kanallisten** unkonfiguriert bleiben, verwendet das VI die letzten Werte im Array für alle verbleibenden unkonfigurierten Kanäle.

In Tabelle 18-5 finden Sie Beispiele für diese Methode. Der Parameter **Kanalabtastliste**, der Teil des VIs AI Gruppen-Konfiguration ist, wird in folgender Tabelle verwendet.

Tabelle 18-5. AI Hardware-Konfiguration Kanalkonfiguration

Konfigura-tionsbasis	Arraywerte	Ergebnisse
Gruppe	Gruppe **Kanalabtastliste** = 1; 3; 4; 5; 7 **Kanalliste** ist leer **untere Eingabegrenze** [0] = –1,0 **obere Eingabegrenze** [0] = +1,0	Alle Kanäle der Gruppe haben Eingabegrenzen von –1,0 bis +1,0.
Gruppe	Gruppe **Kanalabtastliste** = 1; 3; 4; 5; 7 **Kanalliste** ist leer **untere Eingabegrenze** [0] = –1,0 **obere Eingabegrenze** [0] = +1,0 **untere Eingabegrenze** [1] = 0,0 **obere Eingabegrenze** [1] = +5,0 **untere Eingabegrenze** [2] = –10,0 **obere Eingabegrenze** [2] = +10,0	Kanal 1 hat Eingabegrenzen von –1,0 bis +1,0. Kanal 3 hat Eingabegrenzen von 0,0 bis +5,0. Kanäle 4, 5 und 7 haben Eingabegrenzen von –10,0 bis +10,0.
Kanal	Gruppe **Kanalabtastliste** = 1; 3; 4; 5; 7 **Kanalliste** [0] = 1 **Kanalliste** [1] = 3:5 **untere Eingabegrenze** [0] = –1,0 **obere Eingabegrenze** [0] = +1,0	Kanäle 1, 3, 4 und 5 haben Eingabegrenzen von –1,0 bis +1,0. Bei Kanal 7 werden die Standard-Eingabegrenzen durch das Konfigurations-Utility festgelegt. Dies ist unverändert, weil es nicht in der **Kanalliste** enthalten ist.

Tabelle 18-5. AI Hardware-Konfiguration Kanalkonfiguration (Fortsetzung)

Konfigurationsbasis	Arraywerte	Ergebnisse
Kanal	Gruppe **Kanalabtastliste** = 1; 3; 4; 5; 7 **Kanalliste** [0] = 1 **Kanalliste** [1] = 3:5 **untere Eingabegrenze**[0] = –1,0 **obere Eingabegrenze**[0] = +1,0 **untere Eingabegrenze** [1] = 0,0 **obere Eingabegrenze** [1] = +5,0	Kanal 1 hat Eingabegrenzen von –1,0 bis +1,0. Kanäle 3, 4 und 5 haben Eingabegrenzen von 0,0 bis+5,0. Bei Kanal 7 werden die Standard-Eingabegrenzen durch die Konfigurations-Utility festgelegt.
Gruppe	Gruppe **Kanalabtastliste** = 0; 1; 0; 1 **Kanalliste** ist leer **untere Eingabegrenze** [0] = –1,0 **obere Eingabegrenze** [0] = +1,0 **untere Eingabegrenze** [1] = –1,0 **obere Eingabegrenze** [1] = +1.0 **untere Eingabegrenze** [2] = –10,0 **obere Eingabegrenze** [2] = +10,0 **untere Eingabegrenze** [3] = –10,0 **obere Eingabegrenze**[3] = +10,0	Kanäle 0 und 1 haben Eingabegrenzen von –1,0 bis +1,0 bei der ersten Abtastung und von –10,0 bis +10,0 bei der zweiten Abtastung.

Die unteren und oberen Eingabegrenzen werden durch **Bereich**, **Polarität** und **Verstärkung** bestimmt. Wenn Sie gültige Eingabegrenzarrays (d.h. Arrays mit einer Länge von mehr als Null) verbinden, wählt das VI geeignete Eingabebereiche, -polaritäten und -verstärkungen aus, um diese **Eingabegrenzen** zu erreichen. Das VI ignoriert die Arrays **Bereich**, **Polarität** und **Verstärkung**.

Wenn Sie die Eingabegrenzarrays nicht verbinden, prüft das VI **Bereich**, **Polarität** und **Verstärkung**. Wenn das VI ein Array findet, setzt es die entsprechende Eingabeeigenschaft auf die Werte im Array. Wenn das VI kein Array findet, läßt es die entsprechenden Eingabeeigenschaften unverändert.

Bei einigen Geräten und SCXI-Modulen werden **Bereich**, **Polarität** und/oder **Verstärkung** durch Onboard-Jumper festgesetzt. LabVIEW verändert die Einstellungen von rangierten Parametern nicht, wenn Sie **Eingabegrenzen** angeben. Wenn LabVIEW mit den rangierten aktuellen Einstellungen die gewünschten **Eingabegrenzen** nicht erreicht, sendet es eine Warnung aus.

Um die derzeitigen Jumperwerte zu umgehen, müssen Sie das VI AI Hardware-Konfiguration aufrufen und **Bereich**, **Polarität** und/oder **Verstärkung** ausdrücklich angeben. Die anfängliche Einstellung der Parameter (Standardwert ist die werkseitige Jumpereinstellung) wird durch das Konfigurations-Utility bestimmt.

Wenn ein Paar Eingabegrenzwerte beide `0` sind, ändert das VI die **Eingabegrenzen** nicht.

SCXI-Kanal-Hardware-Konfigurationen sind eigentlich eine Kombination eines SCXI-Moduls und von DAQ-Geräteeinstellungen, wofür besondere Überlegungen notwendig sind. Durch die Art und Weise, wie Sie die Angabe von Kanälen vornehmen, wird angezeigt, ob LabVIEW die SCXI-Moduleinstellungen und/oder DAQ-Geräteeinstellungen ändert. Der Eingabegrenzenparameter gilt immer auf den gesamten Erfassungspfad.

Wenn Sie die Konfiguration auf einer Basis pro Gruppe vornehmen, kann LabVIEW sowohl das SCXI-Modul als auch die DAQ-Geräteeinstellungen ändern. In diesem Falle gilt **Verstärkung** für den gesamten Pfad und ist das Produkt der SCXI-Kanalverstärkung und Erfassungsgeräte-Kanalverstärkung. LabVIEW stellt die höchste am SCXI-Modul benötigte Verstärkung ein und fügt dann, wenn nötig, die DAQ-Geräteverstärkung hinzu.

Wenn Sie die Konfiguration auf einer Basis pro Kanal vornehmen, können Sie die Kanäle auf eine von drei Arten bezeichnen. Erstens können Sie, wie in folgendem Beispiel gezeigt, den gesamten Pfad angeben.

```
OB0!SC1!MD1!CH0:7
```

Sie können auch, wie in folgendem Beispiel gezeigt, die Pfadbezeichnung unter Benutzung der im DAQ Channel Wizard konfigurierten Kanalbezeichnungen angeben.

```
temperature
```

Wenn Sie eine dieser Methoden anwenden, kann LabVIEW sowohl die SCXI-Einstellungen als auch die DAQ-Geräteeinstellungen ändern, wobei **Verstärkung** auf das Produkt der SCXI-Kanalverstärkung und der DAQ-Geräteverstärkung anwendbar ist. LabVIEW stellt die höchste, im SCXI-Modul benötigte Verstärkung ein und fügt dann, wenn nötig, die DAQ-Geräteverstärkung hinzu.

Die zweite Methode spezifiziert, wie in folgendem Beispiel gezeigt, nur den SCXI-Kanal anzugeben.

```
SC1!MD1!CH0:7
```

Diese Angabe zeigt an, daß LabVIEW nur die SCXI-Einstellungen ändern soll. Darüber hinaus gilt **Verstärkung** nur für den SCXI-Kanal.

Die dritte Methode ist, wie in folgendem Beispiel gezeigt, nur den Erfassungsgerätekanal anzugeben.

```
OB0
```

In diesem Fall ändert LabVIEW nur die DAQ-Geräteeinstellungen. Der Verstärkungsparameter ist nur auf den Onboard-Kanal anwendbar.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalbereiche, Eingabegrenzen und Abtastreihenfolgen.

In den Tabelles 18-6 bis 18-9 werden Standardeinstellungen und -bereiche für das VI AI Hardware-Konfiguration aufgeführt. Eine Tilde (~) zeigt an, daß der Parameter nur auf einer Basis pro Gruppe konfiguriert werden kann. Dies bedeutet, daß Sie ihn nicht pro Kanal konfigurieren können. In der ersten Reihe dieser Tabellen werden die Werte für die meisten Geräte angegeben, in den anderen Reihen die Werte für die Geräte, die die Ausnahmen zur Regel darstellen. Wenn Sie die Standardeinstellungen nicht mit dem Konfigurations-Utility vorgenommen haben, benutzen Sie die in diesen Tabellen aufgeführten Standardeinstellungen.

Tabelle 18-6. Gerätespezifische Einstellungen und Bereiche für das VI AI Hardware-Konfiguration

Geräte	Kanaleingabe Konfigurationscluster				Anzahl der AMUX		Kanalliste
	Kopplung		Eingabemodus~				
	SE*	B*	SE*	B*	SE*	B*	SE*
Die meisten Geräte	1	1	1	1 ≤*n*≤3	0	0 ≤*n*≤4	leer
NB-A2000	2	1; 2	2	2	0	0	leer
PC-LPM-16, Lab-LC, Lab-NB	1	1	2	2	0	0	leer
Lab und Geräte der 1200-Serie	1	1	2	1 ≤*n*≤3	0	0	leer
AT-MIO-16X, AT-MIO-64F-5	1	1	1 (nein ~)	1 ≤*n*≤3	0	0 ≤*n*≤4	leer
NB-A2100, NB-A2150	1	1; 2	2	2	0	0	leer
DAQCard-500, DAQCard-516, DAQCard-700	1	1	2	1, 2	0	0	leer

Tabelle 18-6. Gerätespezifische Einstellungen und Bereiche für das VI AI Hardware-Konfiguration (Fortsetzung)

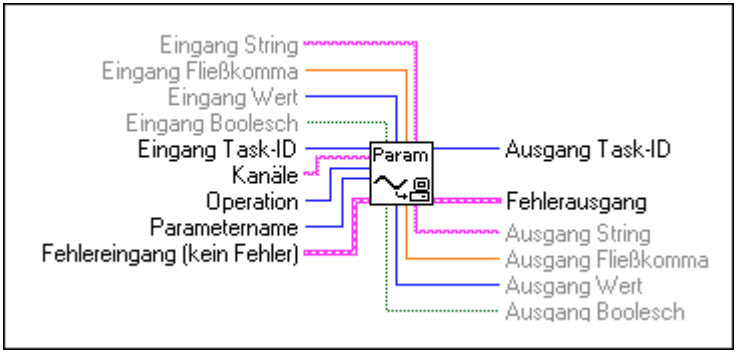
Geräte	Kanaleingabe Konfigurationscluster				Anzahl der AMUX		Kanalliste
	Kopplung		Eingabemodus~				
	SE*	B*	SE*	B*	SE*	B*	SE*
Geräte 5102	1	1; 2	2	2	0	0	leer
PCI-6110E, PCI-6111E, PCI-4451, PCI-4551, PCI-4452, PCI-4552	1	1; 2	1	1	0	0	leer

* SE= Standardeinstellung; B= Bereich

☞ **Hinweis** ***Die Kanäle 0 und 1 sowie 2 und 3 müssen für NB-A2150 über dieselbe Kopplung verfügen.***

AI Parameter

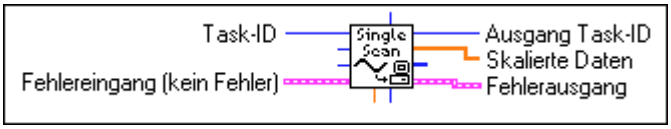
Konfiguriert verschiedene mit dem Analogeingang einer Geräteoperation zusammenhängende Parameter, die nicht von anderen AI-VIs abgedeckt sind und ruft diese ab.



AI einzelner Scan

Gibt eine Datenabtastung zurück. Wenn Sie eine Erfassung mit dem VI AI Steuerung gestartet haben, liest dieses VI eine Abtastung der Daten vom internen Puffer. Auf dem Macintosh und in Windows liest das VI vom Onboard-FIFO, wenn die Erfassung nicht gepuffert ist. Wenn Sie keine Erfassung gestartet haben, so startet dieses VI die Erfassung,

ruft eine Datenabtastung ab und beendet dann die Erfassung. Die Gruppenkonfiguration legt die Kanäle fest, die das VI abtastet. Die Geräte 5102, DSA und 59xx werden von diesem VI nicht unterstützt.



Wenn Sie das VI AI Steuerung nicht aufrufen, löst dieses VI eine einzelne Abtastung aus und benutzt dafür die schnellste und sicherste Taktrate. Sie können die Kanaltaktrate jedoch mit dem VI AI Takt-Konfiguration ändern.

Wenn Sie das VI AI Steuerung mit einem auf `0` (Start) eingestellten **Steuercode** starten, werden die Abtastungen durch ein Taktsignal initiiert.

Wenn Sie extern getaktete Umwandlungen möchten, müssen Sie das VI AI Takt-Konfiguration verwenden, um den Taktgeber auf extern einzustellen.

Wenn die Taktgeber intern sind und Sie keinen Speicherplatz zuweisen, beginnt eine getaktete, nicht gepufferte Erfassung, wenn Sie das VI AI Steuerung mit dem auf `0` eingestellten **Steuercode** ausführen. Diese Erfassungsart ist für die Synchronisation von Analogeingaben und -ausgaben in einer Punkt-zu-Punkt Steueranwendung nützlich. Folgende Geräte unterstützen getaktete, nicht gepufferte Erfassungen nicht:

- **(Macintosh)** Lab-NB, Lab-LC, NB-A2000, NB-A2100 und NB-A2150

☞ **Hinweis** ***Im Falle eines FIFO-Überlaufs während einer getakteten, nichtgepufferten Erfassung startet LabVIEW das Gerät neu.***

In Tabelle 18-7 sind Standardeinstellungen und Bereiche für das VI AI einzelner Scan aufgeführt.

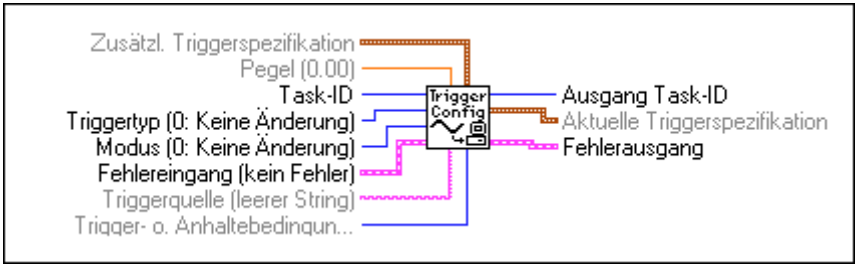
Tabelle 18-7. Gerätespezifische Einstellungen und Bereiche für das VI AI einzelner Scan

Gerät	Ausgabetyp		Op-Code		Zeitgrenze	
	SE	B	SE	B	SE	B
NB-A2000, NB-A2100, NB-A2150	1	$1 \le n \le 3$	1	1	Variable	$n \ge 0$
Alle anderen Geräte	1	$1 \le n \le 3$	1	$1 \le n \le 4$	$1 \le n \le 4$	$n \ge 0$

* SE= Standardeinstellung; B= Bereich

AI Trigger-Konfiguration

Konfiguriert die Triggerbedingungen für den Start der Abtastung, des Kanaltakts und des Abtastzählers.



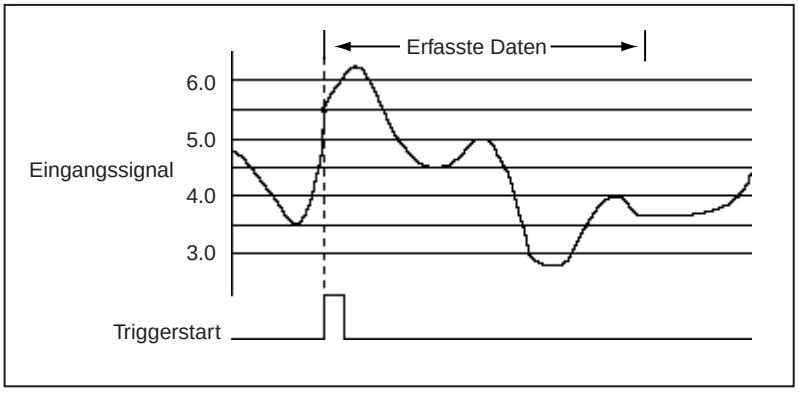
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie Informationen zu den für Ihr DAQ-Gerät verfügbaren Trigger. Eine detaillierte Beschreibung zu den Triggerfähigkeiten Ihres Geräts befindet sich im Benutzerhandbuch für Ihr Gerät der E-Serie.

Nachstehend finden Sie eine Beschreibung der Triggertypen 1 (Analogtrigger), 2 (Digitaltrigger A) und 3 (Digitaltrigger B) in der Reihenfolge, wie Sie auf drei Erfassungstypen anwendbar sind: Posttrigger, Pretrigger mit Software-Start und Pretrigger mit Hardware-Start. Die anderen Triggertypen werden am Ende dieses Abschnitts besprochen.

Anwendungsstyp 1: Posttriggered Erfassung (nur Trigger starten)

Wenn die **zu erfassenden Gesamtabtastungen** ≥ 0 und **die zu erfassenden Pretrigger-Abtastungen** 0 sind, führen Sie eine Posttrigger-Erfassung durch. Der **Triggertyp** 1 oder 2 (Analogtrigger bzw. Digitaltrigger A) startet die Erfassung (Digitaltrigger B ist illegal). Sie stellen einen Starttrigger bereit. In 18-10, Teil 2 und 3, finden Sie Angaben zur Bestimmung des Standardpins, an den Sie Ihr Triggersignal anschließen. An einigen Geräten können Sie mittels des **Triggerquellen**-Parameters eine alternative Quelle angeben.

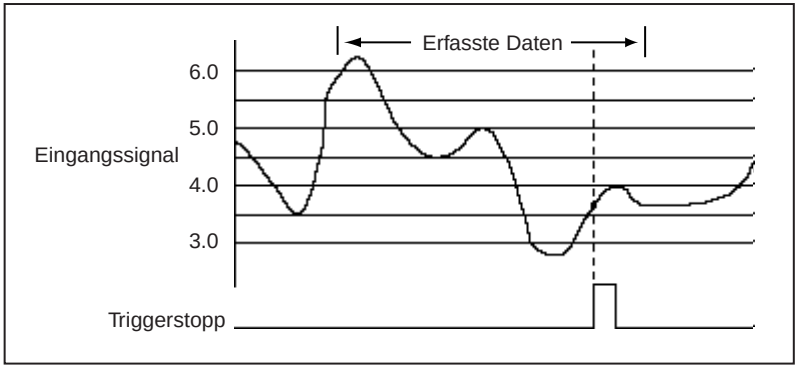
Wenn Sie einen Analogtrigger benutzen und das Analogsignal mit einem der Analogeingangskanäle verbunden ist, muß dieser Kanal bei Geräten der E-Serie auf der Abtastliste an erster Stelle sein. Diese Beschränkung hat keine Gültigkeit, wenn Sie das Analogsignal mit PFI0 verbinden.



In obiger Darstellung beträgt die **Gesamtzahl der zu erfassenden Abtastungen** `1000` und die der **zu erfassenden Pretrigger-Abtastungen** `0`. Der Starttrigger kann vom Digitaltrigger A kommen (**Trigger- oder Anhaltebedingung** =`1`: Trigger an einer steigenden Flanke oder Steigung, **Ebene** = `5,5`, **Fenstergröße**= `0,2`).

Anwendungstyp 2: Pretriggered Erfassung (Für alle Trigger -Typen)

Wenn die **zu erfassenden Gesamtabtastungen** und die **zu erfassenden Pretrigger-Abtastungen** beide > 0 sind, wird die Erfassung von Posttrigger-Daten durch einen **Triggertyp** von `1` oder `2` gestartet (Analog- bzw. Digitaltrigger A), nachdem die Pretrigger-Daten erfaßt wurden. Der Trigger wird als *Stopptrigger* bezeichnet, weil die Erfassung erst bei Auftreten des Triggers gestoppt wird. Die Erfassung wird durch ein Software-Strobe gestartet. Dabei handelt es sich um eine Software-Start-Pretrigger-Erfassung. Sie stellen den Stopptrigger bereit. In 18-10, Teile 2 und 3, finden Sie Angaben zur Bestimmung des Standardpins, an den Sie Ihr Triggersignal anschließen. An einigen Geräten können Sie mittels des **Triggerquellen**-Parameters eine alternative Quelle angeben.



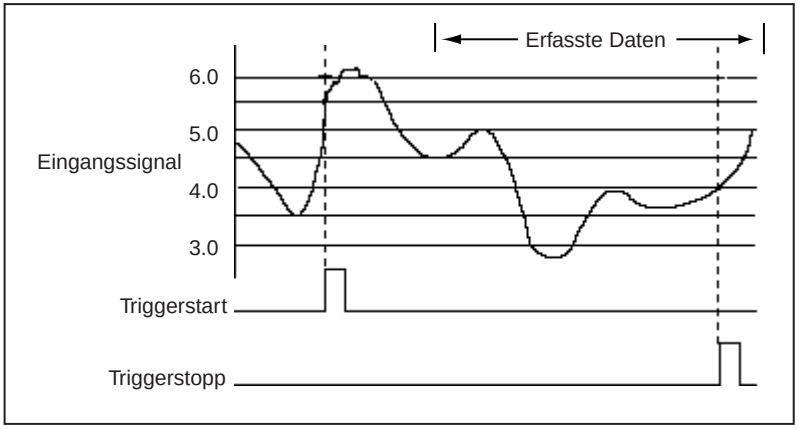
In vorheriger Darstellung beträgt die **Gesamtzahl der zu erfassenden Abtastungen** `1000` und die der **zu erfassenden Pretrigger-Abtastungen** `900`. Der Stopptrigger kann vom Digitaltrigger A oder einem Analogtrigger kommen (**Trigger- oder Anhaltebedingung** =`1`: Trigger an einer steigenden Flanke oder Steigung, **Ebene** = `3,7`, **Fenstergröße**= `0,5`).

Wenn Sie einen Analogtrigger benutzen und das Analogsignal mit einem Analogeingangskanal verbunden ist, muß dieser Kanal bei Geräten der E-Serie der einzige auf der Abtastliste befindliche Kanal sein (Mehrfachkanal-Abtastung nicht gestattet). Diese Beschränkung hat keine Gültigkeit, wenn Sie das Analogsignal mit PFI0 verbinden.

Anwendungstyp 3: Pretriggered Erfassung (Start- und Stopptrigger)

Anwendungstyp 3 wird nicht häufig benutzt. Wenn Sie nicht vorhaben, sowohl einen Start- als auch einen Stopptrigger bereitzustellen, können Sie diesen Abschnitt auslassen.

An MIO-Geräten können Sie sowohl den Start- wie auch den Stopptrigger aktivieren. (Sie müssen hierfür das VI AI Trigger-Konfiguration zweimal aufrufen.) In diesem Fall wird die Erfassung von einem Digital- oder Analogtriggersignal und nicht von einem Software-Strobe gestartet. Dabei handelt es sich um eine Hardware-Start-Pretriggered-Erfassung. Sie stellen den Starttrigger (wie in Anwendungstyp 1 beschrieben) und den Stopptrigger (wie in Anwendungstyp 2 beschrieben) bereit. In Tabelle 18-11 und Tabelle 18-12 finden Sie Angaben zur Bestimmung des Standardpins, an den Sie Ihr Triggersignal anschließen. An einigen Geräten können Sie mittels des **Triggerquellen**-Parameters eine alternative Quelle angeben.



In obiger Darstellung beträgt die **Gesamtzahl der zu erfassenden Abtastungen** `1000` und die der **zu erfassenden Pretrigger-Abtastungen** `900`. Der Starttrigger kann vom Digitaltrigger B oder einem Analogtrigger kommen (**Trigger- oder Anhaltebedingung** =`1`: Trigger an einer steigenden Flanke oder Steigung, **Ebene** = `5,5`, **Fenstergröße**= `0,2`). Der Stopptrigger kann vom Digitaltrigger A oder einem Analogtrigger kommen (**Trigger- oder**

Anhaltebedingung =1: Trigger an einer steigenden Flanke oder Steigung, **Ebene** = 4,0, **Fenstergröße**= 0,2). Bitte beachten Sie, daß einige der Daten nach dem Starttrigger zurückgewiesen wurden, da alle 900 Pretrigger-Scans gesammelt wurden und der Stopptrigger mehr als 900 Scans entfernt ist.

Wenn Sie bei Geräten der E-Serie Analogtrigger benutzen, kommen, wie in Tabelle 18-8 gezeigt, verschiedene Beschränkungen zur Anwendung.

Tabelle 18-8. Beschränkungen für die Anwendung von Analogtriggern an Geräten der E-Serie

Start-trigger	Stop-trigger	Beschränkungen
Digital A	Digital B	Keine
Digital B	Analog	Das Analogsignal muß mit dem PFI0 verbunden sein, außer, Sie tasten nur einen Kanal ab. In diesem Fall kann der Eingang zu diesem Kanal benutzt werden.
Analog	Digital A	Das Analogsignal muß auf der Abtastliste an erster Stelle stehen, wenn es mit einem Analogeingangskanal verbunden ist.

Ein **Triggertyp** von 4 (digitale Abtasttakt-Ausblendung) versetzt ein externes TTL-Signal in die Lage, den Abtasttakt ein- und auszublenden, wodurch die Erfassung auf effiziente Weise angehalten und wieder aufgenommen wird.

Beim NB-MIO-16 sind Kanaltakt und Abtasttakt dasselbe. Wenn die Abtasttakt-Ausblendung FALSE wird, kann die derzeitige Abtastung deshalb nicht abgeschlossen werden und der Abtasttakt stellt die Operation ein. Wenn die Abtasttakt-Ausblendung TRUE wird, nimmt der Abtasttakt die Operation sofort an der Stelle wieder auf, an der diese vorher abgebrochen wurde. Verbinden Sie Ihr Signal mit dem EXTGATE-Pin.

Ein **Triggertyp** von 5 (Analogabtasttakt-Ausblendung) versetzt ein externes Analogsignal in die Lage, den Abtasttakt ein- und auszublenden, wodurch die Erfassung auf effiziente Weise angehalten und wieder aufgenommen wird. Ein Triggertyp von 6 erlaubt Ihnen, die Ausgabe der Analogtriggerschaltung (ATCOUT) als Allzwecksignal zu benutzen. Sie können z.B. ATCOUT zum Start einer Analogausgangs-Operation benutzen oder Sie können die Anzahl der in ATCOUT erscheinenden Analogtrigger zählen.

☞ **Hinweis** ***Triggertypen 1, 5 und 6 von Geräten der E-Serie benutzen dieselbe Analogtrigger-Schaltung. Es können alle drei Typen zur selben Zeit aktiviert sein, wobei der zuletzt aktivierte Trigger bestimmt, wie sich die Triggerschaltung verhält. Die in den Triggeranwendungen beschriebenen Beschränkungen zur E-Serie gelten für alle drei Triggertypen.***

Triggertyp 5 bei Geräten der E-Serie benutzt die digitale Abtasttakt-Ausblendung und die Analogtrigger-Schaltung. Deshalb macht die Aktivierung des Triggertyps 5 alle für den Triggertyp 4 vorgenommen Einstellungen unwirksam.

Einige Geräte unterstützen die Benutzung von Digitaltriggern, wobei aber bei diesen Geräten die Quelle vorher festgelegt wird. Deshalb ist der **Triggerquellen**parameter ungültig. In Tabelle 18-9 werden die Pinbezeichnungen am I/O-Anschluß aufgeführt, mit dem Sie Ihr Digitaltrigger-Signal verbinden müssen.

Tabelle 18-9. Digitaltrigger-Quellen für Geräte mit feststehenden Digitaltrigger-Quellen

Gerät	Posttriggering	Pretriggerung	
	Anfangs-Triggerpin	Anfangs-Triggerpin	End-Triggerpin
MIO-16L/H, MIO-16DL/DH	STARTTRIG*	STARTTRIG*	STOPTRIG
NB-MIO-16L/H	STARTTRIG*	nicht unterstützt	nicht unterstützt
AT-MIO-16X, AT-MIO-16F-5, AT-MIO-64F-5	EXTTRIG*	EXTTRIG*	EXTTRIG*
Lab-Geräte und Geräte der Serie 1200	EXTTRIG	nicht unterstützt	EXTTRIG
PC-LPM-16, DAQCard-500, DAQCard-700	nicht unterstützt	nicht unterstützt	nicht unterstützt
NB-A2000, NB-A2100, NB-A2150	EXTTRIG*	nicht unterstützt	EXTTRIG*
DSA 45xx	EXTTRIG*	EXTTRIG*	EXTTRIG*

* Bei den Geräten AT-MIO-16X, AT-MIO-16F-5 und AT-MIO-64F-5, wird derselbe Pin für den Anfangs- und den Endtrigger benutzt. Weitere Details finden Sie in Ihrem Hardware-Benutzerhandbuch.

In 18-10 sind die Standardeinstellungen und Bereiche für das VI AI Trigger-Konfiguration aufgeführt. In der ersten Reihe dieser Tabellen werden die Werte für die meisten Geräte angegeben, in den anderen Reihen die Werte für die Geräte, die die Ausnahmen zur Regel darstellen.

Tabelle 18-10. Gerätespezifische Einstellungen und Bereiche für das VI AI Trigger-Konfiguration - Teil 1

Gerät	Triggertyp		Modus		Trigger- oder Haltebedingung		Ebene	
	SE*	B*	SE*	B*	SE*	B*	SE*	B*
Die meisten Geräte	2	2, 3	1	1≤*n*≤3	nicht unterstützt		nicht unterstützt	
AT-MIO-16E-10, AT-MIO-16DE-10, AT-MIO-16XE-50, PCI-MIO-16XE-50	2	2≤*n*≤4	1	1≤*n*≤3	1	1; 2; 7; 8	nicht unterstützt	
AT-MIO-16E-2, AT-MIO-64E-3, NEC-MIO-16E-4	2	1≤*n*≤6	1	1≤*n*≤3	1	1≤*n*≤8	0	–10 ≤*n*≤ 10
Lab-Geräte und Geräte der Serie 1200	2	2	1	1≤*n*≤3	nicht unterstützt		nicht unterstützt	
PC-LPM-16, DAQCard-500, DAQCard-700	nicht unterstützt		nicht unterstützt		nicht unterstützt		nicht unterstütztt	
NB-A2100, NB-A2150	1	1; 2	1	1≤*n*≤3	1	1; 2	0	–2,828 ≤*n*≤ 2,828
NB-A2000	1	1; 2	1	1≤*n*≤3	1	1; 2	0	–5.12 ≤*n*≤ 5,12
5102 Geräte	1	1; 2; 3; 6	1	1≤*n*≤3	1	1; 2; 3; 4	0	–5≤*n*≤5
5911, 5912	1	1; 2; 3; 6	1	1≤*n*≤3	1	1; 2; 3; 4	0	–10≤*n*≤ 10
DSA-Geräte	1	1; 2; 3	1	1≤*n*≤3	1	1≤*n*≤4	0	–42≤*n*≤ 42

*SE = Standardeinstellung; B= Bereich

Tabelle 18-11. Gerätespezifische Einstellungen und Bereiche für das VI AI Trigger-Konfiguration - Teil 2

Gerät	Trigger-Quelle (Analog)		Zusätzliche Trigger-Spezifikationscluster			
			Fenstergröße		Kopplung	
	Standard-einstellung	Bereich	Standard-einstellung	Bereich	Standard-einstellung	Bereich
AT-MIO-16E-1 AT-MIO-16E-2 NEC-MIO-16E-4 PCI-MIO-16E-1 PCI-MIO-16E-1 PCI-MIO-16XE-10	0	0≤n≤15, PFI0	0	0≤n≤20	nicht unterstützt	
AT-MIO-64E-3	0	0≤n≤63, PFI0	0	0≤n≤20	nicht unterstützt	
NB-A2000	0	0 ≤*n*≤ 3	nicht unterstützt		2	1; 2
NB-A2100 NB-A2150	0	0 ≤*n* ≤3	0	0 ≤*n*≤ 5,656	1	1; 2
5102 Geräte	0	1, 1, TRIG	0	0 ≤*n*≤ 10	1	1; 2
PCI-6110E	0	0 ≤*n*≤ 4 PFI0	0	0 ≤*n*≤ 80	1	1; 2
PCI-6111E	0	0 ≤*n*≤ 2 PFI0	0	0 ≤*n*≤ 80	1	1; 2
4451, 4551	0	0, 1	0	0 ≤*n*≤ 84	nicht unterstützt	
4452, 4552	0	0 ≤*n*≤ 4	0	0 ≤*n*≤ 84	nicht unterstützt	
Alle anderen Geräte	nicht unterstützt		nicht unterstützt		nicht unterstützt	

Tabelle 18-12. Gerätespezifische Einstellungen und Bereiche für das VI AI Trigger-Konfiguration- Teil 3

Gerät	Trigger-Quelle (Digital)	
	SE	B
Starttrigger E-Serie	PFI0	PFI 0~9, RTSI 0~6, GPCTR0
Stoptrigger E-Serie	PFI1	PFI 0~9, RTSI 0~6
Digitale Abtasttakt-Ausblendung E-Serie	PFI0	PFI 0~9, RTSI 0~6
5102-Geräte mit RTSI, Start- und Stopptrigger	PFI0	PFI 1-2, RTSI 0-6
5102-Geräte ohne RTSI, Start- und Stopptrigger	PFI0	PFI1-2
5911, 5912	PFI1	PFI 1-2, RTSI 0-6
DSA 44xx Starttrigger	PFI0	PFI0, PFI1, PFI3, PFI4, PFI6, RTSI 0~6
OSA 44xx Stopptrigger	PFI1	PFI0, PFI1, PFI3, PFI4, PFI6, RTSI 0~6
DSA 45xx Start- und Stopptrigger	dedizierter EXTTRIG*-Pin	PFI 0~33, RTSI 0~6
Alle anderen Geräte	nicht unterstützt*	

* Siehe 18-9 für Geräte mit festgelegten digitalen Trigger-Quellen.

Tabelle 18-13. Gerätespezifische Einstellungen und Bereiche für das VI AI Trigger-Konfiguration- Teil 4

Gerät	Zusätzliche Trigger-Spezifikationscluster					
	Verzögerung		Skip-Zählung		Zeitbe-grenzung	
	SE	B	SE	B	SE	B
NB-A2000	0	0≤*n*≤655,35	nicht unterstützt		nicht unterstützt	
NB-A2100, NB-A2150S	0	0≤*n*≤32,77	nicht unterstützt		nicht unterstützt	
NB-A2150C	0	0≤*n*≤16,38	nicht unterstützt		nicht unterstützt	
NB-A2150F	0	0≤*n*≤17,05	nicht unterstützt		nicht unterstützt	
Alle anderen Geräte	nicht unterstützt		nicht unterstützt		nicht unterstützt	

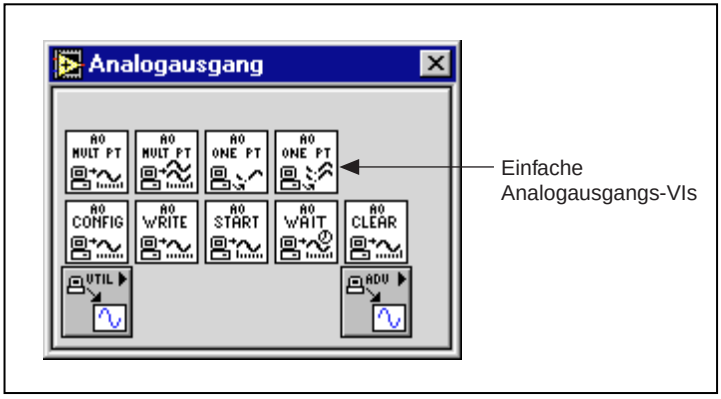
*SE = Standardeinstellung; B= Bereich

19

Einfache Analogausgangs-VIs

In diesem Kapitel werden Einfache Analogausgangs-VIs beschrieben, die einfache Analogausgangsoperationen durchführen. Sie können diese VIs vom Frontpanel aus durchführen oder sie als SubVIs in Grundanwendungen einsetzen.

Sie können auf die Einfachen Analogausgangs-VIs durch Auswahl von **Funktionen»Datenerfassung»Analogausgang** zugreifen. Bei den Einfachen Analogausgangs-VIs handelt es sich, wie nachstehend beschrieben, um die VIs der obersten Reihe der Palette **Analogausgang**.

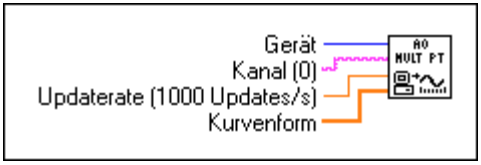


Beschreibungen der Einfachen Analogausgangs-VIs

Folgende Einfache Analogausgangs-VIs sind verfügbar.

AO Signalverlauf erzeugen

Erzeugt auf einem Analogausgangskanal mit der angegebenen Update-Rate eine Spannungskurvenform.



Das VI AO Signalverlauf erzeugt auf einem angegebenen Analogausgangskanal eine Mehrpunkt-Kurvenform. Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, das VI zu stoppen oder fortzusetzen.

AO Signalverläufe erzeugen

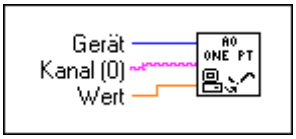
Erzeugt Mehrfach-Kurvenformen auf einem angegebenen Analogausgangskanal mit der angegebenen Updaterate.

Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, das VI zu stoppen oder fortzusetzen.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalnummern und Anzahl der Kanäle.

AO Update-Kanal

Schreibt einen angegebenen Wert auf einen Analogausgangskanal.

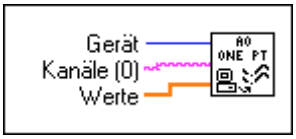


Das VI AO Update-Kanal schreibt ein einzelnes Update auf einen Analogausgangskanal. Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, das VI zu stoppen oder fortzusetzen.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalnummern und Ausgabegrenzen.

AO Update-Kanäle

Schreibt Werte auf jeden der angegebenen Analogausgangskanäle.



Das VI AO Ausgangskanäle aktualisiert Mehrfach-Analogausgangskanäle mit Einzelwerten.

Bei Auftreten eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Möglichkeit haben, das VI zu stoppen oder fortzusetzen.

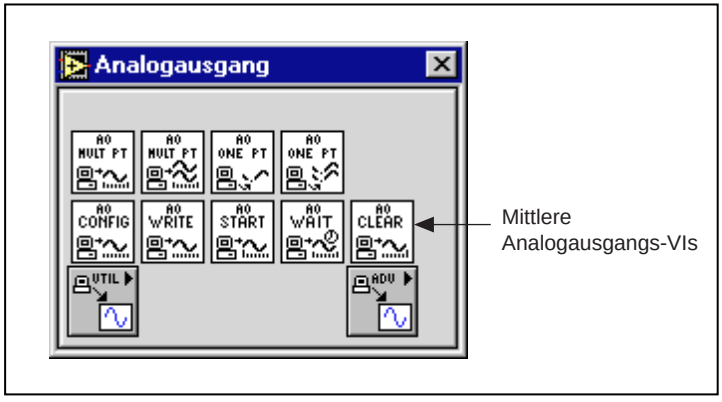
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die Kanalnummern, die Sie für Ihr DAQ-Gerät benutzen können.

20

Mittlere Analogausgangs-VIs

In diesem Kapitel werden die Mittleren Analogausgangs-VIs beschrieben. Diese VIs sind Einzel-VI-Lösungen für häufig vorkommende Analogausgangsprobleme. Die VIs der mittleren Stufe sind benutzerfreundlich; es mangelt ihnen jedoch an Flexibilität. Da alle in diesem Kapitel besprochenen VIs auf der fortgeschrittenen Stufe beruhen, können Sie hinsichtlich weiterer Informationen zu den Ein- und Ausgängen und deren Funktionen in Kapitel 22, *Fortgeschrittene Analogausgangs-VIs*, nachlesen.

Sie können auf die Mittleren Analogausgangs-VIs durch Auswahl von **Funktionen»Datenerfassung»Analogausgang** zugreifen. Die Mittleren Analogausgangs-VIs sind, wie nachstehend dargestellt, die VIs in der zweiten Reihe der Palette **Analogausgang.**



Fehlerbehandlung

LabVIEW macht die Fehlerbehandlung mit den Mittleren Analogeingangs-VIs einfach. Jedes VI der mittleren Stufe hat einen Eingangscluster **Fehlereingang** und einen Ausgangscluster **Fehlerausgang.** Die Cluster enthalten einen Booleschen Wert, wodurch angezeigt wird, ob ein Fehler vorgekommen ist, sowie den Fehlercode für den Fehler und den Namen des VIs, das den Fehler zurückgegeben hat. Wenn **Fehlereingang** einen Fehler anzeigt, gibt das VI die Fehlerinformation an **Fehlerausgang** zurück und stellt die Ausführung ein.

☞ **Hinweis** ***Das VI AO zurücksetzen stellt eine Ausnahme zu dieser Regel dar - dieses VI löscht die Erfassung immer, und zwar unabhängig davon, ob in Fehlereingang ein Fehler angezeigt wird.***

Wenn Sie eines der Mittleren Analogausgangs-VIs in einer While-Schleife benutzen, sollten Sie die Schleife stoppen, wenn der **Status** im Cluster **Fehlerausgang** TRUE anzeigt. Wenn Sie den Fehlercluster mit dem Allgemeinen Error-Handler-VI verbinden, entziffert das VI die Fehlerinformation und beschreibt den Fehler.

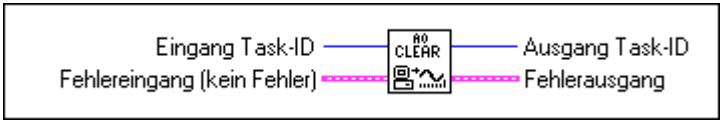
Das Allgemeine Error-Handler-VI befindet sich in **Funktionen**»**Zeit und Dialog** in LabVIEW. Weitere Informationen zu diesem VI finden Sie in Ihrem *LabVIEW Benutzerhandbuch*.

Beschreibungen der Analogausgangs-VIs

Folgende Analogausgangs-VIs sind verfügbar.

AO zurücksetzen

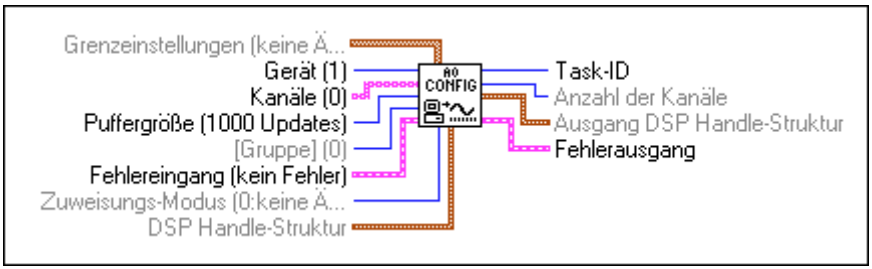
Setzt alle mit **Task-ID** in Zusammenhang stehenden Ausgangstasks zurück.



Das VI AO zurücksetzen beendet die Erzeugung unabhängig davon, ob **Fehlereingang** einen Fehler anzeigt.

AO konfigurieren

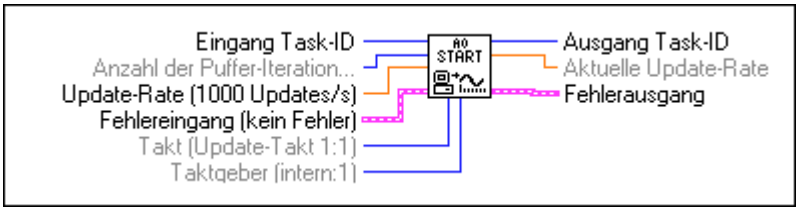
Konfiguriert die Kanalliste und Ausgabegrenzen und weist der Analogausgabe-Operation einen Puffer zu.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalbereiche und Ausgabegrenzen.

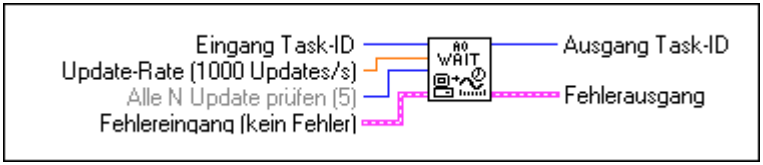
AO starten

Startet die gepufferte Analogausgangs-Operation. Dieses VI stellt die Updaterate ein und beginnt dann mit der Erzeugung.



AO warten

Wartet, bis die Signalerzeugung des Tasks abgeschlossen ist.

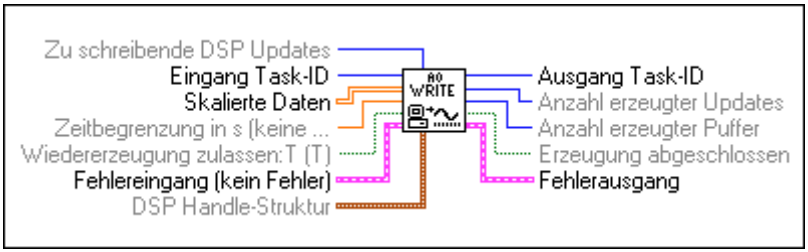


Benutzen Sie das VI AO warten, um auf die Beendigung einer gepufferten, endlichen Signalerzeugung zu warten, ehe Sie das VI AO zurücksetzen aufrufen. Das VI AO warten prüft den Status des Tasks in regelmäßigen Intervallen, indem es das VI AO schreiben aufruft und dessen Ausgabe **Erzeugung abgeschlossen** überprüft. Das AO warten wartet zwischen den Intervallen asynchron, um den Prozessor für andere Operationen freizumachen. Das VI errechnet das Warteintervall, indem es die Eingabe **Alle N Update prüfen** durch die

Update-Rate teilt. Sie dürfen das VI AO warten nicht benutzen, wenn Sie ständig Daten erzeugen, da die Erzeugung nie abgeschlossen wird. Das VI AO zurücksetzen stoppt eine kontinuierliche Signalerzeugung.

AO schreiben

Schreibt Daten für eine gepufferte Analog-Ausgangsoperation in den Puffer.

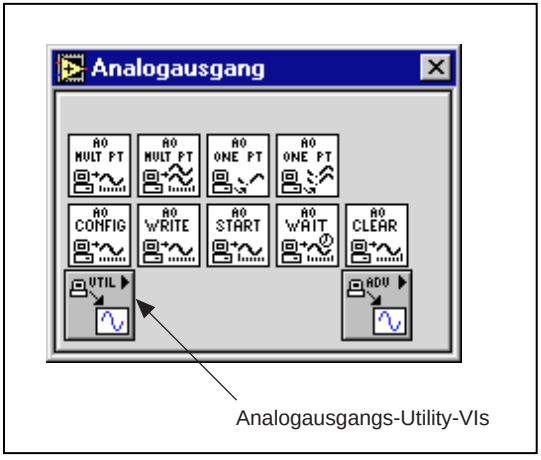


21

Analogausgang-Utility-VIs

Dieses Kapital beschreibt die Analogausgang-Utility-VIs. Diese VIs sind Einzel-VI-Lösungen für häufig vorkommende Analogausgangsprobleme. Die Analogausgang-Utility-VIs sind VIs mittlerer Stufe, die somit auf den VIs Fortgeschrittener Stufe beruhen. In Kapitel 22, *Fortgeschrittene Analogausgangs-VIs*, finden Sie zusätzliche Informationen zu Ein- und Ausgängen und deren Funktionen.

Sie können auf die Palette **Analogausgang-Utility** durch Auswahl von **Funktionen**»**Datenerfassung**»**Analogausgang**»**Analogausgang-Utilities** zugreifen. Das Icon, das Sie auswählen müssen, um auf die Analogausgang-Utility-VIs zugreifen zu können, befindet sich, wie nachstehend dargestellt, in der untersten Reihe der Palette **Analogausgang.**



Fehlerbehandlung

LabVIEW vereinfacht die Fehlerbehandlung durch die Mittleren Analogausgang-Utility-VIs. Jedes VI mittlerer Stufe hat einen Eingangscluster **Fehlereingang** und einen Ausgangscluster **Fehlerausgang**. Die Cluster enthalten ein Boolesch, wodurch angezeigt wird, ob ein Fehler vorgekommen ist, sowie den Fehlercode für den Fehler und den Namen des VIs, das den Fehler zurückgegeben hat. Wenn in **Fehlereingang** ein Fehler angezeigt wird, gibt das VI die Fehlerinformation an **Fehlerausgang** zurück und stellt die Ausführung ein.

Wenn Sie eines der Analogausgang-VIs in einer While-Schleife benutzen, sollten Sie die Schleife stoppen, wenn der **Status** im Cluster **Fehlerausgang** TRUE anzeigt. Wenn Sie den Fehlercluster mit dem Allgemeinen Error-Handler-VI verbinden, entziffert das VI die Fehlerinformation und beschreibt den Fehler.

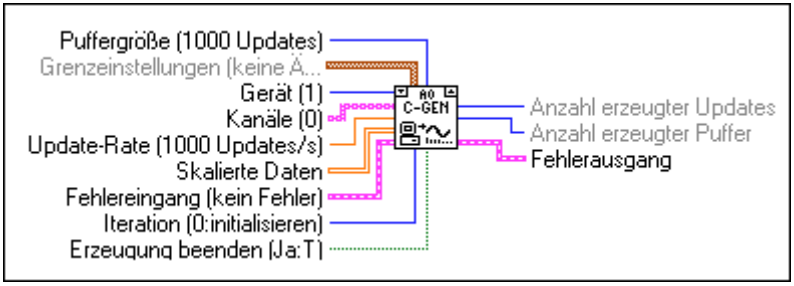
Das Allgemeine Error-Handler-VI befindet sich in **Funktionen**»**Zeit und Dialog** in LabVIEW. Weitere Informationen zu diesem VI finden Sie in Kapitel 10, *Zeit-, Dialog- und Fehlerfunktionen*.

Beschreibungen der Analogausgang-Utility-VIs

Folgende Analogausgang-Utility-VIs sind verfügbar.

AO kontinuierlich erzeugen

Erzeugt mit der angegebenen Update-Rate eine kontinuierliche, getaktete, kreisförmig gepufferte Kurvenform für den jeweiligen Ausgabekanal. Das VI führt während der Datenerzeugung ein kontinuierliches Update des Ausgabepuffers durch. Wenn Sie dieselben Daten kontinuierlich erzeugen möchten, können Sie statt dessen das VI AO Signalverlauf erzeugen benutzen.



Iterations-
terminal

Benutzen Sie das VI AO kontinuierlich erzeugen, wenn sich Ihre Signalverlaufsdaten auf einer Disk befinden und zu umfangreich sind, um sie im Arbeitsspeicher zu behalten, oder wenn Sie den Signalverlauf in Echtzeit erstellen müssen. Geben Sie das VI in eine While-Schleife, und verbinden Sie es mit dem Iterationsterminal des VIs Iterationseingabe.

☞ **Hinweis** ***Wenn Ihr Programm mehr als 2^{31}–1 Mal iteriert, verbinden Sie dieses VI Iterationsterminal nicht mit der Schleife Iterationsterminal. Setzen Sie die* Iteration *statt dessen auf der ersten Schleife auf* 0 *und dann bei allen anderen Iterationen auf einen beliebigen positiven Wert. Das VI rekonfiguriert und startet neu, wenn die* Iteration ≤*0 ist.***

Verbinden Sie auch die Bedingung, die die Schleife abschließt, mit dem VI-Eingang **Erfassung beenden**, invertieren Sie, wenn nötig, das Signal, so daß es bei der letzten Iteration TRUE anzeigt. Bei der Iteration 0 ruft das VI das VI AO konfigurieren auf, um die Kanalgruppe und Hardware zu konfigurieren, und es weist einen Datenpuffer zu. Es ruft auch das VI AO schreiben auf, um die jeweiligen Daten in einen Puffer zu schreiben; dann ruft es das VI AO starten auf, um die Update-Rate einzustellen und die Signalerzeugung zu starten. Bei jeder nachfolgenden Iteration ruft das VI das VI AI lesen auf, um die nächste Datenportion an der derzeitigen Schreibposition in den Puffer zu schreiben. Bei der letzten Iteration (wenn **Erfassung beenden** TRUE ist) oder bei Vorkommen eines Fehlers, ruft das VI das VI beenden auf, um alle momentan laufenden Erfassungen zu beenden. Obwohl dies normalerweise nicht notwendig ist, können Sie das VI AO kontinuierlich erzeugen außerhalb einer Schleife aufrufen (d.h., es wird nur einmal aufgerufen). Wenn Sie dies tun, müssen Sie jedoch die Eingaben **Iteration** und **Erzeugung beenden** unverbunden lassen.

Der erste Aufruf an das VI AO schreiben stellt **Wiedererzeugung zulassen** auf TRUE, so daß dieselben Daten mehr als einmal erzeugt werden können. Wenn Sie die Einstellung von **Wiedererzeugung zulassen** auf FALSE ändern, müssen Sie neue Daten so schnell schreiben, daß neue Daten immer zur Erzeugung verfügbar sind. Wenn der Puffer nicht schnell genug gefüllt wird, erhalten Sie einen Erzeugungsfehler. Um dieses Problem zu korrigieren, müssen Sie die **Updaterate** verringern, die **Puffergröße** vergrößern, das bei jedem Mal geschriebene Datenvolumen vergrößern oder Daten öfters schreiben.

Wenn Sie **Wiedererzeugung zulassen** auf FALSE einstellen und Ihr Gerät ein Analogausgang-FIFO hat, muß Ihre **Puffergröße** mindestens zweimal so groß sein wie Ihr FIFO.

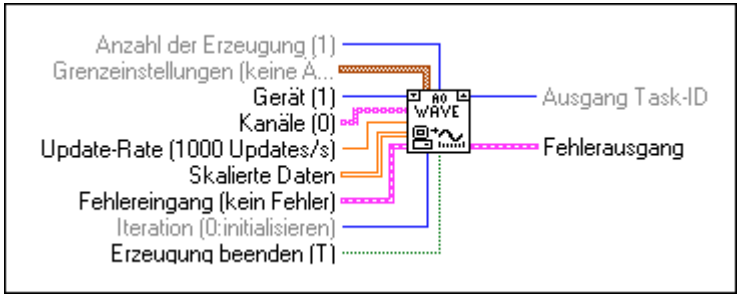
Bei Auftreten eines Fehlers ruft das VI das VI AO beenden auf, um die gerade laufende Erzeugung zu beenden, und gibt dann die unveränderte Fehlerinformation an **Fehlerausgang** weiter. Wenn innerhalb des VIs AO kontinuierliche Erzeugung ein Fehler vorkommt, beendet das VI AO beenden die gerade durchgeführte Erzeugung und gibt deren Fehlermeldung weiter.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalbereiche und Ausgabegrenzen.

☞ **Hinweis** ***Das VI AO kontinuierliche Erzeugung benutzt ein uninitialisiertes Shiftregister als lokalen Speicher, um sich zwischen den Aufrufen an die Task-ID der Ausgangsoperation zu erinnern. Normalerweise benutzen Sie dieses VI an einer Stelle in einem Diagramm. Wenn Sie es jedoch an mehreren Stellen benutzen, teilen sich die Mehrfachvorkommnisse des VIs dieselbe Task-ID. Alle Aufrufe an dieses VI konfigurieren, schreiben Daten oder beenden dieselbe Erzeugung. Gelegentlich ziehen Sie es eventuell vor, dieses VI an mehreren Stellen des Diagramms zu verwenden. Dabei sollte sich aber jedes Vorkommnis auf eine unterschiedliche Task-ID beziehen (wenn Sie z.B. Signalverläufe gleichzeitig mit zwei Geräten erzeugen möchten). Speichern Sie eine Kopie dieses VIs unter einem neuen Namen (z.B. AO kontinuierliche Erzeugung R), und machen Sie Ihr neues VI ablaufinvariant.***

AO Signalverlauf erzeugen

Erzeugt einen getakteten, einfach-gepufferten oder kreisförmig gepufferten Signalverlauf für die jeweiligen Ausgangskanäle mit der angegebenen Update-Rate. Außer, wenn Sie eine unendliche Erzeugung durchführen, gibt das VI die Steuerung an das LabVIEW-Diagramm zurück, wenn die Erzeugung abgeschlossen ist.



Iterations-terminal

Wenn Sie dieses VI zur Erzeugung von Mehrfach-Signalverläufen mit derselben Kanalgruppe in einer Schleife plazieren, müssen Sie das Iterationsterminal mit dem VI Iterationseingang verbinden.

☞ **Hinweis** ***Wenn Ihr Programm mehr als 2^{31}–1 Mal iteriert, dürfen Sie dieses VI-Iterationsterminal nicht mit dem Schleifen-Iterationsterminal verbinden. Setzen Sie statt dessen den Iterationswert bei der ersten Schleife auf*** `0` ***und dann bei allen anderen Iterationen auf einen positiven Wert. Das VI rekonfiguriert und startet neu, wenn die Iteration ≤0 ist.***

Bei Iteration 0 ruft das VI zur Konfiguration der Kanalgruppe und der Hardware und zur Zuweisung der Puffer für die Daten das VI AO konfigurieren auf. Bei jeder Iteration ruft das VI zum Schreiben der Daten in den Puffer das VI AO schreiben auf, dann zur Einstellung der Update-Rate und zum Start der Erzeugung das VI AO starten. Wenn Sie das VI AO Signalverlauf erzeugen nur einmal aufrufen, können **Iteration** unverbunden bleiben. Der Iterationsparameter wird standardmäßig auf `0` eingestellt, wodurch dem VI mitgeteilt wird, das Gerät vor dem Start der Signalverlauferzeugung zu konfigurieren

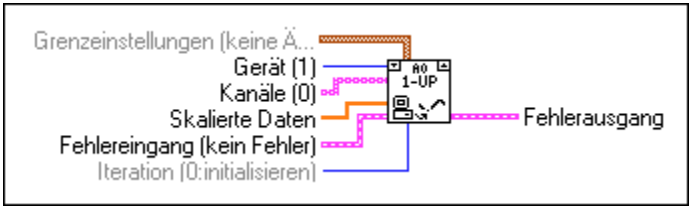
Bei Vorkommen eines Fehlers ruft das VI das VI AO zurücksetzen auf, um alle gerade laufenden Erzeugungen zu beenden, und gibt die Fehlerinformation dann unverändert an **Fehlerausgang** weiter. Wenn innerhalb des VIs AO Signalverlauf erzeugen ein Fehler vorkommt, beendet das VI die gerade laufende Erzeugung und gibt die Fehlerinformation aus.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie für Ihr DAQ-Gerät verfügbare Kanalbereiche, Ausgabegrenzen und Abtastreihenfolgen.

☞ **Hinweis** ***Das VI AO kontinuierliche Erzeugung benutzt ein uninitialisiertes Shiftregister als lokalen Speicher, um sich zwischen den Aufrufen an die Task-ID der Ausgangsoperation zu erinnern. Normalerweise benutzen Sie dieses VI an einer Stelle in einem Diagramm. Wenn Sie es jedoch an mehreren Stellen benutzen, teilen sich die Mehrfachvorkommnisse des VIs dieselbe Task-ID. Alle Aufrufe an dieses VI konfigurieren, schreiben Daten oder beenden dieselbe Erzeugung. Gelegentlich ziehen Sie es eventuell vor, dieses VI an mehreren Stellen des Diagramms zu verwenden, dabei jedoch jedes Vorkommnis auf eine unterschiedliche Task-ID beziehen zu lassen. Speichern Sie eine Kopie des VIs unter einem neuen Namen (z.B. AO kontinuierliche Erzeugung R), und machen Sie Ihr neues VI ablaufinvariant.***

AO ein Update schreiben

Schreibt an jeden der angegebenen Analogausgang-Kanäle einen einzigen Wert.



Das VI AO ein Update schreiben führt ein sofortiges, ungetaktetes Update einer Gruppe von einem oder mehreren Kanälen durch. Wenn Sie das VI zum Schreiben von mehr als einem Wert an dieselbe Kanalgruppe in eine Schleife geben, müssen Sie das Iterationsterminal **Iteration**seingang verbinden.

☞ **Hinweis** ***Wenn Ihr Programm mehr als $2^{31}-1$ Mal iteriert, verbinden Sie dieses VI Iterationsterminal nicht mit der Schleife Iterationsterminal. Setzen Sie die Iteration statt dessen auf der ersten Schleife auf*** `0` ***und dann bei allen anderen Iterationen auf einen beliebigen positiven Wert. Das VI rekonfiguriert und startet neu, wenn die Iteration ≤0 ist.***

Bei einer Iteration von 0 ruft das VI zur Konfiguration der Kanalgruppe und der Hardware das VI AO konfigurieren auf und dann das VI AO ein Update, um die Spannung an die Ausgangskanäle zu schreiben. Bei zukünftigen Iterationen ruft das VI nur das VI AO ein Update auf und vermeidet so unnötige Konfigurationen. Wenn Sie das VI AO ein Update schreiben nur einmal aufrufen, um zu jedem Kanal einen einzelnen Wert zu schreiben, können Sie die **Iteration** unverbunden lassen. Der Standardwert von 0 sagt dem VI, die Konfiguration vor dem Schreiben jeglicher Daten durchzuführen.

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalbereiche, Ausgabegrenzen und Abtastreihenfolgen.

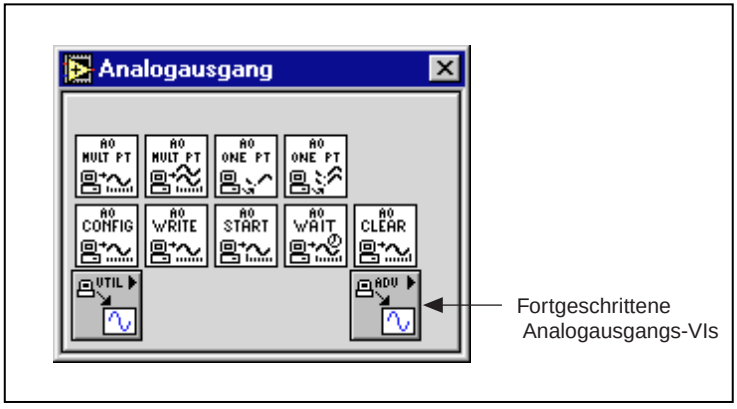
☞ **Hinweis** ***Das VI AO ein Update schreiben benutzt ein nicht formatiertes Shiftregister als lokalen Speicher, um sich an das Task-ID für die Kanalgruppe zu erinnern, wenn Aufrufe zwischen den VIs erfolgen. Normalerweise erscheint dieses VI an einer Stelle in Ihrem Diagramm. Wenn Sie es jedoch an mehreren Stellen benutzen, teilen sich die Mehrfachvorkommnisse des VIs dieselbe Task-ID. Alle Aufrufe an dieses VI konfigurieren oder schreiben Daten an dieselbe Gruppe. Wenn Sie dieses VI an mehr als einer Stelle des Diagramms verwenden möchten und sich jedes Vorkommnis auf ein unterschiedliches TaskID beziehen soll (wenn Sie z.B. Daten gleichzeitig mit zwei Geräten schreiben möchten), speichern Sie eine Kopie des VIs unter einem neuen Namen (z.B. AO ein Update schreiben), und machen Sie Ihr neues VI ablaufinvariant.***

22

Fortgeschrittene Analogausgangs-VIs

Dieses Kapitel enthält Referenzbeschreibungen der Fortgeschrittenen Analogausgangs-VIs. Diese VIs bilden die Schnittstelle zur NI-DAQ-Software und sind die Grundlage für Einfache-, Utility- und Mittlere Analogausgangs-VIs.

Sie können auf die Palette **Fortgeschrittener Analogeingang** durch Auswahl von **Funktionen»Datenerfassung»Analogausgang» Fortgeschrittener Analogausgang** zugreifen. Das Icon, das Sie für den Zugriff auf die Fortgeschrittenen Analogausgangs-VIs auswählen müssen, befindet sich, wie nachstehend dargestellt, in der untersten Reihe der Palette **Analogausgang.**



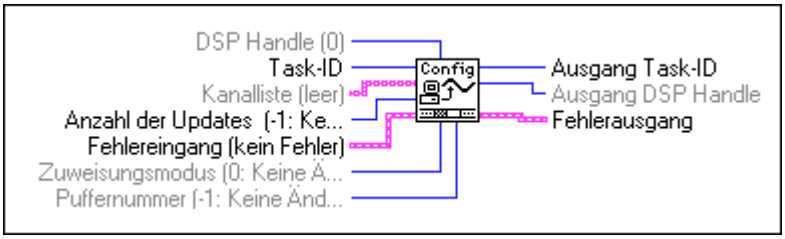
Beschreibungen der Fortgeschrittenen Analogausgangs-VIs

Folgende Fortgeschrittene Analogausgangs-VIs sind verfügbar.

AO Puffer-Konfiguration

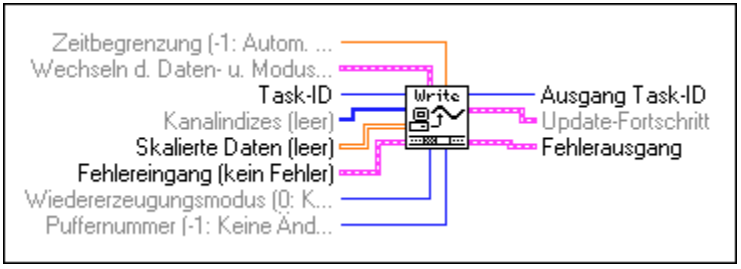
Weist LabVIEW Speicherplatz zur Speicherung von Analogausgangspuffern zu. Wenn Sie Interrupte benutzen, können Sie durch mehrfachen Aufruf des VIs AO Puffer-Konfiguration eine Reihe von Analogausgangspuffern zuweisen. Jeder Puffer kann seine eigene Größe haben. Wenn Sie DMA (direkter Speicherzugriff) benutzen, brauchen Sie nur einen Puffer zuweisen.

Benutzen Sie die Nummer, die Sie diesem Puffer mit diesem VI zuweisen, wenn Sie sich wegen anderer VIs auf diesen Puffer beziehen müssen.



AO in Puffer schreiben

Schreibt Analogausgangsdaten in die Puffer, die vom VI AO Puffer-Konfiguration erstellt wurden.



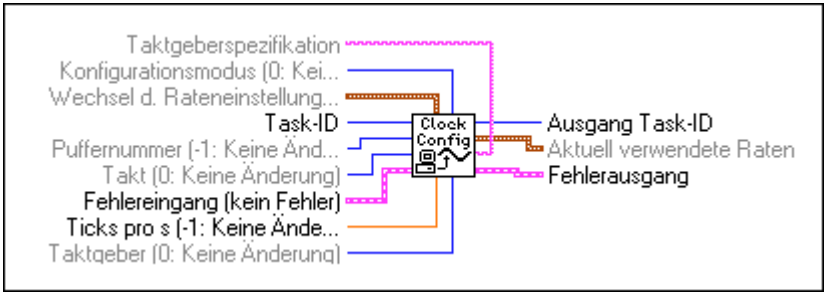
Sie können die neuen Daten mit einem von drei Eingaben verbinden -**skalierte Daten, Binärdaten** oder **DSP-Speicher-Handle**. Das VI durchsucht diese Eingaben in dieser Reihenfolge nach dem ersten Array mit einer Länge, die größer als Null ist. Das VI schreibt die Daten von diesem Array dann in einen Ausgangspuffer. Die Anzahl der Updates, die das VI schreibt, wird von der Länge der Arrays **skalierte Daten** oder **Binärdaten** bestimmt. Wenn **DSP-Speicher-Handle** auf die Datenquelle zeigt, müssen die **zu schreibenden Updates** anzeigen, wieviele Updates das VI schreiben soll. Wenn keine verbundenen Daten

übertragen werden, ist dieses VI für Fortschrittsinformationen über das Update trotzdem nützlich.

Die Gesamtzahl der vor dem Start zu einem Puffer geschriebenen Updates kann geringer sein als die Zahl der Updates, die Sie dem Puffer beim Aufruf des VIs AO Puffer-Konfiguration zugewiesen haben. LabVIEW erzeugt nur die zum Puffer geschriebenen Updates.

AO Takt-Konfiguration

Konfiguriert ein Update oder einen Intervalltakt für die Analogausgabe.

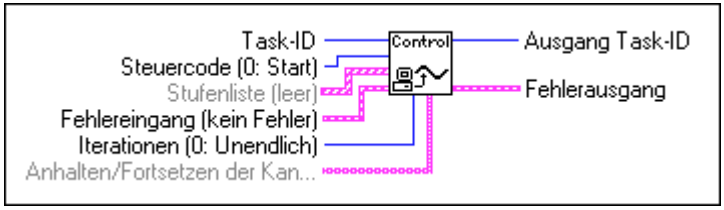


In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Takte.

Sie können Taktraten auf drei verschiedene Arten ausdrücken - durch **Ticks pro Sekunde**, **Sekunden pro Tick** oder durch die drei Zeitbasisparameter. Das VI durchsucht diese Parameter in dieser Reihenfolge und bringt Taktraten des ersten Parameters mit einem verbundenen gültigen Eingang zum Ausdruck. Wenn Sie einen Updatetakt konfigurieren, ist ein Tick gleich einem Intervall.

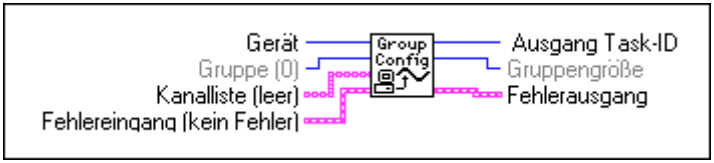
AO Steuerung

Startet und beendet Analogausgangstasks, hält diese an und nimmt sie wieder auf.



AO Gruppen-Konfiguration

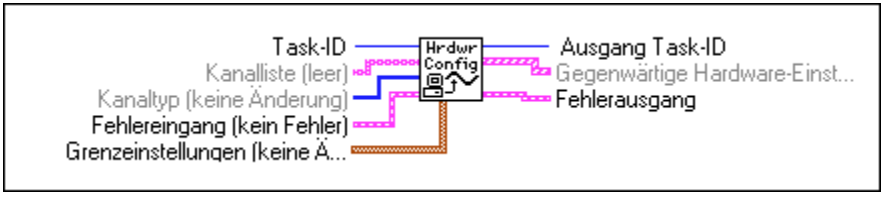
Weist einer Gruppennummer eine Liste von Analogausgangskanälen zu und produziert die Task-ID, das von allen anderen Analogausgangs-VIs benutzt wird.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanäle.

AO Hardware-Konfiguration

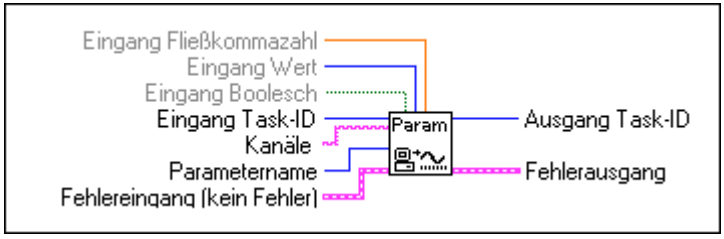
Konfiguriert die Grenzen (Polarität und Referenz) und ob Daten für einen jeweiligen Kanal in Volt-Milliampere ausgedrückt werden, wenn Sie Kanalnummern benutzen. Dieses VI gibt immer die aktuellen Einstellungen für die Kanäle in der Gruppe zurück.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Kanalbereiche und Ausgangsgrenzen.

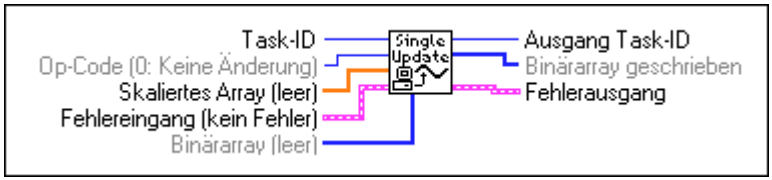
AO Parameter

Stellt verschiedene mit der Analogausgangs-Operation des Geräts verbundene Parameter ein, die von anderen Analogausgangs-VIs nicht abgedeckt werden.



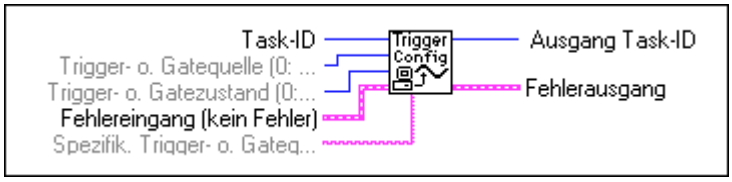
AO ein Update

Führt für die Kanäle der Gruppe ein sofortiges Update durch.



AO Trigger- und Gate-Konfiguration (Windows)

Konfiguriert die Trigger- und Gate-Bedingungen für Analogausgangs-Operationen an Geräten der E-Serie und an 5411-Geräten.

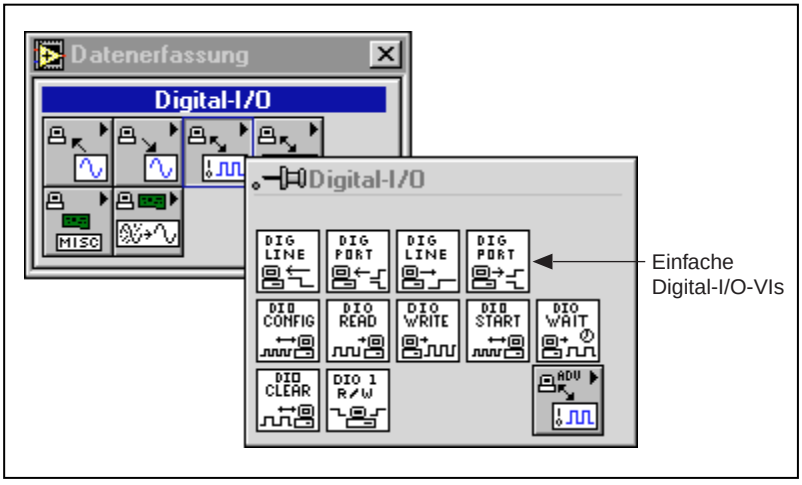


23

Einfache Digital-I/O-VIs

Dieses Kapitel beschreibt die Einfachen Digital-I/O-VIs, die einfache Digital-I/O-Operationen durchführen. Sie können diese VIs vom Frontpanel aus ausführen oder sie als Grundanwendungen in SubVIs einsetzen.

Sie können auf die Einfachen Digital-I/O-VIs durch Auswahl von **Funktionen**»**Datenerfassung**»**Digital-I/O** zugreifen.



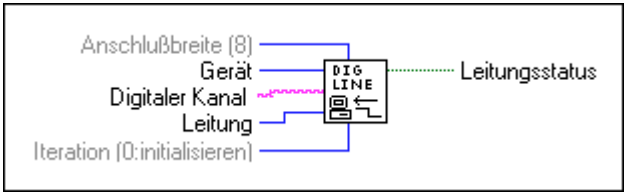
Bei den Einfachen Digital-I/O-VIs handelt es sich um die VIs in der obersten Reihe der Palette **Digital-I/O**. Beispiele für die Anwendung von Einfachen Digital-I/O-VIs finden Sie in der Beispielsbibliothek durch Öffnen von `examples\daq\digital\digital.llb`.

Beschreibungen der Einfachen Digital-I/Os

Folgende Einfache Digital-I/Os sind verfügbar.

Von digitaler Leitung lesen

Liest den logischen Zustand einer digitalen Leitung an einem von Ihnen konfigurierten digitalen Kanal.

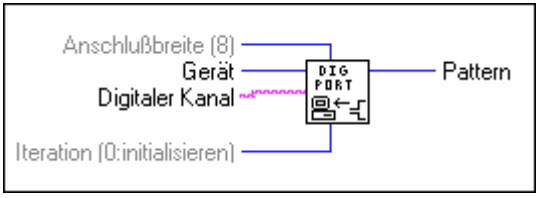


Bei Vorkommen eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Wahl treffen können, das VI zu stoppen oder fortzusetzen.

☞ **Hinweis** ***Wenn Sie dieses VI an einem Digital-I/O-Anschluß aufrufen, der Teil eines 8255 PPI ist, wenn Ihr Iterationsterminal auf 0 bleibt, durchläuft Ihr 8255 PPI eine Konfigurationsphase, bei der alle Anschlüsse innerhalb desselben PPI-Chips unabhängig von der Datenrichtung auf Niedriglogik eingestellt werden. Die Datenrichtungen an anderen Anschlüssen wird jedoch aufrechterhalten. Um diesen Effekt zu vermeiden, können Sie einen anderen Wert als 0 mit dem Iterationsterminal verbinden, sobald Sie die gewünschten Anschlüsse konfiguriert haben.***

Auf digitale Anschluß lesen

Liest den von Ihnen konfigurierten digitalen Kanal.

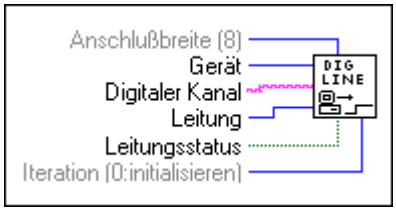


Bei Vorkommen eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Wahl treffen können, das VI zu stoppen oder fortzusetzen.

☞ **Hinweis** ***Wenn Sie dieses VI auf einem Digital-I/O-Anschluß als Teil eines 8255 PPI aufrufen, wenn Ihr Iterationsterminal auf 0 bleibt, durchläuft Ihr 8255 PPI eine Konfigurationsphase, bei der alle Anschlüsse innerhalb desselben PPI-Chips unabhängig von der Datenrichtung auf Niedriglogik eingestellt werden. Die Datenrichtungen an anderen Anschlüssen wird jedoch aufrechterhalten. Um diesen Effekt zu vermeiden, können Sie einen anderen Wert als 0 mit dem Iterationsterminal verbinden, sobald Sie die gewünschten Anschlüsse konfiguriert haben.***

Zu digitaler Leitung schreiben

Stellt den Ausgangs-Logikstatus der digitalen Leitung auf einem von Ihnen angegebenen digitalen Kanal hoch oder niedrig ein.

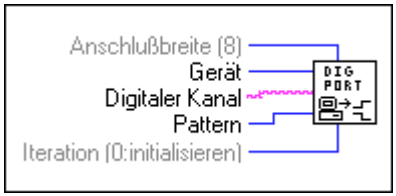


Bei Vorkommen eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Wahl treffen können, das VI zu stoppen oder fortzusetzen.

☞ **Hinweis** ***Wenn Sie dieses VI auf einem Digital-I/O-Anschluß aufrufen, der Teil eines 8255 PPI ist, wenn Ihr Iterationsterminal auf 0 bleibt, durchläuft Ihr 8255 PPI eine Konfigurationsphase, bei der alle Anschlüsse innerhalb desselben PPI-Chips unabhängig von der Datenrichtung auf Niedriglogik eingestellt werden. Die Datenrichtungen an anderen Anschlüssen werden jedoch aufrechterhalten. Um diesen Effekt zu vermeiden, können Sie einen anderen Wert als 0 mit dem Iterationsterminal verbinden, sobald Sie die gewünschten Anschlüsse konfiguriert haben.***

Zu digitalem Anschluß schreiben

Gibt ein Dezimalmuster an einen von Ihnen angegebenen digitalen Kanal aus.



Bei Vorkommen eines Fehlers wird ein Dialogfeld angezeigt, wodurch Sie die Wahl treffen können, das VI zu stoppen oder fortzusetzen.

☞ **Hinweis** ***Wenn Sie dieses VI auf einem Digital-I/O-Anschluß aufrufen, der Teil eines 8255 PPI ist, wenn Ihr Iterationsterminal auf 0 bleibt, durchläuft Ihr 8255 PPI eine Konfigurationsphase, bei der alle Anschlüsse innerhalb desselben PPI-Chips unabhängig von der Datenrichtung auf Niedriglogik eingestellt werden. Die Datenrichtungen an anderen Anschlüssen werden jedoch aufrechterhalten. Um diesen Effekt zu vermeiden, können Sie einen anderen Wert als 0 mit dem Iterationsterminal verbinden, sobald Sie die gewünschten Anschlüsse konfiguriert haben.***

24

Mittlere Digital-I/O-VIs

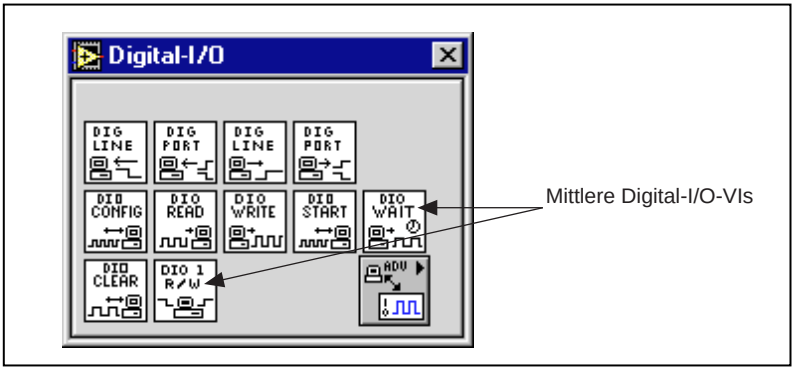
Dieses Kapitel beschreibt Mittlere Digitale-I/O-VIs. Bei diesen VIs handelt es sich um einzelne nichtgepufferte VI-Lösungen für häufig vorkommende digitale Probleme.

Das VI DIO einzeln lesen/schreiben ist eine Einzel-VI-Lösung für nichtgepufferte digitale Lesen-/Schreiben-VIs. Das VI DIO einzeln lesen/schreiben funktioniert mit jedem Gerät mit Digitalanschluß.

Sie können die anderen VIs - DIO konfigurieren, DIO starten, DIO lesen, DIO schreiben, DIO warten und DIO zurücksetzen - kombinieren, um anspruchsvollere Anwendungen zu erstellen, die gepuffertes digitales Lesen und Schreiben verwenden. Um diese VIs benutzen zu können, muß Ihr Gerät die Handshake-Funktion unterstützen.

Alle in diesem Kapitel beschriebenen VIs wurden auf der fundamentalen Bausteinblockschicht, den VIs fortgeschrittener Stufe, erstellt.

Sie können auf die Digitalen Zwischen-I/O-VIs durch Auswahl von **Funktionen**» **Datenerfassung**»**Digital-I/O** zugreifen. Bei den Mittleren Digitalen-I/O-VIs handelt es sich, wie nachstehend gezeigt, um VIs auf der zweiten und dritten Reihe der Palette **Digital-I/O**.



Fehlerbehandlung

LabVIEW vereinfacht die Fehlerbehandlung mit den Mittleren Digital-I/O-VIs. Jedes Zwischenstufen-VI hat einen Eingangscluster **Fehlereingang** und einen Ausgangscluster **Fehlerausgang.** Die Cluster enthalten einen Booleschen Wert, wodurch angezeigt wird, ob ein Fehler vorgekommen ist, sowie den Fehlercode für den Fehler und den Namen des VIs, das den Fehler zurückgegeben hat. Wenn der **Fehlereingang** einen Fehler anzeigt, gibt das VI die Fehlerinformation an **Fehlerausgang** zurück und stellt die Ausführung ein.

☞ **Hinweis** ***Das VI DIO zurücksetzen stellt eine Ausnahme zu dieser Regel dar - dieses VI beendet die Erfassung immer, und zwar unabhängig davon, ob in Fehlereingang ein Fehler angezeigt wird.***

Wenn Sie einen der Mittleren Digital-I/O-VIs in einer While-Schleife benutzen, sollten Sie die Schleife stoppen, wenn der **Status** im Cluster **Fehlerausgang** TRUE anzeigt. Wenn Sie den Fehlercluster mit dem Allgemeinen Error-Handler-VI verbinden, entziffert das VI die Fehlerinformation und beschreibt den Fehler.

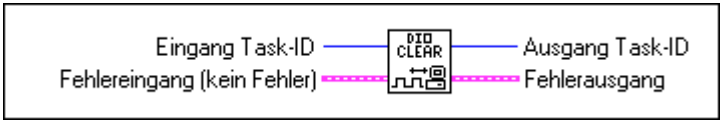
Das Allgemeine Error-Handler-VI befindet sich in **Funktionen**» **Zeit und Dialog** in LabVIEW. Weitere Informationen zu diesem VI finden Sie in Kapitel 10, *Zeit-, Dialog- und Fehlerfunktionen*.

Beschreibungen der Mittleren Digitalen-I/O-VIs

Folgende Mittlere Digitale-I/O-VIs sind verfügbar.

DIO zurücksetzen

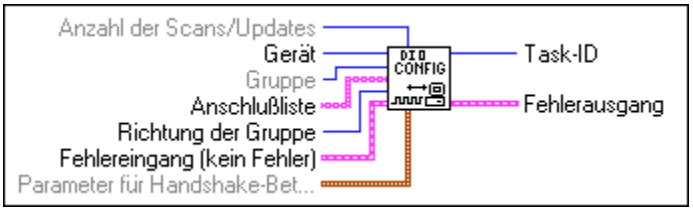
Ruft das VI Digitale Gruppenpuffer-Steuerung auf, um die Übertragung anzuhalten und die Gruppe zu beenden.



DIO Konfigurieren

Das VI DIO Konfigurieren ruft das VI Fortgeschrittene Digitale Gruppenkonfiguration auf, um der Gruppe eine Anschlußliste zuzuweisen, die Richtung der Gruppe festzulegen und die **Task-ID** zu erstellen. Das VI ruft dann das VI Digitale Modus-Konfiguration auf, um die Handshake-Parameter festzulegen, die sich nur auf die Operation der DIO-32-Geräte

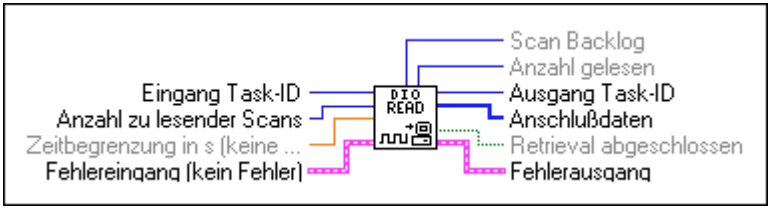
auswirken. Schließlich ruft das VI das VI Digitale Puffer-Konfiguration auf, um einen Puffer zuzuweisen, der die Abtastungen während des Lesens oder die zu schreibenden Updates aufnimmt.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Anschlüsse und Richtungen.

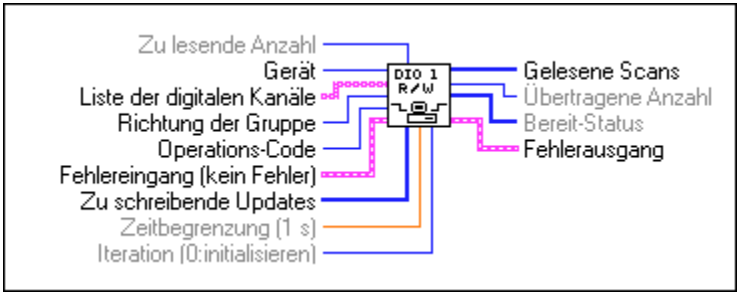
DIO lesen

Ruft das VI Digitalen Puffer lesen auf, um Daten vom internen Übertragungspuffer zu lesen und gibt die gelesenen Daten in **Mustern** zurück.



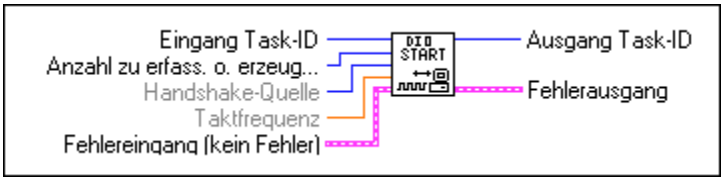
DIO einzeln lesen/schreiben

Liest oder schreibt digitale Daten auf die in der digitalen Datenliste angegebenen digitalen Kanäle. Dieses Einzel-VI konfiguriert und überträgt Daten. Wenn Sie dieses VI in einer Schleife benutzen, müssen Sie den Iterationszähler mit der Iterationseingabe verbinden, damit die Konfiguration nur einmal vorgenommen wird.



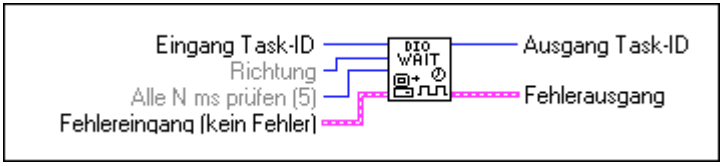
DIO starten

Startet eine gepufferte digitale I/O-Operation. Dieses VI ruft zur Einstellung der Taktrate das VI Digitale Taktkonfiguration auf, wenn der interne Takt das Handshake-Signal erzeugt, und startet dann durch Aufruf des VIs Digitale Puffer-Steuerung die Datenübertragung.



DIO warten

Wartet bis die digitale gepufferte Eingabe- oder Ausgabe-Operation vor der Rückgabe abgeschlossen ist. Bei der Eingabe erkennt das VI den Abschluß, wenn der vom VI Digitaler Puffer lesen zurückgegebene Erfassungsstatus mit oder ohne Backlog beendet wird. Bei der Ausgabe erkennt das VI den Abschluß, wenn das Anzeigeelement **Erzeugung abgeschlossen** des VIs DIO schreiben TRUE ist.

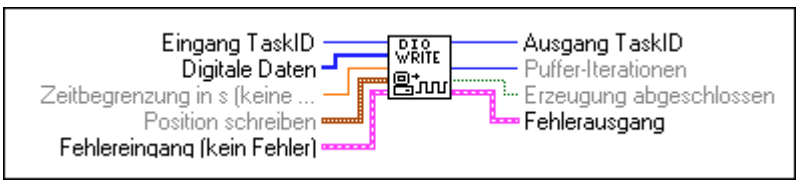


In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Handshake-Modi.

DIO schreiben

Ruft das VI Digitalen Puffer schreiben auf, um auf den internen Übertragungspuffer zu schreiben.

(Macintosh) Sie müssen den Puffer mit Daten füllen, ehe Sie das VI DIO starten für den Start der digitalen Ausgabeoperation benutzen. Sie können das VI DIO schreiben nach dem Start der Übertragung zur Erlangung von Statusinformationen aufrufen.

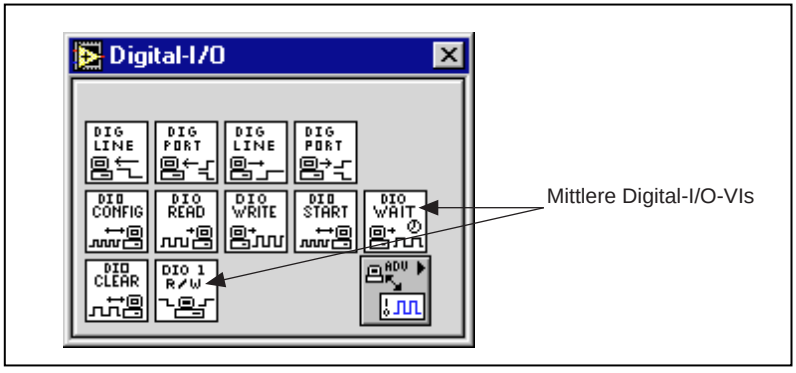


25

Fortgeschrittene Digital-I/O-VIs

Dieses Kapitel beschreibt die Fortgeschrittenen Digital-I/O-VIs, zu denen die VIs Digitaler Anschluß und Digitale Gruppe gehören. Die VIs digitaler Anschluß können Sie für sofortiges Lesen und Schreiben zu den digitalen Leitungen und Anschlüssen benutzen. Die VIs digitale Gruppe können Sie für sofortige oder getaktete I/Os oder für solche mit Handshake benutzen. Diese VIs stellen die Schnittstelle zur NI-DAQ-Software und die Grundlage für die Einfachen und Mittleren Digitalen-I/O-VIs dar.

Sie können auf die Palette **Fortgeschrittene Digital-I/O** durch Auswahl von **Funktionen»Datenerfassung»Digital-I/O»Fortgeschrittene Digital I/O** zugreifen. Das Icon, das Sie zum Zugriff auf die VIs Fortgeschrittene Digital-I/O auswählen müssen, befindet sich, wie nachstehend gezeigt, auf der untersten Reihe der Palette **Digital-I/O**.

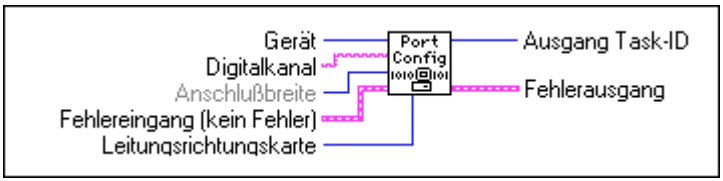


Beschreibungen der VIs Digitaler Anschluß

Die VIs Digitaler Anschluß führen nur sofortiges digitales Lesen und Schreiben durch.

DIO Anschluß-Konfiguration

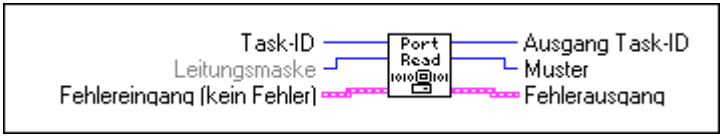
Erstellt eine digitale Kanalkonfiguration. Sie können die von diesem VI zurückgegebene **Task-ID** nur für VIs mit digitalem Anschluß benutzen.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Anschlüsse und Richtungen.

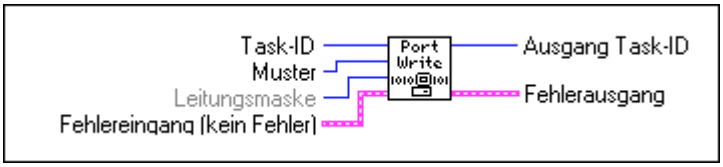
DIO Anschluß lesen

Liest den durch das **Task-ID** identifizierten Digitalkanal und gibt das Muster in **Mustern** gelesen zurück.



DIO Anschluß schreiben

Schreibt den Wert in **Mustern** an den vom **Task-ID** identifizierten digitalen Anschluß.

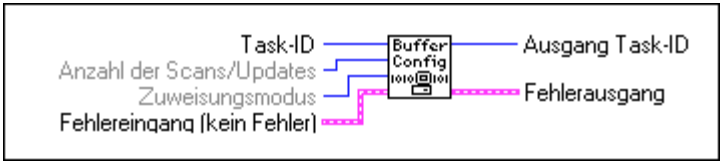


Beschreibungen der VIs Digitale Gruppe

Die VIs Digitale Gruppe führen sofortige oder getakte Digital-I/Os oder solche mit Handshake quittierte aus.

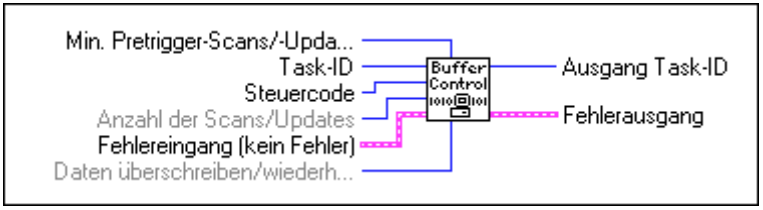
Konfiguration digitaler Puffer

Weist Speicherplatz für einen digitalen Eingabe- oder Ausgabepuffer zu.



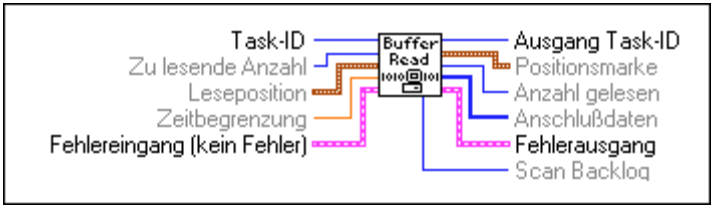
Steuerung digitaler Puffer

Startet eine Eingabe- oder Ausgabeoperation.



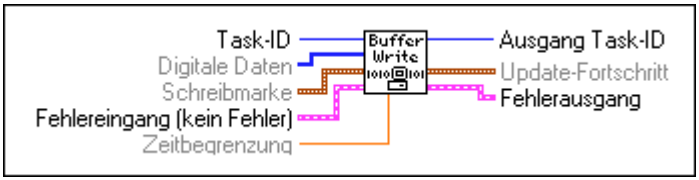
Lesen des digitalen Puffers

Gibt digitale Eingabedaten vom internen Datenpuffer zurück.



Schreiben des digitalen Puffers

Schreibt digitale Ausgabedaten an den durch das VI Digitale Puffer-Konfiguration erstellten Puffer. Das Schreiben beginnt immer an der Schreibmarke. Nach dem Schreiben zeigt die Schreibmarke auf das Update, das auf das zuletzt geschriebene Update folgt.

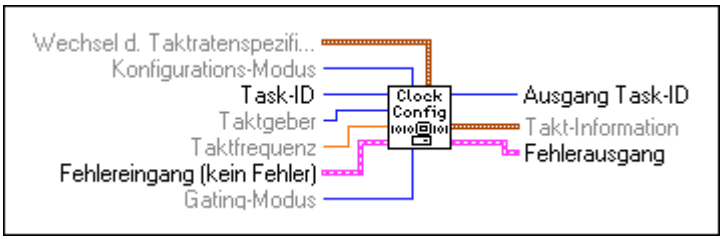


(Macintosh) Füllen Sie den Puffer mit Daten, ehe Sie das VI Digitale Pufferkontrolle für den Start der digitalen Ausgabeoperation benutzen. Sie können das VI Schreiben des digitalen Puffers nach Start der Übertragung aufrufen, um Statusinformationen abzurufen.

Die Gesamtzahl der vor dem Start an einen Puffer geschriebenen Updates kann weniger als die dem Puffer beim Aufruf des VIs Digitale Puffer-Konfiguration zugewiesenen Updates sein. Das VI erzeugt nur die an den Puffer geschriebenen Updates.

Konfiguration Digital-Clock

Konfiguriert ein DIO-32-Gerät, auf Grundlage der Taktausgabe für getaktete Digital-I/Os Handshake-Signale zu produzieren.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Takte.

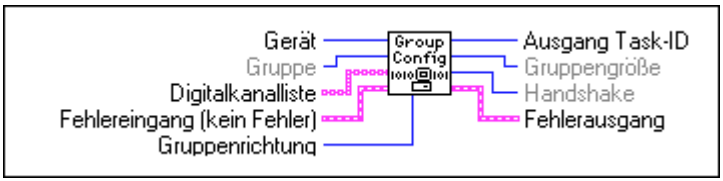
Folgendes Beispiel veranschaulicht, wie die drei Zeitbasis-Parameter zur Angabe einer Taktrate zu benutzen sind. Gehen Sie bitte davon aus, daß diese Parameter folgende Einstellungen haben:

Zeitbasis-Quelle:	1
Zeitbasis-Signal:	1.000.000,0 Hz
Zeitbasis-Divisor:	25

In diesem Fall betragen die Ticks pro Sekunde 1.000.000,0 geteilt durch 25, so daß LabVIEW ein Update der Digitalgruppe von 40.000 Mal pro Sekunde durchführt.

Konfiguration der Digitalgruppe

Definiert eine digitale Eingabe- oder Ausgabegruppe. Sie können die von diesem VI zurückgegebene **Task-ID** nur für die VIs Digitalgruppe benutzen.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Anschlüsse und Richtungen.

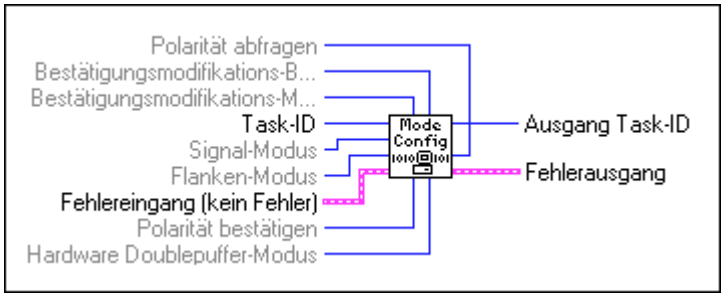
☞ **Hinweis** ***Derselbe Digitalkanal kann nicht zu zwei verschiedenen Gruppen gehören. Wenn Sie für die Benutzung eines bestimmten Digitalkanals eine Gruppe konfigurieren, so muß es sich bei diesem Digitalkanal um einen Digitalkanal handeln, der noch nicht in einer anderen Gruppe definiert wurde, da Sie anderenfalls eine Fehlermeldung erhalten.***

MIO-Geräte (außer AT-MIO-16D und AT-MIO-16DE-10) sowie die Geräte NB-TIO-10, LPM, DAQCard-500, 516-Geräte, DAQCard-700, PC-TIO-10, AO-2DC-Geräte, PC-OPDIO-16 und AT-AO-6/10 erlauben kein Handshake. Für diese Geräte sind die VIs Digitalanschluß besser geeignet. Handshake ist nicht erlaubt, wenn die **Digitalkanalliste** aus Kanalnamen besteht. AT-MIO-16D und AT-MIO-16DE-10-Geräte erlauben keinen Handshake, wenn zur **Digitalkanalliste** die Anschlüsse 0, 1 und/oder 4 gehören. DIO-96-Geräte erlauben keinen Handshake, wenn zur **Digitalkanalliste** die Anschlüsse 2, 5, 8 und/oder 11 gehören. Die Geräte DIO-24 und Lab sowie Geräte der Serie 1200 erlauben keinen Handshake, wenn zur **Digitalkanalliste** Anschluß 2 gehört. Das Gerät DIO-32F erlaubt Handshake nur für folgende Konfigurationen:

- Eine Gruppe, zu der ein beliebiger Anschluß gehört.
- Eine Gruppe, zu der die Anschlüsse 0 und 1 oder 2 und 3 gehören (in dieser Reihenfolge).
- Eine Gruppe, zu der die Anschlüsse 0, 1, 2 und 3 gehören (in dieser Reihenfolge).

Konfiguration des Digitalmodus

Konfiguriert die Handshake-Charakteristika für DIO-32-Geräte.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Handshake-Modi.

DIO-Parameter

Konfiguriert verschiedene mit der Digital-Eingabe- und Ausgabe zusammenhängende, nicht von anderen DIO-VIs konfigurierte Parameter und ruft diese ab.

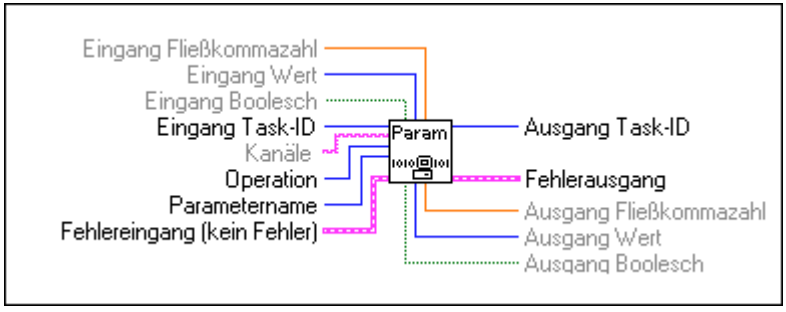


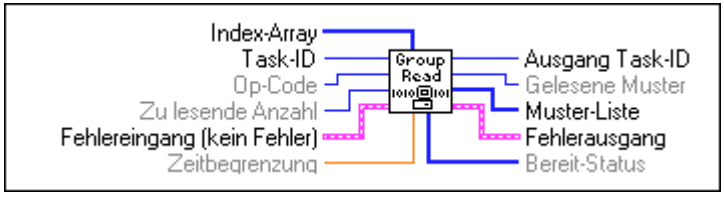
Tabelle 25-1 führt gerätespezifische Parameter und zulässige Bereiche für Geräte auf.

Tabelle 25-1. Gerätespezifische Parameter und zulässige Bereiche.

Gerät	Parameter-name	Unter-stützung	Einstellung möglich	Eingang/Aus-gang, den Sie benutzen sollten	Zulä-ssige Werte	Stan-dard-werte
VXI-DIO -128	0: Eingangs-anschluß logischer Schwellwert	pro Eingangs-anschluß	ja	Kanäle, einfließen, ausfließen	N/A	N/A
DAQ-DIO -6533 (DIO-32HS)	1: ACK/erfor-derlicher Austausch	pro Gruppe	ja	Task-ID in, Wert ein, Wert aus	Aus, ein	N/A
	2: Taktum-kehrung	pro Gruppe	ja	Task-ID in, Wert ein, Wert aus	Aus, ein	N/A

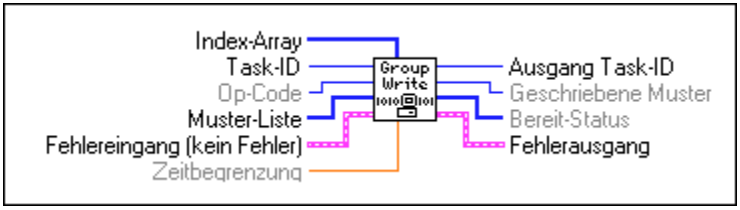
Digital einzeln lesen

Liest die Digitalkanäle, die zur von **Task-ID** identifizierten Kanalgruppe gehören, und gibt die Lesemuster zurück.



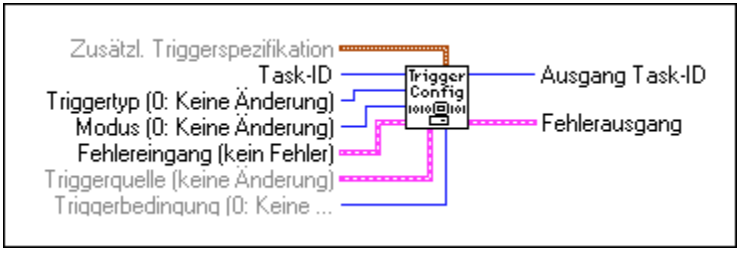
Digital einzeln schreiben

Schreibt die Daten in **Musterarrays** zu den Digitalkanälen, die zur von **Task-ID** identifizierten Gruppe gehören.



Konfiguration Digitaltrigger

Konfiguriert die Triggerbedingung für den Start und/oder Stopp einer Digitalmuster-Erzeugungsoperation. Dieses VI ist nur gültig, wenn der Parameter **Handshake-Quelle** des VIs Konfiguration Digitalclock auf 1 oder 4 eingestellt hat (interne oder externe Mustererzeugung mit externem Takt).

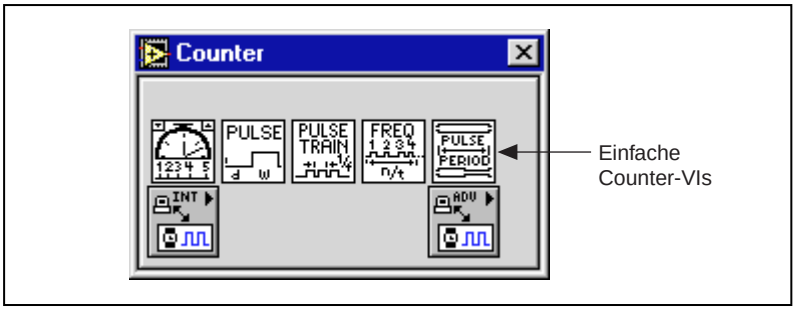


26

Einfache-Counter-VIs

Dieses Kapitel beschreibt die Einfachen-Counter-VIs, die einfache Zähloperationen durchführen. Sie können diese VIs vom Frontpanel aus durchführen oder sie in SubVIs in Grundanwendungen benutzen.

Sie können auf die Einfachen-Counter-VIs durch Auswahl von **Funktionen**»**Datenerfassung**»**Counter** zugreifen. Bei den Einfachen-Counter-VIs handelt es sich um die VIs auf der obersten Reihe der Palette **Counter**.



Dieses Kapitel beschreibt High-Level-VIs für die Programmierung von Counters an MIO, TIO- und anderen Geräten mit DAQ-STC oder Am9513 Counter-/Timer-Chips. Diese VIs rufen zur einzelnen verzögerten TTL-Impulserzeugung, zur endlichen und fortgesetzten Erzeugung von Impulsfolgen und zur Messung von Frequenz, Impulsbreite oder -periode eines TTL- Signals die Mittleren Counter-VIs auf.

☞ **Hinweis** ***Diese VIs funktionieren nicht mit Lab-Geräten und Geräten der Serie 1200, DAQCards und anderen Geräten mit 8253/54-Chip. Benutzen Sie für diese Geräte die ICTR-Steuerung mittlerer Stufe. In Kapitel 27, Mittlere Counter-VIs, finden Sie weitere Informationen zur VI ICTR-Steuerung.***

Einige dieser VIs benutzen zusätzlich zu den genannten noch andere Counter. In diesem Fall wird ein logisch benachbarter Counter ausgewählt, auf den als **Counter+1** Bezug genommen wird, wenn er der benachbarte, logisch höhere Counter ist, und als **Counter–1**, wenn er der benachbarte, logisch niedrigere Counter ist.

Wenn bei einem Gerät mit dem Am9513-Chip der Counter 1 ist, dann ist **Counter+1** Counter 2 und **Counter–1** ist Counter 5.

Weitere Informationen zu Benachbarten Counter-VIs finden Sie in Kapitel 27, *Mittlere Counter-VIs*.

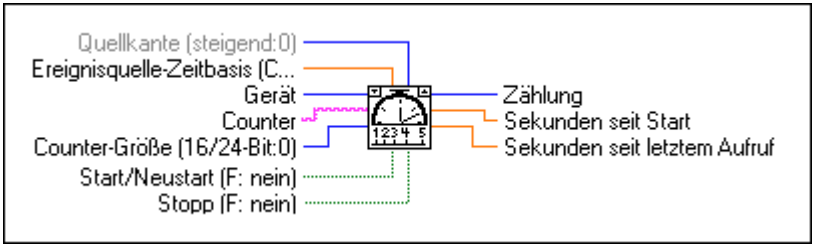
Beispiele für die Anwendung von Einfachen Counter-VIs finden Sie in der Beispielsbibliothek durch Öffnen von `examples\daq\counter`.

Beschreibungen Einfacher Counter-VIs

Folgende Einfache Counter-VIs sind verfügbar.

Ereignis oder Zeit zählen

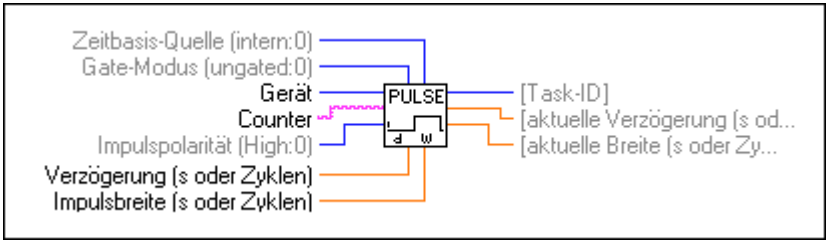
Konfiguriert einen oder zwei Counter, um externe Ereignisse oder abgelaufene Zeit zu zählen. Ein externes Ereignis ist ein hoher oder niedriger Signalübergang am angegebenen SOURCE-Pin des Counters.



Stellen Sie zur Zählung von Ereignissen **Ereignisquelle-Zeitbasis** auf 0.0, und verbinden Sie das Signal, das Sie zählen möchten, mit dem SOURCE-Pin des Counters. Stellen Sie zur Zählung von Zeit diese Steuerung auf die Zeitbasis-Frequenz ein, die Sie benutzen möchten.

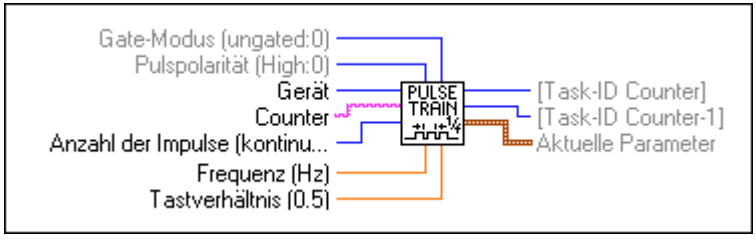
Verzögerten Impuls erzeugen

Konfiguriert und startet einen Zähler zur Erzeugung eines Einzelimpulses mit der angegebenen Verzögerung und Impulsbreite am OUT-Pin des Counters. Ein Einzelimpuls besteht aus einer Verzögerungsphase (Phase 1), die von einer Impulsphase gefolgt wird (Phase 2) und dann zum Level Phase 1 zurückkehrt. Bei Auswahl einer internen Zeitbasis wählt das VI zur Erreichung der gewünschten Charakteristika die höchste Auflösungs-Zeitbasis für den Counter aus. Bei Auswahl einer externen Zeitbasis gibt der Benutzer die Verzögerung und Breiten als Zyklen dieses Signals an. Führen Sie das VI Counter starten mit der Task-ID dieses VIs zur Erzeugung eines anderen Impulses aus. Sie können den Impuls mit einem Signal am GATE-Pin des Counters wahlweise gattern oder triggern.



Impulsfolge erzeugen

Konfiguriert den angegebenen Counter zur Erzeugung einer kontinuierlichen Impulsfolge am OUT-Pin des Counters oder zur Erzeugung einer Impulsfolge mit endlicher Länge unter Benutzung des angegebenen Counters und eines benachbarten Counters. Das Signal hat die vorgeschriebene Frequenz, Tastverhältnis und Polarität. Jeder Zyklus der Impulsfolge besteht aus einer Verzögerungsphase (Phase 1), die von einer Impulsphase (Phase 2) gefolgt wird.

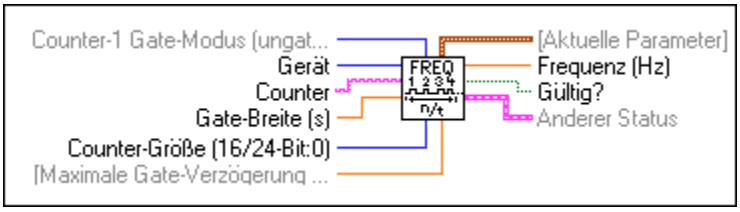


Dieses VI benutzt zur Erzeugung eines kontinuierlichen Impulses nur den angegebenen **Counter**. Zur Erzeugung eines Impulses endlicher Länge benutzt das VI auch **Counter–1**, um zum Gattern des **Counters** einen Impuls mit Mindestverzögerung zu erzeugen. Führen Sie zur Erzeugung einer anderen Impulsfolge das VI Zwischen-Counter starten mit den von diesem VI bereitgestellten **Task-IDs** aus. Führen Sie zum Stoppen einer kontinuierlichen Impulsfolge das VI Mittlere Counter stoppen aus, oder führen Sie diesen Counter zur Erzeugung eines kurzen Impulses wieder aus. Sie müssen den OUT-Pin von **Counter–1** zur Erlangung einer Impulsfolge mit endlicher Länge extern mit dem GATE-Pin des **Counters** verbinden. Sie können den Start der Folge mit einem Signal am GATE-Pin von **Counter–1** wahlweise gattern oder triggern.

☞ **Hinweis** ***Eine Impulsfolge besteht aus einer Serie verzögerter Impulse, bei der Phase 1 oder die erste Phase eines jeden Impulses der inaktive Status der Ausgabe ist (niedrig für einen hohen Impuls) und die Phase 2 oder zweite Phase der Impuls selbst ist.***

Frequenz messen

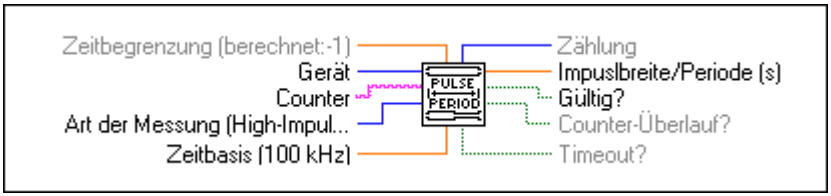
Mißt die Frequenz eines TTL-Signals an dem angegebenem SOURCE-Pin durch Zählen der positiven Signalflanken während einer festgelegten Zeitspanne. Zusätzlich zu dieser Verbindung müssen Sie den GATE-Pin des Counters mit dem OUT-Pin von Counter 1 verbinden. Dieses VI ist für relative Hochfrequenzsignale nützlich, wenn viele Signalzyklen während der Timingperiode vorkommen. Benutzen Sie für relative Niedrigfrequenzsignale das VI Impulsbreite oder -periode messen. Denken Sie daran, daß eine Periode = 1/Frequenz (Hz) ist.



Dieses VI konfiguriert den angegebenen **Counter** und den **Counter+1** (für Am9513 wahlweise) als Ereigniscounter, um steigende Signalflanken am SOURCE-Pin des Counters zu zählen. Das VI konfiguriert auch **Counter–1** zur Erzeugung eines Mindestverzögerungs-Impulses, um den Ereigniscounter zu gattern, startet den Ereigniscounter und dann den Gatecounter, wartet die erwartete Gateperiode ab und liest dann den Gatecounter, bis dessen Ausgabestatus niedrig ist. Dann liest das VI den Ereigniscounter und errechnet die Signalfrequenz (**Anzahl der Ereignisse/tatsächliche Gate-Impulsbreite**) und stoppt die Zähler. Sie können die Operation wahlweise mit einem Signal am GATE-Pin von **Counter–1** gattern oder triggern.

Impulsbreite oder -periode messen

Mißt die Impulsbreite (Länge der Zeit, die ein Signal hoch oder niedrig ist) oder Impulsperiode (Länge der Zeit zwischen benachbarten steigenden oder fallenden Flanken) eines mit dem GATE-Pin des **Counters** verbundenen TTL-Signals. Die angewandte Methode gattert eine interne Zeitbasis-Clock, wobei das Signal gemessen wird. Dieses VI ist zur Messung der Periode oder Frequenz (1/Period) von relativen Niedrigfrequenz-Signalen nützlich, wenn während des Gates viele Zeitbasis-Zyklen vorkommen. Benutzen Sie zur Messung der Periode oder Frequenz relativer Hochfrequenz-Signale das VI Frequenz messen.



Das VI iteriert, bis eine gültige Messung, ein **Timeout**- oder **Counter**-Überlauf vorkommt. Eine gültige Messung liegt dann vor, wenn die **Zählung**≥ 4 ohne Counter-Überlauf ist. Wenn ein **Counter-Überlauf** vorkommt, müssen Sie die **Zeitbasis** reduzieren. Wenn Sie mit einer Impulsbreiten-Messung während der Phase, die Sie messen möchten, beginnen, erhalten Sie eine unrichtige niedrige Messung. Stellen Sie daher sicher, daß der Impuls nicht vorkommt, bis der **Counter** gestartet wurde. Diese Beschränkung trifft auf Periodenmessungen nicht zu.

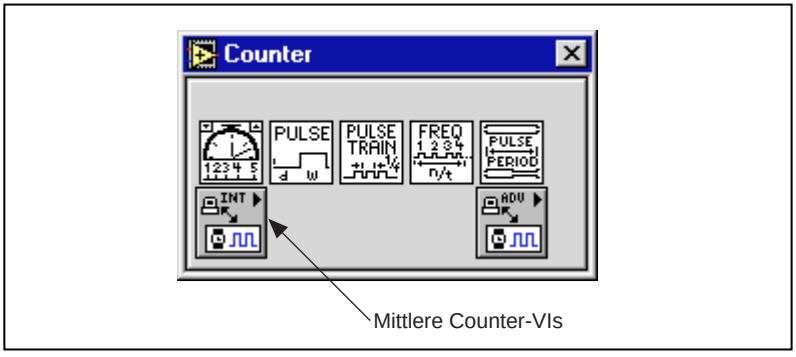
27

Mittlere Counter-VIs

Dieses Kapitel beschreibt Mittlere Counter-VIs, die Sie zur Programmierung von Counters an MIO,- TIO- und anderen Geräten mit DAQ-STC- oder Am9513-Counterchips benutzen können. Diese VIs rufen die Fortgeschrittenen Counter-VIs auf, um die Counter für allgemeine Operationen zu konfigurieren sowie die Counter zu starten, zu lesen und zu stoppen. Sie können diese Counter zur Erzeugung von Einzelimpulsen und kontinuierlichen Impulsfolgen und zur Zählung von Ereignissen oder abgelaufener Zeit benutzen, sowie zum Herunterdividieren eines Signals und zur Messung der Impulsbreite oder -periode. Die Einfachen Counter-VIs rufen die Mittleren Counter-VIs für verschiedene Impulserzeugungen, Zählvorgänge und Meßoperationen auf.

Dieses Kapitel beschreibt auch das VI ICTR Steuerung, das Sie mit Lab-Geräten und Geräten der Serie 1200 und PC-LPM-Geräten mit dem 8253/54 Counter-/Timer-Chip benutzen.

Sie können auf die Mittleren Counter-VIs durch Auswahl von **Funktionen»Datenerfassung»Counter»Mittlere Counter** zugreifen. Bei den Mittleren Counter-VIs handelt es sich, wie nachstehend gezeigt, um die VIs auf der zweiten Reihe der Palette **Counter.**



Fehlerbehandlung

LabVIEW vereinfacht die Fehlerbehandlung mit den Mittleren Counter-VIs. Jedes VI der mittleren Stufe hat einen Eingangscluster **Fehlereingang** und einen Ausgangscluster **Fehlerausgang.** Die Cluster enthalten einen Booleschen Wert, wodurch angezeigt wird, ob ein Fehler vorgekommen ist, sowie den Fehlercode für den Fehler und den Name des VIs, das den Fehler zurückgegeben hat. Wenn **Fehlereingang** einen Fehler anzeigt, gibt das VI die Fehlerinformation an **Fehlerausgang** zurück und stellt die Funktion ein.

Wenn Sie eines der Mittleren-Counter-VIs in einer While-Schleife benutzen, sollten Sie die Schleife stoppen, wenn der **Status** im Cluster **Fehlerausgang** TRUE anzeigt. Wenn Sie den Fehlercluster mit dem Allgemeinen Error-Handler-VI verbinden, entziffert das VI die Fehlerinformation und beschreibt den Fehler.

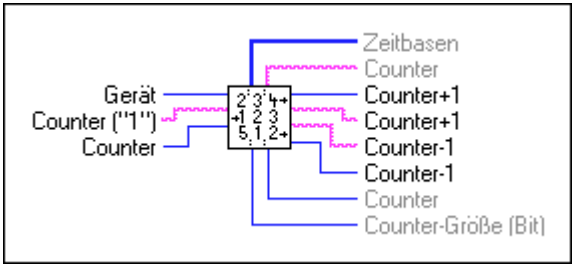
Das Allgemeine Error-Handler-VI befindet sich in **Funktionen**»**Utilities** in LabVIEW. Weitere Informationen zu diesem VI finden Sie in Ihrem LabVIEW Benutzerhandbuch.

Beschreibungen von Mittleren Counter-VIs

Folgende Mittlere Counter-VIs sind verfügbar.

Benachbarte Counter

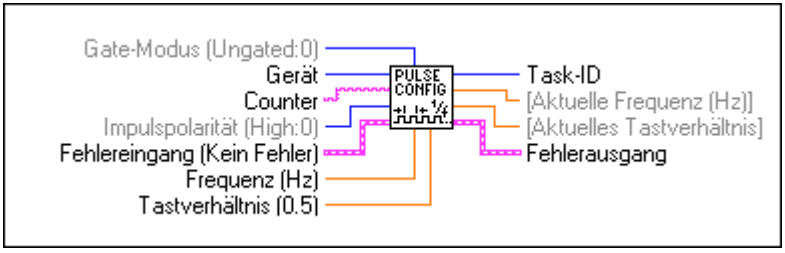
Identifiziert die Counter, die logisch mit einem angegebenen Counter eines MIO- oder TIO-Geräts benachbart sind. Es gibt auch die Counter-Größe (Anzahl der Bits) und die Zeitbasis zurück.



Geräte mit Am9513-Chip haben ein oder zwei Sätze von fünf, 16-Bit Countern (1–5, 6–10), die kreisförmig verbunden werden können. So ist z.B. der nächsthöhere Counter zu Counter (genannt **Counter+1**) 2 und der nächstniedrigere (genannt **Counter–1**) ist 5.

Konfig. für kontinuierliche Impulserzeugung

Konfiguriert einen Counter zur kontinuierlichen Erzeugung von TTL-Impulsfolgen an dessen OUT-Pin.



Das Signal wird durch wiederholtes zweimaliges Dekrementieren des Counters erzeugt, zuerst für die Verzögerung für den Impuls (Phase 1), dann für den Impuls selbst (Phase 2). Das VI wählt zur Erreichung der gewünschten Charakteristika die Zeitbasis mit der höchsten Auflösung aus. Sie können die Operation wahlweise mit einem Signal am GATE-Pin des Counters gattern oder triggern. Zum Start der Impulsfolge oder zur Aktivierung von dessen Gatterung können Sie das VI Counter starten aufrufen.

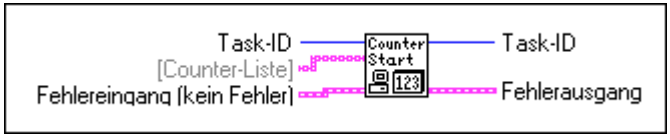
Counter lesen

Liest den oder die von **Task-ID** identifizierten Counter.

Das VI ist so ausgelegt, daß es einen Counter oder zwei verknüpfte Counter eines Am9513-Counter-Chips oder einen Counter eines DAQ-STC-Counter-Chips liest.

Counter starten

Startet den durch **Task-ID** identifizierten Counter.

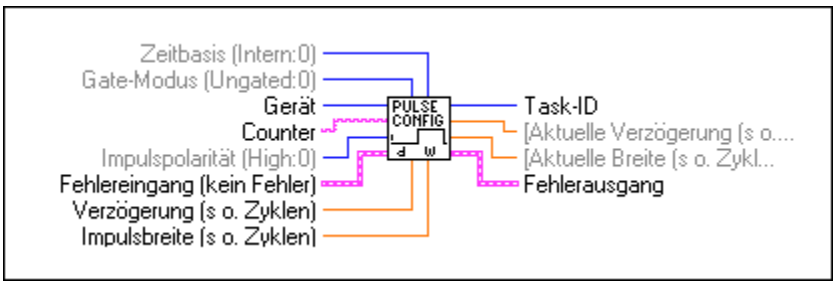


Counter stoppen

Stoppt eine Zähloperation sofort oder bedingt durch einen Eingabefehler.

Konfiguration für verzögerte Impulserzeugung

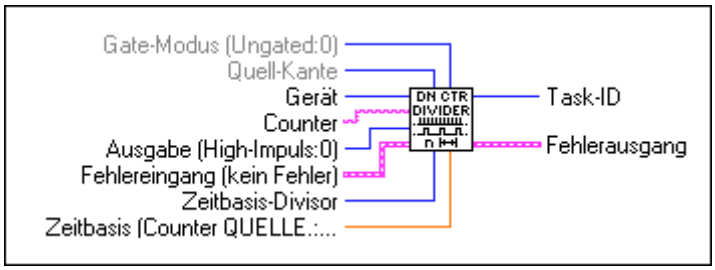
Konfiguriert einen Counter zur Erzeugung eines einzelnen, verzögerten TTL-Impulses an seinem OUT-Pin.



Das Signal wird durch zweimaliges Dekrementieren des **Counters** erzeugt, zuerst für die Verzögerung für den Impuls (Phase 1), dann für den Impuls selbst (Phase 2). Bei Auswahl einer internen Zeitbasis wählt das VI die Zeitbasis höchster Auflösung für den **Counter** zur Erreichung der gewünschten Charakteristika aus. Bei Auswahl eines externen Zeitbasissignals bestimmt der Benutzer die Verzögerung und die Breite als Zyklen dieses Signals. Sie können die Operation wahlweise durch ein Signal am GATE-Pin des Counters gattern oder triggern. Zum Start des Impulses oder zur Aktivierung von dessen Gatterung können Sie das VI Counter starten aufrufen.

Konfiguration für Abwärtszähler oder Teiler

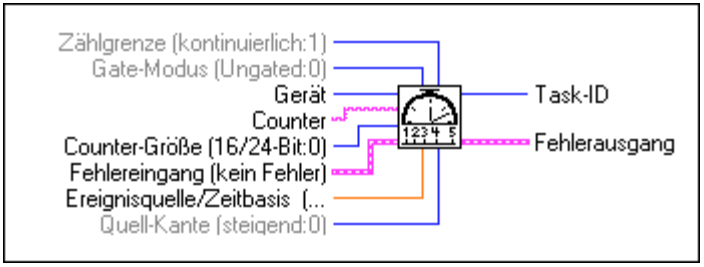
Konfiguriert den angegebenen **Counter**, um ein Signal am SOURCE-Pin des Counters oder an einem internen Zeitbasis-Signal mit Hilfe eines **Zeitbasis-Divisors** genannten Zählwerts abwärtszuzählen oder zu teilen. Das Ergebnis besteht darin, daß das Signal am OUT-Pin des Counters dieselbe Frequenz besitzt wie das Eingabesignal geteilt durch den **Zeitbasis-Teiler.**



Sie können dieses VI zur Erzeugung von endlichen Impulsfolgen durch Aktivierung kontinuierlicher Impulserzeugung benutzen, bis die gewünschte Impulszahl vorgekommen ist. Sie können es auch anstelle des VIs Konfig. für kontinuierliche Impulserzeugung benutzen, um eine Folge von Strobe- oder Trigger-Signalen zu erzeugen.

Konfiguration Ereignis- oder Zeit-Counter

Konfiguriert einen oder zwei Counter, um Flanken am Signal am SOURCE-Pin des angegebenen Counters oder die Anzahl der Zyklen eines angegebenen internen Zeitbasissignals zu zählen.

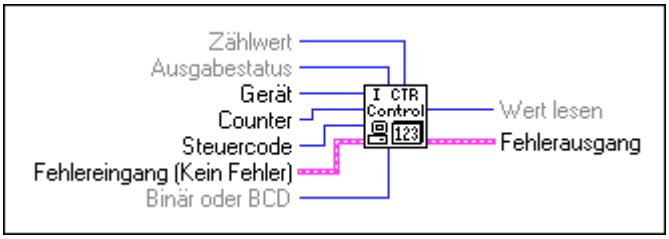


Wenn die interne Zeitbasis benutzt wird, funktioniert dieses VI wie die Funktion Tick Count (ms), benutzt aber am DAQ-Gerät einen Hardware-Counter mit programmierbarer Auflösung. Sie können die Operation wahlweise am GATE-Pin des Counters mit einem Signal gattern oder triggern. Rufen Sie das VI Counter starten auf, um die Operation zu starten oder um dessen Gatterung zu aktivieren.

ICTR-Steuerung

Steuert Counter an Geräten, die den 8253/54-Chip benutzen. Dazu gehören:

- Lab-Geräte und Geräte der Serie 1200, DAQCard-500 und DAQCard-700
- **(Windows)** LPM-Geräte, 516-Geräte.



Im Setup-Modus 0, wie in Abbildung 27-1 gezeigt, wird die Ausgabe nach der Moduseinstellungs-Operation niedrig und der **Counter** beginnt herunterzuzählen, während die Gateeingabe hoch ist. Die Ausgabe wird hoch, wenn der **Counter** das TC erreicht (dies ist dann der Fall, wenn der Counter auf 0 absinkt), und er bleibt hoch, bis Sie den ausgewählten Counter auf einen anderen Modus einstellen.

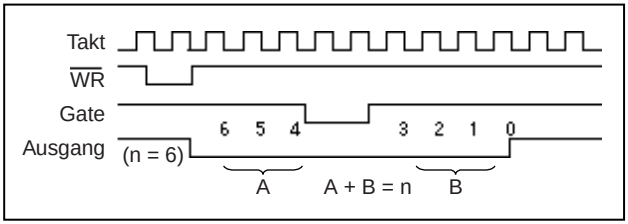


Abbildung 27-1. Setup-Modus in ICTR-Steuerung

Im Setup-Modus 1, wie in Abbildung 27-2 gezeigt, wird die Ausgabe niedrig im **Zählwert,** der der führenden Flanke des Gateeingangs folgt, und wird hoch am TC.

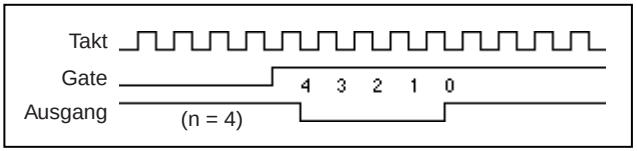


Abbildung 27-2. Setup-Modus 1 in ICTR-Steuerung

Im Setup-Modus 0, wie in Abbildung 27-3 gezeigt, wird die Ausgabe eine Periode der Clockeingabe lang niedrig. Der **Zählwert** gibt die Periode zwischen den Ausgangsimpulsen an.

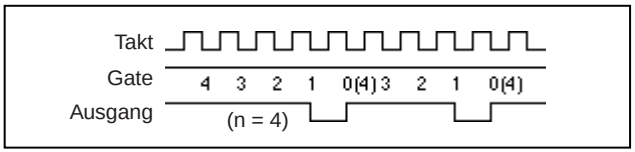


Abbildung 27-3. Setup-Modus 2 in ICTR-Steuerung

Im Setup-Modus 3, bleibt die Ausgabe für die Hälfte der **Zähl-**Clockimpulse hoch und bleibt für die andere Hälfte niedrig. Siehe Abbildung 27-4.

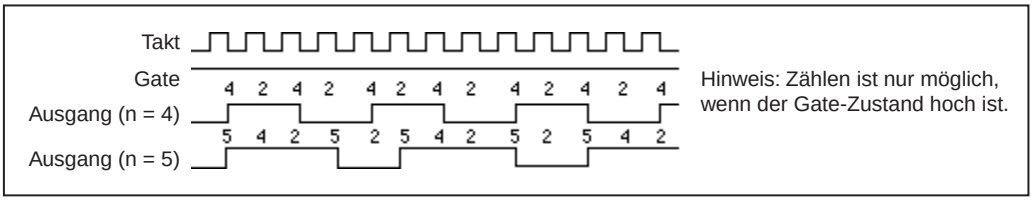


Abbildung 27-4. Setup-Modus 3 in ICTR-Steuerung

Im Setup-Modus 4, wie in Abbildung 27-5 gezeigt, ist die Ausgabe anfänglich hoch, und der **Zähler** beginnt herunterzuzählen, während die Gate-Eingabe hoch ist. Am TC wird die Ausgabe für einen Clock-Impuls niedrig und wird dann wieder hoch.

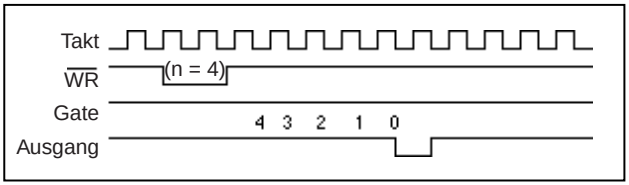


Abbildung 27-5. Setup-Modus 4 in ICTR-Steuerung

Setup-Modus 5 ähnelt Modus 4, außer daß die Gate-Eingabe den Zählerstart triggert. In Abbildung 27-6 finden Sie eine Abbildung von Modus 5.

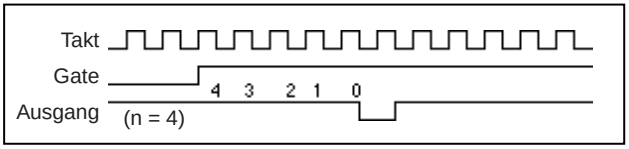
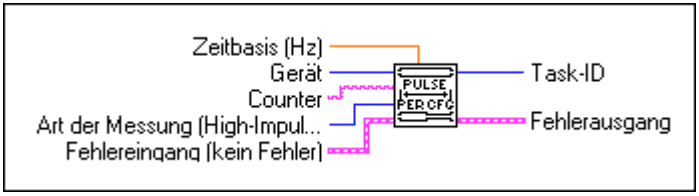


Abbildung 27-6. Setup-Modus 5 in ICTR-Steuerung

Im Datenblatt 8253 Programmierbarer Intervall-Timer in Ihrem Lab-Gerät Benutzerhandbuch finden Sie Details über diese Modi und die damit verbundenen Timing-Diagramme.

Konfiguration der Impulsbreite oder Periodenmessung

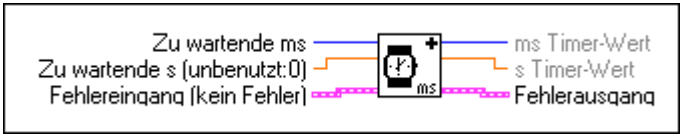
Konfiguriert den angegebenen Counter, um die Pulsbreite oder -periode eines mit seinem GATE-Pin verbundenen TTL-Signals zu messen.



Die Messung wird durch Zählung der Anzahl der Zyklen der angegebenen Zeitbasis zwischen den entsprechenden Start- und Endereignissen durchgeführt. Zur genauen Messung der Impulsbreite muß der Impuls vorkommen, nachdem der **Zähler** gestartet wurde. Rufen Sie zum Start der Operation das VI Counter starten auf. Sie können dieses VI auch zur Messung der Frequenz von Niedrigfrequenz-Signalen verwenden. Um genaure Messungen zu erhalten, müssen Sie eine schnellere **Zeitbasis** benutzen.

Warten+ (ms)

Ruft die Funktion Warten (ms) nur auf, wenn kein Eingabefehler vorliegt.



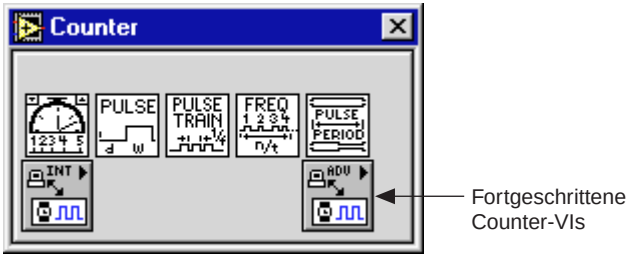
Dieses VI ist nützlich, wenn Sie zwischen den Aufrufen von I/O-SubVIs, die den Fehler-I/O-Mechanismus verwenden, warten möchten. Ohne diesen Mechanismus müssen Sie zur Steuerung des Ausführungsbefehls eine Sequenzstruktur verwenden.

28

Fortgeschrittene Counter-VIs

Dieses Kapitel beschreibt die VIs, die Hardware-Counter konfigurieren und steuern. Sie können diese VIs zur Erzeugung von Rechtecktsignalen variabler Tastverhältnisse, zum Zählen von Ereignissen oder Zeit und zum Messen von Periodendauer und Frequenzen benutzen.

Sie können auf die Palette **Fortgeschrittener Counter** durch Auswahl von **Funktionen»Datenerfassung»Counter»Fortgeschrittener Counter** zugreifen. Das Icon, das Sie für den Zugriff auf die VIs Fortgeschrittene Counter auswählen müssen, befindet sich, wie nachstehend gezeigt, auf der untersten Reihe der Palette **Counter**.



☞ **Hinweis** ***Benutzen Sie bei der Arbeit mit Datenerfassung nur die bei jedem VI notwendigen Eingänge. Lassen Sie die restlichen Eingänge unverbunden; LabVIEW stellt sie auf deren Standardwerte ein. Im Hilfe-Fenster sind die wichtigsten Terminals in Fettschrift bezeichnet, die am seltensten benutzten sind in eckigen Klammern angegeben. Die in runden Klammern angegebenen Werte sind Standardwerte.***

Nachfolgend sind die Arten von Counter-Chips aufgeführt, die Ihr Gerät haben muß, um mit Ihrer LabVIEW-Version arbeiten zu können:

- DAQ-STC-Counter-Chip
- Am9513-Counter-Chip
- 8253/54-Counter-Chip

Das VI ICTR-Steuerung ist das einzige VI, das mit Geräten arbeitet, die den 8253/54-Counter-Chip beinhalten.

In Tabelle 28-1 sind die mit den verschiedenen Geräten benutzten Counter-Chips aufgeführt.

Tabelle 28-1. Counter-Chips und ihre verfügbaren DAQ-Geräte

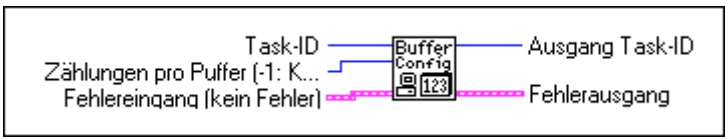
Counter-Chip	DAQ-Gerät
Am9513	AT-MIO-16, AT-MIO-16D, AT-MIO-16F-5, AT-MIO-16X, AT-MIO-64F-5, PC-TIO-10, All AO-2DC-Geräte, EISA-A2000, NB-MIO-16, NB-MIO-16X, NB-DMA-8-G, NB-DMA2800, NB-TIO-10, NB-A2000
DAQ-STC	Alle Geräte der E-Serie, 5102-Geräte
8253/54	Alle Lab-Geräte und Geräte der Serie 1200, DAQCard-500, DAQCard-700, LPM-Geräte, 516-Geräte

Beschreibungen Fortgeschrittener Counter-VIs

Folgende Fortgeschrittene Counter-VIs sind verfügbar.

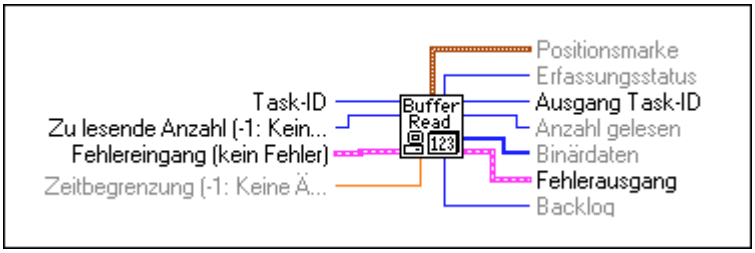
CTR Puffer-Konfiguration

Weist Speicherplatz zu, wo LabVIEW Daten speichert. Das VI CTR Puffer-Konfiguration konfiguriert auch die angegebene Gruppe, anstelle der üblichen Einzelpunkt-Operationen gepufferte Counter-Operationen durchzuführen.



CTR Puffer lesen

Gibt Daten vom Puffer zurück, der von CTR Puffer-Konfiguration zugewiesen wurde.



☞ **Hinweis** ***Das Lesen vom Zählpuffer in Inkrementen wird unterstützt. Der kreisförmige Gebrauch des Puffers ist jedoch nicht implementiert. Sie müssen deshalb einen endlichen Puffer einrichten. Sie können vom endlichen Puffer lesen, während er sich füllt.***

CTR Gruppen-Konfiguration

Vereint einen oder mehrere Counter in einer Gruppe. Sie können Counter-Gruppen mit mehr als einem Counter zum gleichzeitigen Starten, Stoppen oder Lesen mehrerer Counter verwenden. DAQ-STC-Geräte unterstützen derzeit nicht mehrere Counter in derselben Gruppe.

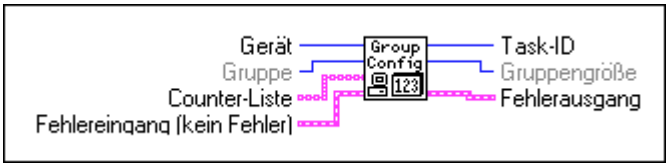


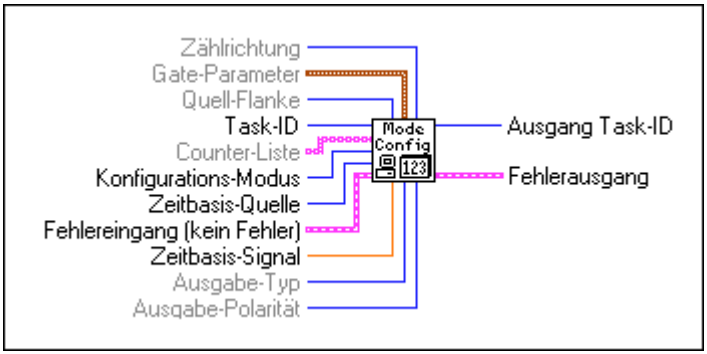
Tabelle 28-2 enthält die gültige Anzahl von Countern für von diesem VI unterstützten Geräte.

Tabelle 28-2. Gültige Counternummer für CTR-Gruppenkonfigurationsgeräte

Gerätetyp	Gültige Anzahl
DAQ-STC-Geräte	0 und 1
Am9513 MIO-Geräte	1, 2 und 5
NB-DMA-8-G, NB-DMA2800	1 bis 5
PC-TIO-10, NB-TIO-10	1 bis10
EISA-A2000, NB-A2000	2

CTR Modus-Konfiguration

Konfiguriert einen oder mehrere Counter für eine bestimmte Counter-Operation und wählt das Quellen-Signal, den Gate-Modus und das Ausgabeverhalten beim Terminalcount aus (TC).



Dieses VI startet die Counter nicht. Benutzen Sie zum Starten der Counter das VI CTR-Steuerung mit **Steuercode** `1` (Start). Wenn Sie einen Counter zur Impulserzeugung benutzen, müssen Sie dieses VI nicht aufrufen, außer Sie möchten **Gate-Modus** oder das Ausgabeverhalten ändern.

Die Modi 3, 4 und 6 können mit oder ohne gepuffertes Zählen verwendet werden. Modus 7 muß mit gepuffertem Zählen verwendet werden. Rufen Sie beim gepufferten Zählen das VI CTR Puffer-Konfiguration zum Start der Operation vor oder nach dem VI CTR Modus-Konfiguration und vor dem VI CTR Steuerung auf, und rufen Sie zum Lesen der gepufferten Zählwerte dann das VI CTR-Puffer lesen auf. Rufen Sie bei gepufferten oder ungepufferten Operationen das VI CTR Steuerung auf, um die zuletzt erfaßten, ungepufferten Zählwerte zu lesen.

Wenn nichts anderes angegeben ist, werden in den nachstehenden Abbildungen Timing und Counter-Werte für Operationen gezeigt, in denen der **Gate-Modus** auf High-Level oder steigende Flanke und die **Quell-Flanke** auf steigende Flanke eingestellt ist.

Benutzen Sie Modus 1 zum Rücksetzen aller CTR Modus-Konfigurations-VI-Parameter auf ihre Standardeinstellung. Dieser Modus macht alle widersprüchlichen Parametereinstellungen unwirksam.

Benutzen Sie Modus 2 zum Zählen von Ereignissen des ausgewählten Signals und zum Stoppen des ersten TCs. Das Überlauf-Statusbit ist auf das TC eingestellt. Benutzen Sie das VI CTR Steuerung zum Lesen des Überlauf-Status. Dieser Modus ist nur für Am9513-Geräte verfügbar. Die Zählung bei Modus 2 ist ungepuffert. In Abbildung 28-1 werden die Zählwerte gezeigt, die Sie mit diesem Modus bei Benutzung von zwei **Gate-Modus**-Einstellungen (High-Level-Gating und Gating mit steigender Flanke) lesen würden.

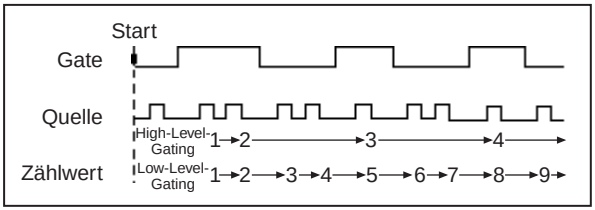


Abbildung 28-1. Ungepufferte Zählung Modus 2 und 3

Benutzen Sie Modus 3 zum kontinuierlichen Zählen von Ereignissen des ausgewählten Signals; läuft der Counter über, wird das TC-Signal ausgegeben und der Counter fährt – wieder bei Null beginnend – fort. Abbildung 28-1 zeigt ungepuffertes Zählen im Modus 3. Abbildung 28-2 stellt eine gepufferte Operation mit Modus 3 unter Gating mit steigender Flanke dar. Diese gepufferte Operation ist nur mit DAQ-STC-Geräten verfügbar. Bei gepufferten Operationen im Modus 3 speichert LabVIEW den derzeitigen Zählwert im Puffer einer jeden ausgewählten Flanke des Quell-Signals.

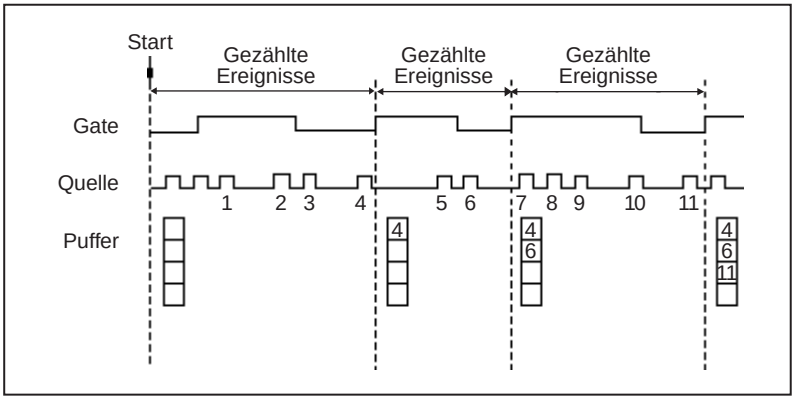


Abbildung 28-2. Gepufferte Zählung Modus 3

Benutzen Sie Modus 4 mit Level-Gating zum Messen von Impulsbreiten und mit Flanken-Gating zum Messen der Periode des ausgewählten Gate-Signals.

☞ **Hinweis** ***Für die folgenden Beschreibungen von Impulsbreitenmessungen (Modi 4, 6 und 7) wird ein hoher Impuls einfach als High-Level-Phase eines Signals definiert, wenn der* Gate-Modus *auf Gating mit High-Level eingestellt ist. Diese Definition unterscheidet sich von einer solchen mit hohem Impuls, bei dem Impulserzeugung benutzt wird (Modus 5), welche aus einer Low-Level-Verzögerungsphase gefolgt von einer High-Level-Impulsphase besteht. (Niedrige Impulse werden durch Auswechseln der Wörter High und Low ähnlich definiert.)***

Setzen Sie den **Gate-Modus** zum Messen der Impulsbreite auf High- oder Low-Level. Abbildung 28-3 zeigt ungepufferte Impulsbreitenmessungen Modus 4. Sie können jederzeit einen Am9513-Counter starten, der Impulse mißt, bis Sie ihn wieder stoppen. Wenn Sie die Messung in der Mitte des Impulses, den Sie messen möchten, starten (z.B. während eines hohen Impulses für High-Level-Gating), gibt LabVIEW für diese Messung eine kurze Zählung zurück. Sie dürfen einen DAQ-STC-Counter nur dann starten, wenn sich das Signal in der entgegengesetzten Polarität vom ausgewählten Gate-Level befindet (z.B. eine Low-Level-Phase für High-Level-Gating). Anderenfalls gibt das VI Fehlernummer 10890 zurück. Beim ungepufferten Zählen stoppt das DAQ-STC die Zählung nach einer Messung. Modus 5 konfiguriert den Counter zur Impulserzeugung. Benutzen Sie das VI CTR Impulskonfiguration, um den Impuls anzugeben, den Sie erzeugen möchten.

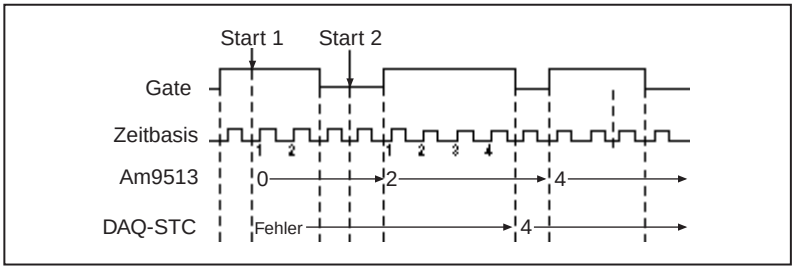


Abbildung 28-3. Ungepufferte High-Impulsbreiten-Messung Modus 4

Abbildung 28-4 zeigt die gepufferte Impulsbreiten-Messung Modus 4, die nur mit DAQ-STC-Geräten verfügbar ist. Die gemessenen Werte werden am Ende eines Impulses im Puffer gespeichert. Modus 6 zeigt eine andere Art der Impulsbreiten-Messung mit einem DAQ-STC-Gerät.

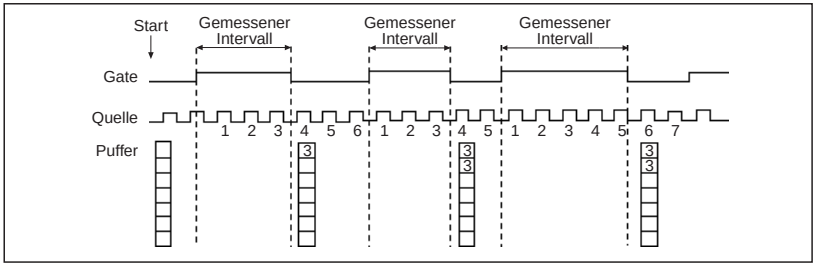


Abbildung 28-4. Gepufferte Messung der Impulsbreite mit steigender Flanke Modus 4

Um die Periodendauer zu messen, müssen Sie den **Gate-Modus** auf steigende oder fallende Flanke einstellen. Abbildung 28-5 zeigt die ungepufferte Impulsbreiten-Messung Modus 4.

Sie können jederzeit entweder einen Am9513- oder einen DAQ-STC-Counter starten. Der Counter beginnt beim Start der nächsten Periode zu zählen. Der Am9513-Counter mißt Perioden kontinuierlich. Beim ungepufferten Zählen stoppt das DAQ-STC die Zählung nach einer Messung.

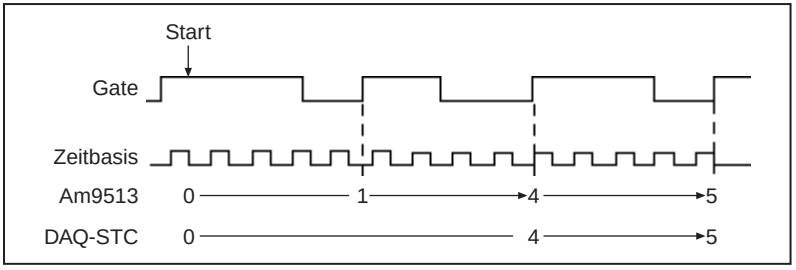


Abbildung 28-5. Ungepufferte Periodenmessung mit steigender Flanke Modus 4

Abbildung 28-6 zeigt eine gepufferte Periodenmessung Modus 4, die nur bei DAQ-STC-Geräten verfügbar ist. Der gemessene Wert wird am Ende einer jeden Periode im Puffer gespeichert.

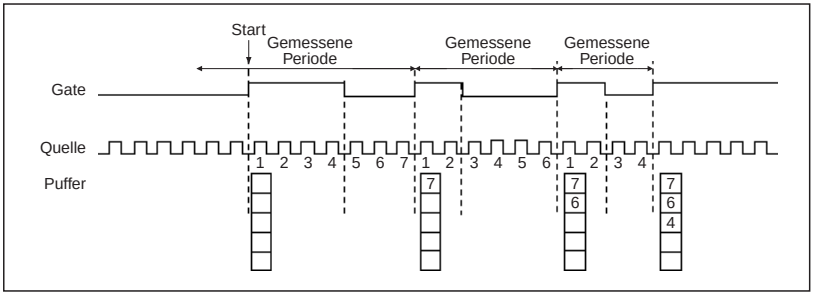


Abbildung 28-6. Gepufferte Impulsbreiten-Messung mit steigender Flanke Modus 4

Benutzen Sie Modus 5 zur Konfiguration der Impulserzeugung, wenn Sie auch den **Gate-Modus, Ausgabe-Typ** oder **Ausgabe-Polarität** auf nicht-standardmäßige Werte einstellen müssen. Anderenfalls sollten Sie den Aufruf des VIs CTR Modus-Konfiguration vermeiden und zur Impulserzeugung nur das VI CTR Impuls-Konfiguration verwenden. In der Beschreibung zum VI CTR Impuls-Konfiguration sind weitere Details zu dieser Operation enthalten.

Benutzen Sie zur Messung der Impulsbreite des ausgewählten Signals Modus 6 mit Level-Gating. Dieser Modus ist nur mit DAQ-STC-Geräten verfügbar. Modus 6 unterscheidet sich von Modus 4 dahingehend, daß die Messung eines hohen (niedrigen) Impulses nicht bis

zur ersten fallenden (steigenden) Signalflanke nach dem Start des Counters beginnt. Wenn Sie ungepuffertes Zählen benutzen, mißt der Counter weiterhin Impulse, bis Sie das VI CTR Steuerung aufrufen, um den zuletzt gemessenen Wert zu lesen, wobei der Counter zu diesem Zeitpunkt stoppt. Ungepuffertes Zählen im Modus 6 ist in Abbildung 28-7 gezeigt.

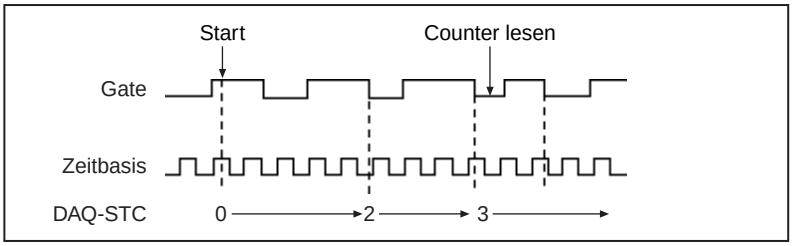


Abbildung 28-7. Ungepufferte hohe Impulsbreiten-Messung Modus 6

Beim gepufferten Zählen Modus 6 wird, wie in Abbildung 28-8 dargestellt, der gemessene Wert im Puffer am Ende eines jeden Impulses gespeichert. Rufen Sie das VI CTR Puffer lesen zum Lesen der Werte auf.

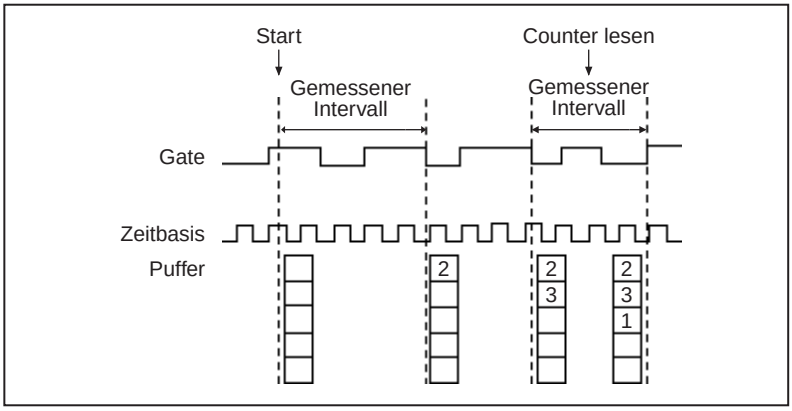


Abbildung 28-8. Gepufferte hohe Impulsbreiten-Messung Modus 6 (Zählung auf steigender Flanke der Quelle)

Benutzen Sie Modus 7 zur Messung einer jeden Phase des ausgewählten Signals, wobei gepuffertes Zählen benutzt wird. Dieser Modus ist nur für DAQ-STC-Geräte verfügbar. Der Zählwert wird bei jedem Low-zu-High- und High-zu-Low-Übergang im Puffer gespeichert. Benutzen Sie zum Lesen der Werte das VI CTR Puffer lesen. Fassen Sie zum Messen der Periodendauer in diesem Modus aufeinanderfolgende Signalpaare zusammen. Zum Messen der Phase können Sie jeden anderen Wert benutzen. LabVIEW ignoriert den Wert des **Gate-Modus** mit Modus 7, was bedeutet, daß Sie nicht feststellen können, ob die erste Messung an einer steigenden oder einer fallenden Flanke beginnt.

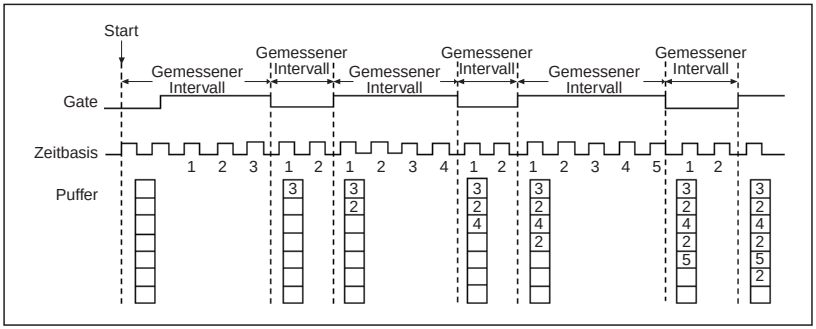


Abbildung 28-9. Gepufferte Halbperioden-Messung Modus 7

Tabelle 28-3 zeigt die zulässigen Werte und Standardeinstellungen für das **Zeitbasis-Signal**. Ein Wert von `-1` weist LabVIEW an, die Standardeinstellungen zu benutzen. Wenn auf der Tabelle Counter angegeben ist, so bezieht sich dies auf den gerade konfigurierten Counter. Wenn mehrere Counter vorhanden sind, konfiguriert LabVIEW Counter nacheinander.

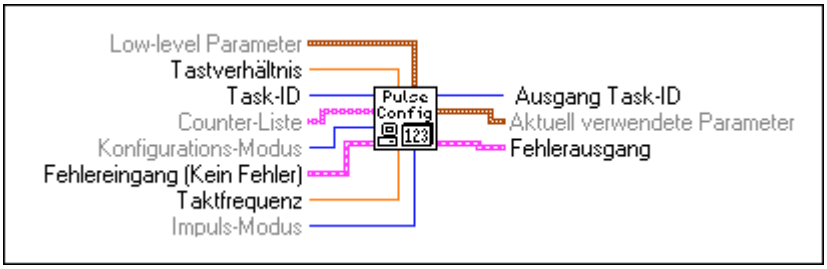
Aus Tabelle 28-3 können Sie ersehen, welches der nächsthöhere oder niedrigere aufeinanderfolgende Counter ist.

Tabelle 28-3. Benachbarte Counter

Gerätetyp	Nächst niedrigerer Counter	Counter	Nächst höherer Counter
Am9513	5	1	2
	1	2	3
	2	3	4
	3	4	5
	4	5	1
	10	6	7
	6	7	8
	7	8	9
	8	9	10
	9	10	6
DAQ-STC	1	0	1
	0	1	0

CTR Impuls-Konfiguration

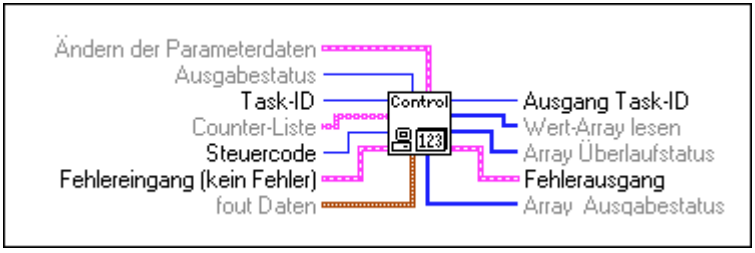
Gibt die Parameter für die Impulserzeugung an. Dieses VI konfiguriert die Counter, startet sie aber nicht. Benutzen Sie zur Erzeugung des Impulses das VI CTR Steuerung mit Steuercode 1 (Start).



Benutzen Sie dieses VI zur Angabe der Charakteristika Ihres Impulses. Sie können auch das VI CTR Modus-Konfiguration zur Einstellung Ihres gewünschten Gate-Modus, Ausgabe-Polarität und Ausgabe-Typs benutzen. Benutzen Sie das VI CTR Impuls-Konfiguration zur Angabe der **Zeitbasis-Quelle** und des **Zeitbasis-Signals** zur Impulserzeugung, da LabVIEW diese im VI CTR Modus-Konfiguration angegebenen Werte ignoriert.

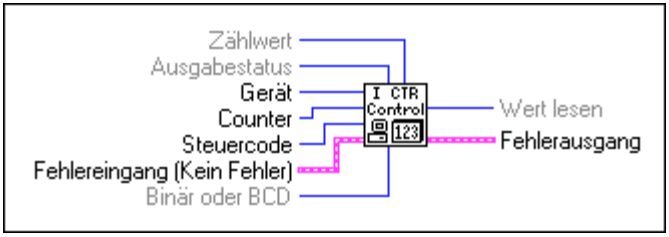
CTR Steuerung

Steuert und liest Counter-Gruppen. Zu den Steuer-Operationen gehören Start, Stoppen und Einstellung des Ausgangsstatus.



ICTR Steuerung

Steuert Counter an Geräte, die 8253-Chips (Lab-Geräte und Geräte der Serie 1200, 516-Geräte PC-LPM-16, DAQCard-500 und DAQCard 700) benutzen.



29

Kalibrierungs- und Konfigurations-VIs

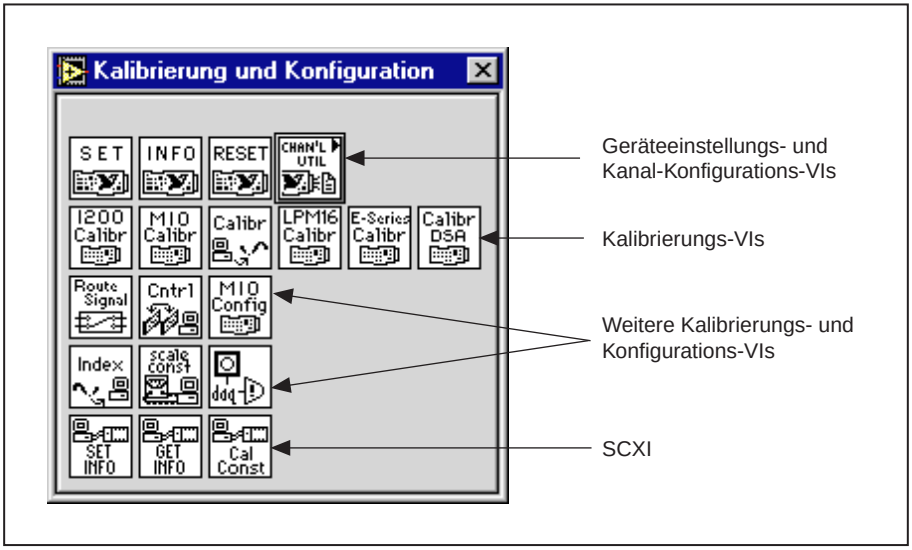
Dieses Kapitel beschreibt die VIs, die bestimmte Geräte kalibrieren und Konfigurationsinformationen einstellen und zurückgeben.

In diesem Kapitel ist auch ein VI für die Steuerung des RTSI-Busses enthalten, bei dem es sich um einen Trigger- und Timing-Bus zur Synchronisation, sowie zum Timing und Triggern mehrerer DAQ-Geräte handelt.

(Windows) Es wird außerdem ein VI behandelt, das Sie zur Einstellung von Vorkommnissen von Datenerfassungs-Ereignissen einstellen können.

Bestimmte DAQ-Geräte können Sie mit den gerätespezifischen VIs kalibrieren, wobei dies nicht immer notwendig ist, weil National Instruments alle Geräte werkseitig kalibriert.

Sie können auf die VIs Kalibrierung und Konfiguration, wie nachstehend gezeigt, durch Auswahl von **Funktionen»Datenerfassung»Kalibrierung und Konfiguration** zugreifen.



Folgende VIs sind nur in der DAQ-VI-Bibliothek vorhanden:

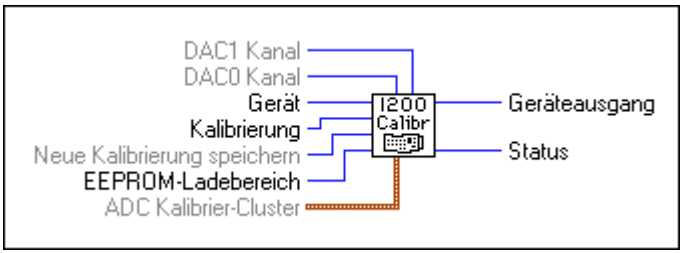
- Kalibrierung A2000
- Konfiguration A2000
- Kalibrierung A2100
- Konfiguration A2100
- Kalibrierung A2150
- Konfiguration A2150
- Kalibrierung DSP 2200
- Konfiguration DSP 2200

Beschreibungen der Kalibrierungs- und Konfigurations-VIs

Folgende Kalibrierungs- und Konfigurations-VIs sind verfügbar.

Kalibrierung 1200

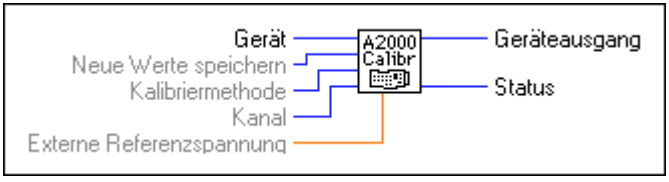
Dieses VI kalibriert Verstärkungs- und Offset-Werte für die ADCs und DACs an Geräten der Serie 1200 (d.h., DAQPad-1200, DAQCard-1200 usw).



Sie können eine neue Kalibrierung durchführen (und wahlweise die neuen Kalibrierungskonstanten in einer von vier Benutzerbereichen im Onboard-EEPROM speichern) oder ein bestehendes Set von Kalibrierungskonstanten durch Kopieren von deren Speicherplatz in den Onboard-EEPROM laden. LabVIEW lädt die im Onboard-EEPROM-Ladebereich gespeicherten Kalibrierungskonstanten automatisch, wenn LabVIEW gestartet wird oder wenn Sie das Gerät zurücksetzen. Der EEPROM-Ladebereich enthält standardmäßig eine Kopie der Kalibrierungskonstanten im Werksbereich.

Kalibrierung A2000

Kalibriert die NB-A2000- oder EISA-A2000 A/D-Verstärkungs- und Offset-Werte oder stellt deren ursprüngliche werkseitig eingestellte Werte wieder her.



Sie können Ihr NB-A2000- oder EISA-A2000-Gerät kalibrieren, um die Genauigkeit der Ablesungen von den vier Analogeingangs-Kanälen einzustellen. LabVIEW lädt die gespeicherten Kalibrierungswerte automatisch, wenn es gestartet wird oder wenn Sie Ihr NB-A2000- oder EISA-A2000-Gerät zurücksetzen.

Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI folgende Mitteilung zurück:*** `deviceSupportError.` ***Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zu NB-A2000- oder EISA-A2000-DAQ-Geräte.

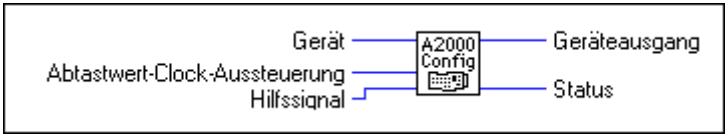
Warnung ***Lesen Sie zur Benutzung des VIs A2000 Kalibrierung die Kapitel über Kalibrierung im*** *NB-A2000- oder EISA-A2000-Benutzerhandbuch.*

Wenn Sie **Neue Werte speichern** auf `1` einstellen, speichert dieses VI die Verstärkungs- und Offset-Kalibrierungswerte in einem EEPROM auf dem NB-A2000- oder EISA-A2000-Gerät, bei dem selbst bei Stromausfall kein Datenverlust eintritt. LabVIEW liest diese EEPROM-Werte und lädt sie in das NB-A2000- oder EISA-A2000-Gerät, das Sie auswählen können, um die permanenten Kopien der Verstärkungs- und Offset-EEPROM-Werte zu ersetzen, und Sie können die neuen Werte bis zur nächsten Kalibrierung benutzen, selbst wenn Sie das Gerät neu initialisieren. Sie können auch entscheiden, die EEPROM-Werte nicht zu ersetzen, sondern die neuen Werte bis zur nächsten Kalibrierung oder Initialisierung zu benutzen.

Wenn Sie z.B. nach dem Rücksetzen des Gerätes ständig ungenaue Ablesungen von einem oder mehrere Eingangskanälen erhalten, können Sie die neuen Verstärkungs- und Offset-Werte als permanente Kopien im EEPROM speichern. Wenn die Erfassungsergebnisse jedoch nach der Initialisierung stimmen, nach einigen Stunden der Geräteoperation abzuweichen beginnen, wenn sich die Temperatur des Geräts erhöht, können Sie das Gerät auf diese Betriebstemperatur kalibrieren und die derzeitigen EEPROM-Werte zur Benutzung nach der nächsten Initialisierung aufrechterhalten.

Konfiguration A2000

Konfiguriert Hilfssignale und ob die SAMPCLK*-Leitung für die NB-A2000 -oder EISA-A2000-Geräte ausgesteuert werden soll.



Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI daher folgende Mitteilung zurück:*** `deviceSupportError.` ***Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

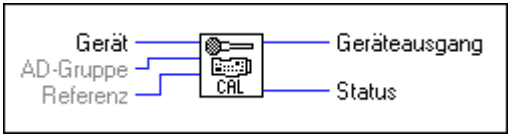
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zu den NB-A2000- oder EISA-A2000-DAQ-Geräten.

Nach dem Systemstart konfiguriert LabVIEW die NB-A2000- oder EISA-A2000-Geräte wie folgt.

- **Abtastwert-Clock-Aussteuerung**= `0`:Abtast-Clock-Signal steuert SAMPCLK*-Leitung nicht aus.
- **Hilfssignal**= 0:Hilfssignal deaktiviert.

Kalibrierung A2100

Wählt die gewünschte Kalibrierungsreferenz und führt einen Offset-Kalibrierungszyklus an ADCs an den NB-A210- oder den NB-A2150-Geräten durch.



Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI daher folgende Mitteilung zurück:*** `deviceSupportError`***. Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

NI-DAQ-Treibersoftware kalibriert die beiden A/D-Kanäle und benutzt dazu die Analog-Eingangserdung als Referenz für jeden Kanal, wenn Sie den Computer einschalten.

Konfiguration A2100

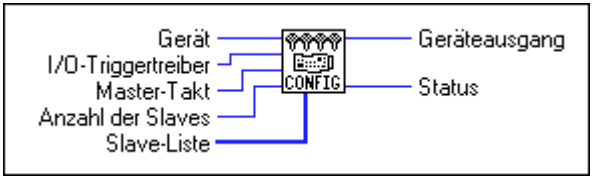
Wählt die Signalquelle aus, die zur Versorgung der DACs mit Daten benutzt wird, und erlaubt Ihnen, den externen Digitaltrigger zu konfigurieren, der von der Datenerfassung und der Signalerzeugungs-Operation am NB-A2100 gemeinsam benutzt wird.

Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI daher folgende Mitteilung zurück:*** `deviceSupportError`***. Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

Wenn LabVIEW mehrere Datenerfassungsrahmen erfaßt und mehrere Signalzyklen mit einem am Anfang eines jeden Zyklus erforderlichen Trigger erzeugt, wird die externe Triggererkennung synchronisiert, so daß jeder Trigger die Erfassung des nächsten Datenrahmens gleichzeitig initialisiert, während die Ausgabe des nächsten Signalzyklus erzeugt wird.

Konfiguration A2150

Wählt aus, ob LabVIEW einen intern erzeugten Trigger an den NB-A2150-I/O-Anschluß aussteuert. Dieses VI bestimmt auch, ob LabVIEW das NB-A2150-Abtastwert-Clock-Signal über den RTSI-Bus an andere Geräte für eine synchronisierte Datenerfassung für Mehrfachgeräte aussteuert.



Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI daher folgende Mitteilung zurück:*** `deviceSupportError`***. Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

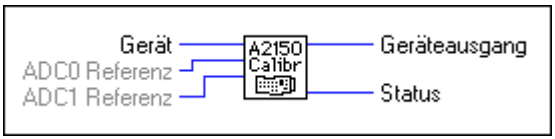
Aktivieren Sie **I/O-Triggertreiber** nur, wenn Sie das VI RTSI Steuerung zum Erhalt eines `RTSITRIG*`-Signals über den RTSI-Bus ausgeführt haben, oder wenn Sie den Analog-Level-Trigger unter Benutzung des VIs AI Trigger-Konfiguration ausgeführt haben. In diesen Fällen können Sie das an die A/D-Triggerschaltung gesendete Signal nach dem Start der Erfassung an der `EXTTRIG*`-Leitung des I/O-Anschlusses überwachen. Die Datenerfassung wird durch eine High-zu-Low-Flanke getriggert.

Das NB-A2150-Gerät benutzt Signale über den RTSI-Bus zur Abtast-Clock-Synchronisation zwischen zwei und mehreren NB-A2150-Geräten. Die Abtast-Clock-Synchronisationsschaltung macht die gleichzeitige Abtastung an mehr als vier Kanälen unter Benutzung zusätzlicher NB-A2150-Geräte möglich. Wenn die **Master-Takt** `1` ist, muß die **Slave-Liste** die Liste der Geräte enthalten, die die Abtast-Takt vom **Gerät** akzeptieren. Nachdem Sie die A2150 Konfiguration mit **Master-Takt** gleich `1` und **der Anzahl der Slaves** größer als `0` ausgeführt haben, können Sie AI Clock-Konfiguration zur Einstellung der Abtastrate für Geräte in der **Slave-Liste** solange nicht benutzen, bis Sie A2150 Konfiguration wieder auf dem **Gerät** mit **Master-Clock** gleich `1` und der **Anzahl der Slaves** gleich `0` ausgeführt haben.

Hinweis ***Die Ausführung der A2150 Konfiguration mit Master-Clock gleich 1 und der Anzahl der Slaves gleich 0 dekonfiguriert die vorher in der Slave-Liste enthaltenen Geräte und stellt sie so ein, daß sie ihre eigenen Abtast-Clock-Signale benutzen.***

A2150 Kalibrierung (Macintosh)

Führt an den ADCs der genannten AT-A2150-Geräte Offset-Kalibrierungen durch.



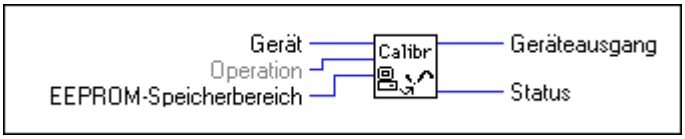
Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI folgende Mitteilung zurück:*** `deviceSupportError.` ***Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zu den NB-A2000- oder EISA-A2000-DAQ-Geräten.

Wenn Sie LabVIEW starten oder wenn Sie AT-A2150 zurücksetzen, führt LabVIEW unter Benutzung der Analog-Erdung als Referenz eine Offset-Kalibrierung durch. Benutzen Sie dieses VI nur zur Gerätekalibrierung an eine externe Referenz oder zur Geräte-Neukalibrierung zur Erdungsreferenz nach Benutzung einer externen Referenz.

AO-6/10 Kalibrierung (Windows)

Lädt ein Set von Kalibrierungskonstanten in die Kalibrierungs-DACs oder kopiert ein Set von Kalibrierungskonstanten von einem von vier EEPROM-Bereichen auf EEPROM-Bereich 1.



Sie können ein bestehendes Set von Kalibrierungskonstanten von einem Speicherbereich im Onboard-EEPROM in die Kalibrierungs-DACs laden. Sie können EEPROM-Speicherbereiche 2 mit 5 auf Speicherbereich 1 kopieren. EEPROM-Bereich 5 enthält die Kalibrierungskonstanten vom Werk. LabVIEW lädt die im EEPROM-Bereich 1

gespeicherten Kalibrierungskonstanten automatisch beim Start oder wenn Sie das AT-AO-6/10-Gerät zurücksetzen.

☞ **Hinweis** ***Sie können auch die mit dem AT-AO-6/10-Gerät gelieferte Kalibrierungs-Utility zur Durchführung einer Kalibrierung benutzen. Im Kapitel zur Kalibrierung im AT-AO-6/10-Benutzerhandbuch sind weitere Informationen enthalten.***

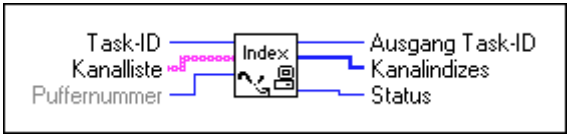
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zu AT-AO-6/10-DAQ-Geräten.

Wenn LabVIEW das AT-AO-6/10-Gerät initialisiert, liefern die in der **EEPROM-Position** 1 (Benutzerkalibrierungs-Bereich 1) gespeicherten DAC-Kalibrierungskonstanten die Verstärkungs- und Offset-Werte, die die ordnungsgemäße Geräteoperation sicherstellen. Somit ist diese Initialisierung mit der Ausführung des AO-6/10 Kalibrierungs-VIs mit **Operation** auf `1` und **EEPROM-Position** auf `1`eingestellt gleichzusetzen. Wenn das AT-AO-6/10-Gerät das Werk verläßt, enthält die **EEPROM-Position** 1 eine Kopie der in **EEPROM-Position** 5 (Werkskalibrierung) gespeicherten Kalibrierungskonstanten.

Eine im bipolaren Modus durchgeführte Kalibrierungsprozedur ist im unipolaren Modus ungültig und umgekehrt. Im Kapitel *Calibration* des *AT-AO-6/10 User Manuals* sind weitere Informationen enthalten.

Kanal an Index

Benutzt die derzeitige Gruppenkonfiguration für die angegebene Aufgabe, um eine Liste von Indizes in der Abtast- oder Upgrade-Liste der Gruppe für jeden in der Kanalliste angegebenen Kanal zu erstellen.



Sie können diese Liste von Kanalindizes zum Auffinden von Daten für einen bestimmten Kanal innerhalb eines Mehrfach-Kanalpuffers benutzen. Sie können die Indizes auch zum Lesen oder Schreiben an ein Gruppen-Subset mit den VIs Puffer lesen und schreiben verwenden.

In Ihrer gerätespezifischen Information in Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die auf Ihr Gerät zutreffenden Kanalbegrenzungen.

Tabelle 29-1 zeigt mögliche Werte für die **Kanal-Abtastliste,- Kanalliste-** und **Kanalindizes-**Parameter. Tabelle 29-2 zeigt mögliche Werte für Sun. Der **Kanal-Abtastlisten-**Parameter ist eine Eingabe für die VIs Gruppen-Konfiguration.

Tabelle 29-1. Kanal an Index VI Parameter-Beispiele

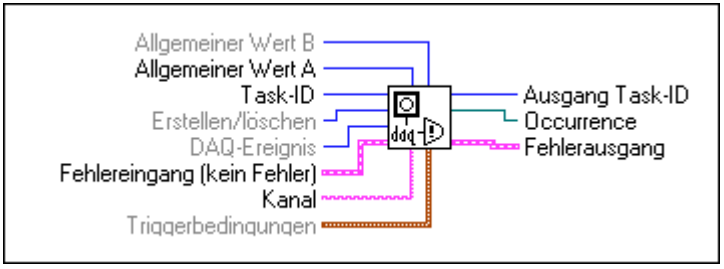
Kanal-Abtastlisten	Kanallisten	Kanalindizes
1, 3, 4, 5, 7	**Kanalliste**[0] = 5	**Kanalindizes**[0] = 3. Daten für Kanal 5 sind bei Position 3 innerhalb einer Abtastung. Indizes basieren auf Null.
1, 3, 4, 5, 7	**Kanalliste** hat die Länge 0.	**Kanalindizes** hat die Länge 0. (In diesem Fall ist der Status nicht Null.)
1, 2, 1, 3, 1, 4 (Das Gerät tastet Kanal 1 während einer Abtastung dreimal ab.)	**Kanalliste**[0] = 1, 1, 1	**Kanalindizes**[0] = 0, **Kanalindizes**[1] = 2 und **Kanalindizes**[2] = 4. Das erste Vorkommen von Kanal 1 innerhalb einer Abtastung ist bei Index 0, das zweite bei Index 2 und das dritte bei Index 4.
0, 1, 3, 4 (Bei diesem Beispiel ist die **Kanal-Abtastliste** eine digitale Ausgabegruppe.)	**Kanalliste**[0] = 3	**Kanalindizes**[0] = 2. Die acht Datenbits von Anschluß 3 sind bei Index 2 in der Abtastliste.
0:3 (Ein AMUX-64T in Benutzung.)	**Kanalliste**[0] = AM1!9	**Kanalindizes**[0] = 9. Daten von Kanal 9 auf dem AMUX-64T-Gerät Nummer 1 sind bei Index 9 im Datenpuffer.
SC1!MD1!CH0:7, SC1!MD2!CH0:4	**Kanalliste**[0] = SC1!MD2!CH3	**Kanalindizes**[0] = 11. Daten von Kanal 3 vom SCXI-Modul in Slot 2 sind in Index 11 im Datenpuffer.

Tabelle 29-2. Kanal an Index VI Parameter-Beispiele für Sun

Kanal-Abtastliste	Kanalliste	Kanalindizes
1, 3, 4, 5, 7	**Kanalliste**[0] = 5	**Kanalindizes**[0] = 3. Daten für Kanal 5 sind bei Position 3 innerhalb einer Abtastung. Indizes basieren auf Null.
1, 3, 4, 5, 7	**Kanallisten** haben die Länge 0.	**Kanalindizes** haben die Länge 0. (In diesem Fall ist der Status nicht Null.)
1, 2, 1, 3, 1, 4 (Das Gerät tastet Kanal 1 während eines Scans dreimal ab.)	**Kanalliste**[0] = 1, 1, 1	**Kanalindizes**[0] = 0, **Kanalindizes**[1] = 2und **Kanalindizes**[2] = 4. Das erste Vorkommen von Kanal 1 innerhalb einer Abtastung ist bei Index 0, das zweite bei Index 2 und das dritte bei Index 4.

DAQ Ereignis-Konfiguration (Windows)

Erzeugt Occurrences, die durch Datenerfassungsereignisse festgelegt werden.



Ein DAQ-Ereignis kann der Abschluß einer Erfassung sein, die Erfassung einer bestimmten Anzahl von Abtastungen, ein Analog-Signal, das auf bestimmte Triggerbedingungen trifft, ein periodisches Ereignis, ein aperiodisches (extern ausgesteuertes) Ereignis oder eine digitale Muster-Übereinstimmung oder -Abweichung derselben sein. Ihr VI kann während des Wartens auf die Einstellung einer Occurrence schlafen, wodurch Ihr Computer für die Ausführung anderer VIs freigemacht wird.

Wenn Sie die Steuerung **Erstellen/löschen** auf 1 (erstellen) einstellen und das VI aufrufen, erstellt dieses VI eine Occurrence. Benutzen Sie die Steuerung **DAQ-Ereignis** zur Auswahl des Ereignisses, das die Occurrence einstellt. Verbinden Sie die von diesem VI produzierte Occurrence mit der Funktion Warten auf Occurrence. Alles, was Sie mit der Funktion Warten

auf Occurrence verbinden, wird nicht ausgeführt, bis die Occurrence eingestellt ist. Die Occurrence wird jedes Mal eingestellt, wenn das Ereignis eintritt. Die Occurrence wird nicht gelöscht, bis Sie die Steuerung **Erstellen/löschen** auf `0` (löschen) einstellen und dieses VI oder das VI Gerät zurücksetzen für das Gerät aufrufen.

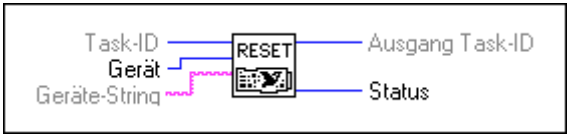
LabVIEW gibt eine Refnum-Datei-I/O-Konstante zusammen mit einem Nicht-Null-Statuscode zurück, wenn es das Vorkommnis nicht erstellen kann.

LabVIEW begrenzt für jede Plattform die Anzahl der Occurrences pro Sekunde, die Sie einstellen können. Obwohl diese Begrenzung von der Schnelligkeit Ihres Computers abhängt, sollten Sie es vermeiden, 500 Occurrences pro Sekunde zu überschreiten.

Für einige Ereignisse müssen Sie für die Ausführung Ihrer Operation anstelle von DMA Interrupts benutzen. In der Beschreibung **DAQ-Ereignis**-Steuerung in diesem Abschnitt finden Sie weitere Informationen.

Gerät zurücksetzen

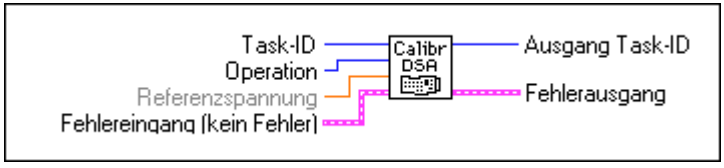
Setzt entweder ein Gerät als Gesamtheit oder eine durch **Task-ID** identifizierte Funktion zurück.



Das Zurücksetzen einer **Task-ID**-Funktion hat dasselbe Ergebnis wie der Aufruf des VIs Steuerung für diese Funktion mit **Steuercode** auf Löschen eingestellt. Wenn Sie das gesamte Gerät zurücksetzen, löscht LabVIEW alle Tasks und ändert alle Geräteeinstellungen auf ihre Standardwerte.

Kalibrierung DSA

Benutzen Sie dieses VI zur Kalibrierung Ihres DSA-Geräts.



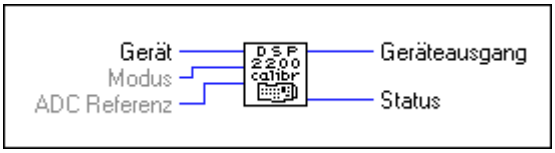
Ihr Gerät verfügt über Kalibrierungs-D/A-Konverter (calDACs), die die Analogschaltung genaustens abstimmen. Die calDACs müssen mit bestimmten Zahlen, genannt Kalibrierkonstanten, programmiert (geladen) werden. Diese Konstanten sind auf Ihrem Gerät in nichtflüchtigem Speichern (EEPROM) gespeichert. Führen Sie zur Erlangung von

Spezifikationsgenauigkeit kurz vor einer Meßsession, aber nachdem Ihr Computer und das Gerät mindestens 15 Min. eingeschaltet waren, eine interne Kalibrierung Ihres Geräts durch. Eine häufige Kalibrierung erzeugt die stabilsten und am besten wiederholbaren Meßleistungen.

Vor dem Versand des Geräts vom Werk wird eine externe Kalibrierung durchgeführt, und EEPROM enthält Kalibrierkonstanten, die LabVIEW nach Bedarf automatisch in die calDACs lädt. Der Wert der Onboard-Referenzspannung wird auch in EEPROM gespeichert; dieser Wert wird bei jeder späteren Durchführung einer Selbstkalibirerung benutzt. Die Kalibrierkonstanten werden bei Durchführung einer Selbstkalibrierung neu errechnet und in EEPROM gespeichert. Wenn Sie eine externe Kalibrierung durchführen, errechnet LabVIEW den Wert der Onboard-Referenzspannung neu und nimmt dann eine Selbstkalibrierung vor. Dieser neue Onboard-Referenzwert wird für alle nachfolgenden Selbstkalibrierungsoperationen benutzt. Wenn bei der Durchführung einer externen Kalibrierung ein Fehler vorkommt, können Sie die werkseitige Kalibrierung der Karte wiederherstellen, so daß die Karte benutzbar bleibt.

DSP2200 Kalibrierung (Windows)

Führt an dem Analog-Eingang und/oder Analog-Ausgang von AT-DSP2200 Offset-Kalibrierungen durch.



Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI folgende Mitteilung zurück:*** `deviceSupportError.` ***Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

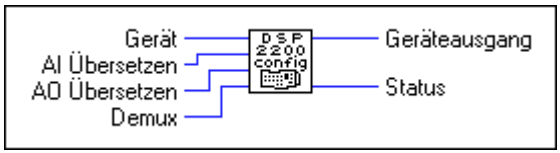
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zum AT-DSP2200-DAQ-Gerät.

Wenn Sie LabVIEW starten oder das AT-DSP2200-Gerät zurücksetzen, führt LabVIEW sowohl am Analog-Eingang wie auch am Analog-Ausgang eine Offset-Kalibrierung durch und benutzt die Analog-Erde als Referenz.

Sie können dieses VI zur Kalibrierung eines Analog-Eingangs unter Benutzung externer Referenzen oder zur Neukalibrierung des AT-DSP2200-Geräts zur Kompensation für Konfigurations- oder Umgebungsänderungen benutzen.

DSP2200 Konfiguration (Windows)

Spezifiziert die Daten-Übersetzung- und Demultiplexierungsoperationen, die das AT-DSP2200-Gerät an Analog-Eingangs- und Ausgangs-Daten durchführt.



Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI folgende Mitteilung zurück:*** `deviceSupportError.` ***Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

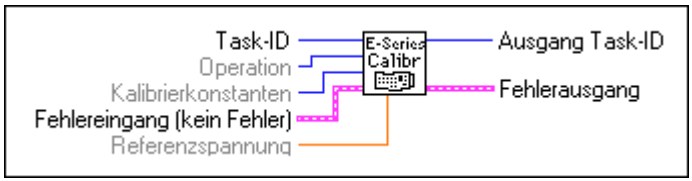
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen für das AT-DSP2200-DAQ-Gerät.

Da lokal auf dem AT&T WE DSP32C DSP-Chip ausgeführte Software Daten von den ADCs liest und Daten zu den DACs schreibt, können Sie die Daten während dieser Übertragungen manipulieren. Wenn Sie Analog-Eingangsdaten zum DSP-Speicher schreiben, können Sie die Daten als nicht-skalierte 16-Bit-Integer-Werte, unskalierte 32C-Fließkomma-Zahlen oder skalierte 32C-Fließkomma-Spannungen schreiben. Sie können die **Demux**-Option nur benutzen, wenn Sie Analog-Eingangsdaten zum DSP-Speicher schreiben. Wenn Sie **Demux** aktivieren, schreibt das Gerät nacheinander Daten in den DSP-Speicher, wobei es am Start eines jeden Puffers beginnt und Daten des Kanals 1 nacheinander schreibt, wobei es am Halbzeitpunkt eines jeden Puffers beginnt. Wenn das Gerät Analog-Eingangsdaten an den PC-Speicher schreibt, kann es die Daten als unskalierte 16-Bit-Integer, unskalierte IEEE-Einzelpräzisions-Fließkomma-Zahlen oder als skalierte IEEE-Einzelpräzisions-Spannungen schreiben.

Die Analog-Ausgangsübersetzungen sind in gegengesetzter Richtung zu Analog-Eingangsübersetzungen. Wenn **AO Übersetzen** `0` ist, müssen sich die Quell-Daten in einem für die DACs geeignetem Format befinden (16-Bit Integer DAC-Werte). Wenn **AO Übersetzen** `1` oder `3` ist, sind die Quell-Daten DAC-Werte im 32C-Format im DSP-Speicher oder im IEEE-Einzelpräzisions- Format im PC-Speicher. Wenn **AO Übersetzen** `2` oder `4` ist, handelt es sich bei den Quell-Daten um Spannungen im 32C-Format oder im IEEE-Einzelpräzisions-Format im PC-Speicher.

Kalibrierung E-Serie

Benutzen Sie dieses VI zur Kalibrierung Ihres Gerätes der E-Serie und zur Auswahl eines Satzes Kalibrierungskonstanten zur Benutzung durch LabVIEW.



Warnung ***Lesen Sie vor Benutzung des VIs Kalibrierung E-Serie das Kapitel über Kalibrierung in Ihrem Benutzerhandbuch.***

Ihr Gerät enthält Kalibrier-D/A-Konverter (calDACs), die zur Feinabstimmung der Analogschaltung verwendet werden. Die calDACs müssen mit bestimmten Nummern programmiert (geladen) sein, die *Kalibrierkonstante* genannt werden. Diese Konstanten werden in nichtflüchtigen Speichern (EEPROM) auf Ihrem Gerät gespeichert oder von LabVIEW instandgehalten. Zum Erhalt der Spezifikationsgenauigkeit sollten Sie eine interne Kalibrierung Ihres Geräts unmittelbar vor der Messung, aber nachdem Ihr Computer und das Gerät eingeschaltet wurden und sich mindestens 15 Minuten aufwärmen konnten, durchführen. Eine häufige Kalibrierung erzeugt die stabilste und am besten wiederholbare Leistung. Das Gerät wird in keiner Weise beschädigt, wenn Sie es so oft Sie möchten kalibrieren.

Im EEPROM können zwei Sätze Kalibrierkonstanten in zwei Bereichen, *Ladebereiche* genannt, vorhanden sein. Ein Konstantensatz wird werkseitig programmiert; die Programmierung des anderen Satzes wird dem Benutzer überlassen. Ein Ladebereich in EEPROM entspricht einem Konstantensatz. Der Ladebereich, den LabVIEW zum Laden von calDACs Kalibrierkonstanten verwendet, wird Standardladebereich genannt. Wenn das Gerät vom Werk geliefert wird, befindet sich der Standardladebereich in dem Bereich, der die durch die werkseitige Kalibrierung erlangten Kalibrierkonstanten enthält. LabVIEW lädt beim ersten Aufruf eines VIs, das die Kalibrierkonstanten benötigt, automatisch die im Ladebereich gespeicherten relevanten Kalibrierkonstanten.

Hinweis ***Die Kalibrierung Ihrer Geräte der E-Serie nimmt geraume Zeit in Anspruch. Seien Sie daher nicht überrascht, wenn das VI für die Ausführung einige Sekunden benötigt.***

Warnung ***Wenn Sie dieses VI ausführen und* Operation *ist auf Selbstkalibrierung oder externe Kalibrierung eingestellt, bricht LabVIEW jegliche gerade laufenden Operationen des Geräts ab und setzt alle Konfigurationen auf Ihre Standardwerte. Deshalb sollten Sie dieses VI vor allen anderen DAQ-VIs ausführen oder wenn keine anderen Operationen ausgeführt werden.***

12-Bit-Geräte E-Serie

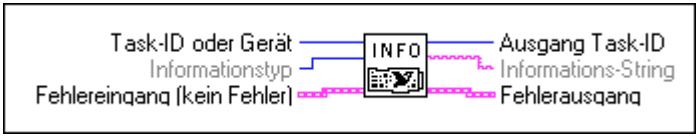
- Verbinden Sie den positiven Ausgang Ihrer Referenz-Spannungsquelle mit dem Analog-Eingabekanal 8.
- Verbinden Sie den negativen Ausgang Ihrer Referenz-Spannungsquelle mit der AISENSE-Leitung.
- Verbinden Sie die DAC0-Leitung (Analog-Ausgabekanal 0) mit Analog-Eingabekanal 0.
- Wenn Ihre Referenz-Spannungsquelle und Ihr Computer im Verhältnis zueinander schwebend sind, müssen Sie die AISENSE-Leitung sowohl mit der AIGND-Leitung als auch mit der negativen Ausgabe Ihrer Referenz-Spannungsquelle verbinden.

16-Bit-Geräte E-Serie

- Verbinden Sie den positiven Ausgang Ihrer Referenz-Spannungsquelle mit dem Analog-Eingabekanal 0.
- Verbinden Sie die negative Ausgabe Ihrer Referenz-Spannungsquelle mit dem Analog-Ausgabekanal 8 (durch diese beiden Verbindungen wird der für differentielle Operation konfigurierte Analog-Eingabekanal 0 mit der Referenzspannung versorgt.)
- Wenn Ihre Referenz-Spannungsquelle und Ihr Computer im Verhältnis zueinander schwebend sind, müssen Sie die negative Ausgabe Ihrer Referenz-Spannungsquelle sowohl mit der AIGND-Leitung als auch mit dem Analog-Eingabekanal 8 verbinden.

DAQ-Geräteinformation erlangen

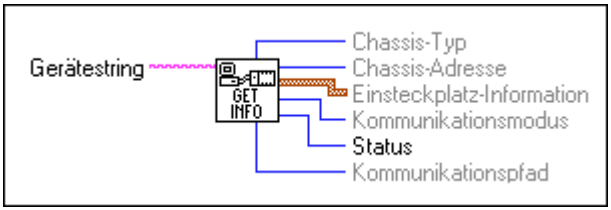
Gibt Informationen über ein DAQ-Gerät zurück.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Übertragungsmethoden.

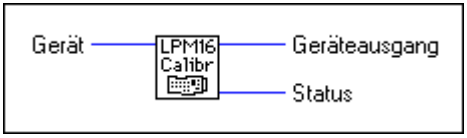
SCXI-Information erlangen

Gibt die SCXI-Chassis-Information mit Hilfe der Konfigurations-Utility oder dem VI SCXI-Information zurück.



LPM-16 Kalibrierung

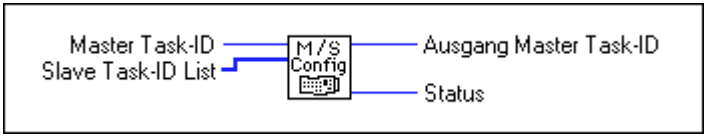
Kalibriert den PC-LPM-16- oder PC-LPM-16PnP-Konverter. Die Kalibrierung errechnet die korrekte Offset-Spannung für den Spannungskomparator, gleicht die positive Linearität und Skalenendfehler auf jeweils weniger als ±0,5 LSB an und gleicht Nullfehler auf weniger als ±1 LSB an.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zu den PC-LPM-16,- DAQCard-500- oder DAQCard-700-Geräten.

Master-Slave-Konfiguration

Konfiguriert ein Gerät als Master-Gerät und andere Geräte als Slave-Geräte für mehrfach gepufferte Analog-Eingabeoperationen.



Warnung ***Dieses VI wird nur bis zur NI-DAQ Version 4.9.0 unterstützt und wurde von der Kalibrierungs- und Konfigurations-Palette entfernt. Dieses VI ist ausschließlich aus Kompatibilitätsgründen in der DAQ-VI-Bibliothek enthalten. Wenn Sie die NI-DAQ Version 5.0 oder eine spätere Version benutzen, gibt dieses VI folgende Mitteilung zurück:*** `deviceSupportError.` ***Wenn Sie dieses VI benutzen möchten, müssen Sie die NI-DAQ Version 4.9.0 oder eine frühere Version neu installieren.***

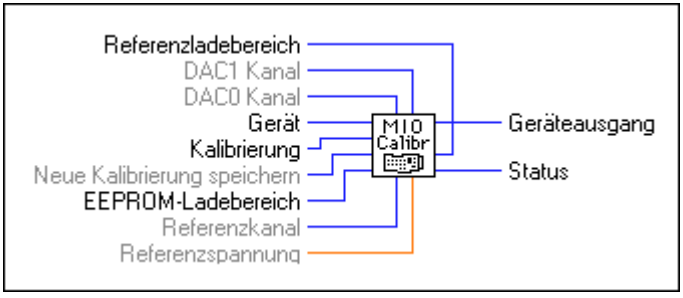
Vergewissern Sie sich, daß LabVIEW in einer Mehrfach-Puffer-Analog-Eingabeoperation die Slave-Geräte immer *vor* dem Master-Gerät erneut aktiviert. Nur folgende Geräte, die mehrfach gepufferte Erfassungen unterstützen, können dieses VI benutzen.

- **(Macintosh)** NB-A2000, NB-A2100 und NB-A2150.

Das Master-Gerät sendet zur Steuerung der Abtastung ein Trigger- oder Clock-Signal an das (die) Slave-Gerät(e). Bei einer Mehrfach-Puffer-Erfassung müssen Sie das Slave-Gerät vor dem Master-Gerät aktivieren, um sicherzustellen, daß das Slave-Gerät immer auf ein Master-Signal antwortet. Wenn Sie das Master-Gerät zuerst aktivieren, kann dieses ein Signal an die Slave-Geräte senden, bevor diese antworten können. Sie sind für die anfängliche Start-Reihenfolge verantwortlich. Starten Sie das Master-Gerät immer zuletzt. Das VI Master-Slave-Konfiguration stellt sicher, daß LabVIEW das Master-Gerät nach jedem nachfolgenden, während der Mehrfach-Puffer-Erfassung erfaßten Puffer, ausrüstet.

MIO-Kalibrierung (Windows)

Kalibriert die AT-MIO-16F-5-, AT-MIO-64F-5- und AT-MIO-16X-Verstärkungs- und Offset-Werte für die ADCs und DACs. Sie können entweder eine neue Kalibrierung durchführen oder einen bestehenden Satz von Kalibrierkonstanten durch Kopieren der Konstanten von ihrem Speicherplatz in das Onboard-EEPROM benutzen. Sie können mehrere Sätze Kalibrierkonstanten speichern. LabVIEW lädt während des Starts des Geräts oder wenn Sie das Gerät zurücksetzen automatisch die im EEPROM-Ladebereich gespeicherten Kalibrierkonstanten



Der Ladebereich für AT-MIO-16F-5 ist Benutzerbereich 5. Der Ladebereich für AT-MIO-64F-5 und AT-MIO-16X ist Benutzerbereich 8.

Warnung ***Lesen Sie vor Benutzung des VIs MIO-Kalibrierung das Kapitel Kalibrierung in Ihrem Geräte-Benutzerhandbuch.***

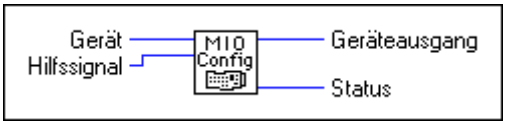
In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zu den Geräten AT-MIO-16F-5, AT-MIO-64F-5 und AT-MIO-16X DAQ.

Hinweis ***Kalibrieren Sie das ADC und die DACs immer nach dem Kalibrieren der internen Referenzspannung.***

☞ **Hinweis** ***Wenn das Gerät analoge Eingabemessungen mit dem falschen Satz geladener Kalibrierkonstanten aufnimmt, können Sie falsche Daten erhalten.***

MIO-Konfiguration (Windows)

Schaltet Hilfssignale an und ab. Dieses VI unterstützt folgende Geräte: AT-MIO-16F-5, AT-MIO-64F-5, alle12-Bit-Geräte der E-Serie und alle Geräte der Serie 1200.

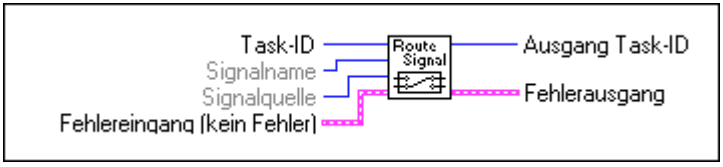


In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie weitere Informationen zu Geräten, die von diesem VI unterstützt werden.

Signal verschalten

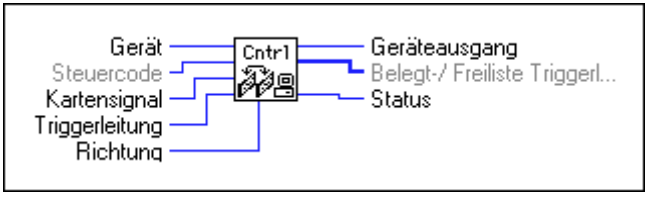
Benutzen Sie dieses VI zum Verschalten interner Signale an den genannten I/O-Anschluß oder die RTSI-Busleitung oder zum Aktivieren von gemeinsam benutzten Clocks durch die RTSI-Bus-Taktleitung.

☞ **Hinweis** ***Dieses VI wird nur von Geräten der E-Serie und 54XX unterstützt.***



RTSI-Steuerung

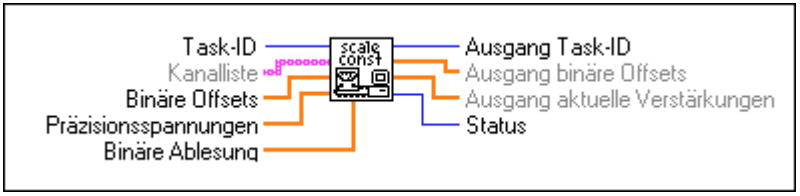
Verbindet und trennt Trigger- und Timing-Signale zwischen DAQ-Geräten zusammen mit dem Real-Time System Integration (RTSI)-Bus (Bus zur Echtzeit-Systemintegration).



Dieses VI wird für Geräte der E-Serie nicht unterstützt. Bei Geräten der E-Serie können Mehrfach-RTSI-Verbindungen direkt in der Analogeingabe, Analogausgabe und Counter-VIs eingestellt werden und zusammen mit dem VI Signal verschalten benutzt werden. Andere RTSI-Verbindungen müssen mit dem VI Signal verschalten vorgenommen werden.

Abstimmen der Skalierkonstante

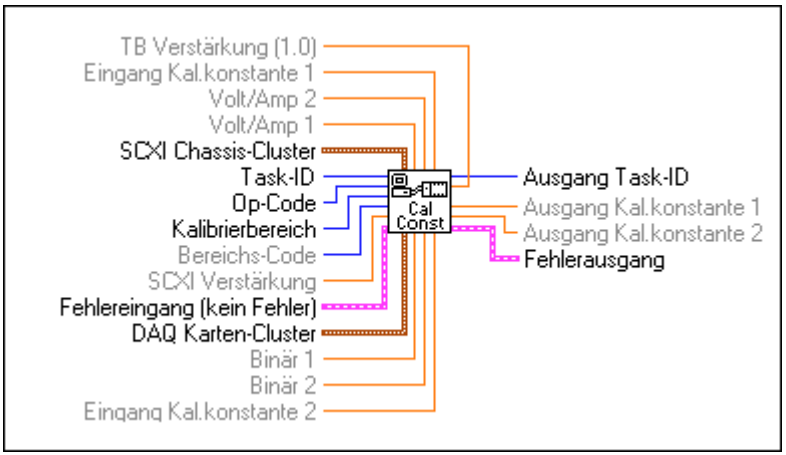
Paßt die Skalierkonstanten an, die LabVIEW beim Konvertieren von Analogeingangs-Binärdaten in Spannungsdaten benutzt, um für Offset und nichtideale Verstärkung Rechnung zu tragen.



Weitere Informationen zum VI Abstimmen der Skalierkonstante finden Sie in der Beschreibung Abstimmung der Skalierkonstante in Kapitel 30, *Signalkonditionierungs-VIs*.

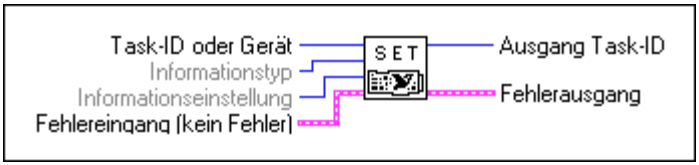
SCXI-Kalibrierkonstante

Errechnet die Kalibrierungskonstanten für den jeweiligen Kanal und Bereich oder Verstärkung unter Benutzung von Spannungs-/Binär-Paaren. Sie können dieses VI mit jedem beliebigen SCXI-Modul benutzen.



DAQ-Geräteinformation einstellen

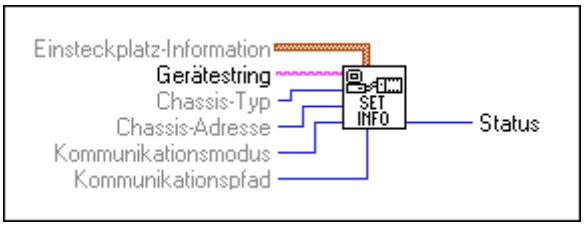
Stellt den Datenübertragungsmodus für verschiedene Operationsarten ein.



In Anhang B, *DAQ-Hardware-Leistungsfähigkeit*, finden Sie die für Ihr DAQ-Gerät verfügbaren Übertragungsmethoden.

SCXI-Information einstellen

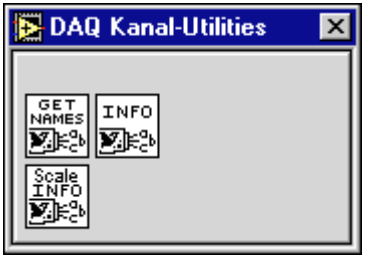
Stellt die SCXI-Chassis-Konfigurationsinformation ein.



Benutzen Sie dieses VI, mit dem Konfigurations-Utility, um die bereits eingestellten Konfiguration außer Kraft zu setzen. Sie können zur Eingabe der Chassis-Konfigurationsinformation dieses VI *anstelle* des Konfigurations-Utility benutzen. Wenn Sie dieses VI nicht benutzen, versucht das erste VI, das auf ein SCXI-Chassis zugreift, automatisch Informationen von der Konfigurationsdatei zu laden.

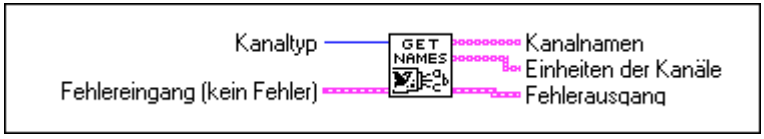
Kanal-Konfigurations-VIs

Folgende Abbildung zeigt die Palette Kanal-Utilities-VIs.



DAQ-Kanalnamen erlangen

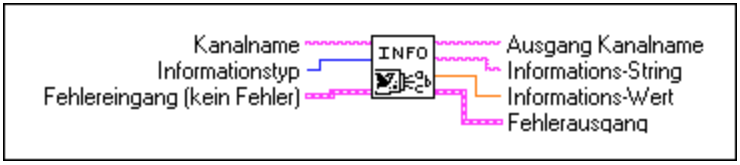
Gibt ein Array aller Kanalnamen in der Standard-Konfigurationsdatei zurück. Ein entsprechendes Array der konfigurierten physischen Einheiten der Kanäle wird auch zurückgegeben. Durch Benutzung des **Kanaltyps** können Sie die Auswahl treffen, alle Kanäle abzufragen oder nur Analog-Eingabe- und Analog-Ausgabe-Kanäle oder digitale I/O-Kanäle.



☞ **Hinweis** ***Dieses VI ist auf Computer ausgelegt, die NI-DAQ 5.0 oder eine spätere Version ausführen. LabVIEW gibt eine Fehlermeldung Nicht unterstützt zurück, wenn Sie versuchen, dieses VI auf Computern auszuführen, die nicht NI-DAQ 5.0 oder eine spätere Version ausführen.***

Kanal-Informationen erlangen

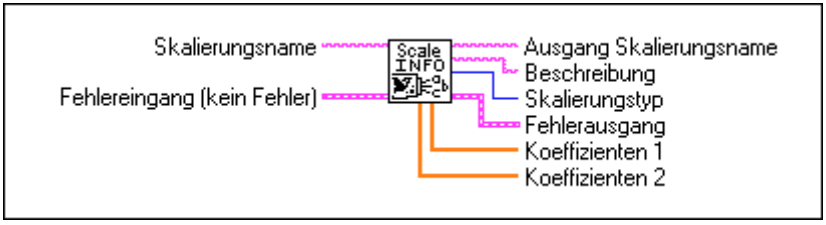
Gibt Konfigurationsinformationen über einen im DAQ Channel Wizard konfigurierten Kanal zurück.



☞ **Hinweis** ***Dieses VI ist auf Computer ausgelegt, die NI-DAQ 5.0 oder eine spätere Version ausführen. LabVIEW gibt eine Fehlermeldung Nicht unterstützt zurück, wenn Sie versuchen, dieses VI auf Computern auszuführen, die nicht NI-DAQ 5.0 oder eine spätere Version ausführen.***

Skalierungsinformation erlangen

Gibt Konfigurationsinformationen über eine im DAQ Channel Wizard konfigurierte Skalierung zurück.



☞ **Hinweis** ***Dieses VI ist auf Computer ausgelegt, die NI-DAQ 5.0 oder eine spätere Version ausführen. LabVIEW gibt eine Fehlermeldung Nicht unterstützt zurück, wenn Sie versuchen, dieses VI auf Computern auszuführen, die nicht NI-DAQ 5.0 oder eine spätere Version ausführen.***

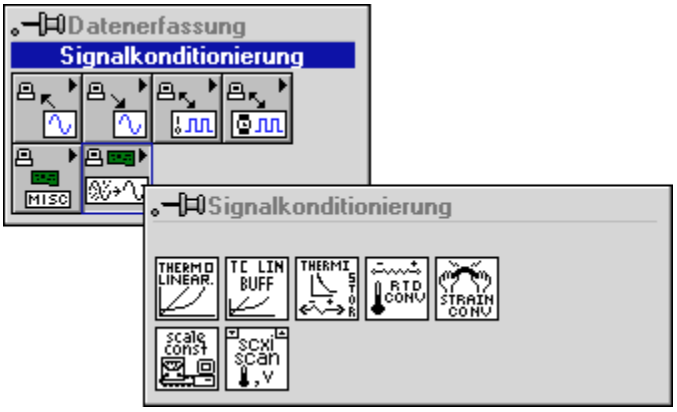
30

Signalkonditionierungs-VIs

Dieses Kapitel beschreibt die Datenerfassungs-VIs zur Signalkonditionierung, die Sie benutzen, um analoge Eingabespannungen zu konvertieren, die von Widerstandstemperaturfühlern (Resistance Temperature Detectors, RTDs), Dehnungsmessern oder Thermoelementen in Dehnungs- oder Temperatureinheiten gelesen werden.

Sie können die in diesen VIs benutzten Verwandlungsgleichungen benutzen oder diese mit Ihren eigenen ersetzen, um den besonderen Genauigkeitsanforderungen Ihrer Anwendung gerecht zu werden. Wenn Sie die Formeln bearbeiten oder ersetzen, müssen Sie das neue VI in einem Ihrer eigenen Verzeichnisse oder Ordner außerhalb von `vi.lib` speichern.

Sie können auf die Signalkonditionierungs-VIs, wie nachstehend gezeigt, durch Auswahl von **Funktionen»Datenerfassung» Signalkonditionierung** zugreifen.

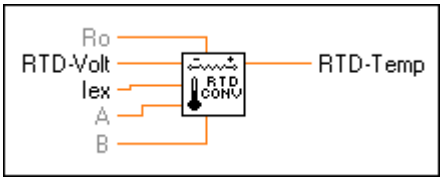


Beschreibungen der Signalkonditionierungs-VIs

Folgende Signalkonditionierungs-VIs sind verfügbar.

RTD Meßwerte konvertieren

Konvertiert von einem RTD abgelesene Spannung in Temperatur in Celsius.



Dieses VI findet zuerst den RTD-Widerstand, indem es **RTD-Volt** durch **Iex** teilt. Das VI konvertiert dann den Widerstand in Temperatur, wozu es folgende Lösung der Callendar Van-Dusen-Gleichung für RTDs benutzt:

$$Rt = Ro[1 + At + Bt^2 + C(t–100)t^3]$$

Für Temperaturen über 0° C ist der C-Koeffizient 0, und die vorausgehende Gleichung reduziert sich auf eine quadratische Gleichung, für die der im VI implementierte Algorithmus die entsprechende Wurzel erteilt. Dieses Konvertierungs-VI ist also nur bei Temperaturen über 0° C genau.

Ihre RTD-Dokumentation sollte Ihnen **Ro** und die Koeffizienten **A** und **B** für die Callendar Van-Dusen-Gleichung geben. Die gebräuchlichsten RTDs sind 100-Ω Platin-RTDs, die entweder der europäischen Temperaturkurve (DIN 43760) oder der amerikanischen Temperaturkurve folgen. In folgender Tabelle sind die Werte für **A** und **B** der europäischen und der amerikanischen Kurven angegeben.

Europäische Kurve (DIN 43760)	Amerikanische Kurve
A = 3,90802e–03 **B** = –5,80195e–07 (α = 0,00385; ∂ = 1,492)	**A** = 3,9784e–03 **B** = –5,8408e–07 (α = 0,00392; ∂ = 1,492)

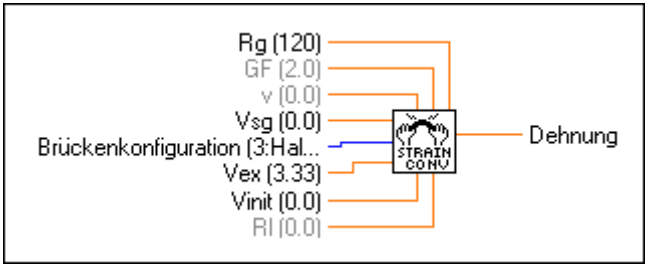
Einige RTD-Dokumentationen geben Werte für α und ∂, aus denen Sie mit Hilfe folgender Gleichungen **A** und **B** errechnen können:

$$\mathbf{A} = \alpha(1 + \partial/100)$$

$$\mathbf{B} = -\alpha\partial/100^2$$

DMS-Meßwerte konvertieren

Konvertiert Spannungen, die Sie von einem Dehnungsstreifen abgelesen haben, in Dehnungseinheiten.



Die vom VI benutzte Umwandlungsgleichung basiert einzig auf der Brückenkonfiguration. In Abbildungs 30-1 bis 30-3 sind die sieben Brückenkonfigurationen abgebildet, die Sie benutzen können, sowie die entsprechenden Formeln. Das VI benutzt für alle Brückenkonfigurationen folgende Formel, um **Vr** zu erhalten:

$$\mathbf{Vr} = (\mathbf{Vsg} - \mathbf{Vinit}) / \mathbf{Vex}$$

In den Schaltplänen ist VOUT die gemessene Spannung, die als **Vsg**-Parameter an das Konvertierungs-VI weitergegeben wird. In den Viertelbrücken- und Halbbrücken-Konfigurationen, sind **R1** und **R2** Dummy-Widerstände, die nicht direkt Teil der Konvertierungsformel sind. Die Module SCXI-1121 und SCXI-1122 stellen, wenn nötig, **R1** und **R2** für das Brückenfertigstellungs-Netzwerk zur Verfügung.

In Ihrem Handbuch *Getting Started with SCXI* finden Sie weitere Informationen zu Brückenergänzungs-Netzwerken und Spannungsspeisung.

In den Abbildungs 30-1 bis 30-3 sind die verfügbaren Halbbrückenschaltungen abgebildet.

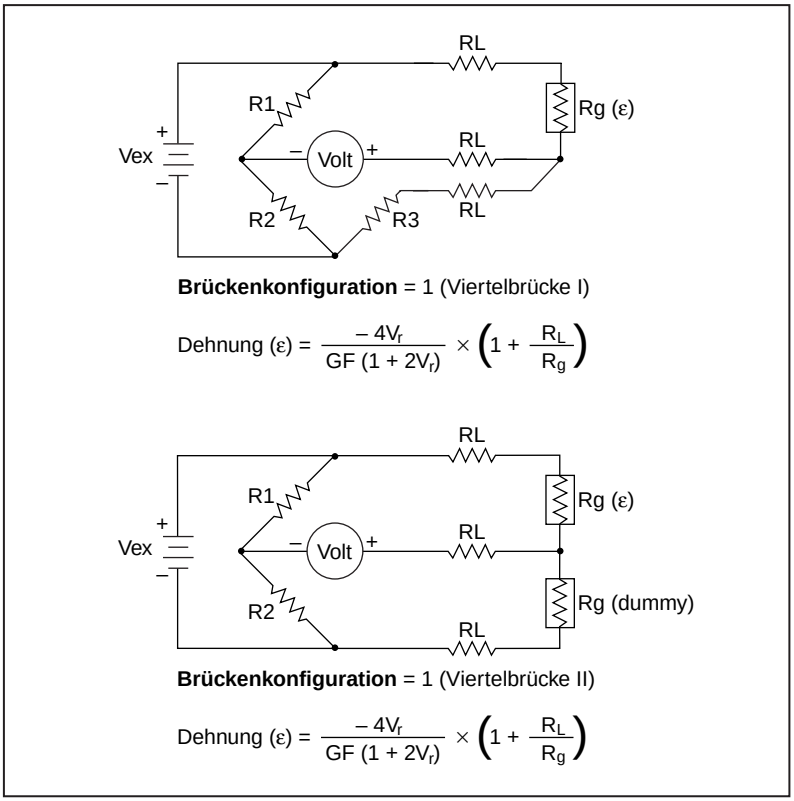


Abbildung 30-1. Meßbrückenschaltung für Dehnungsmessung (Viertelbrücken-Konfiguration)

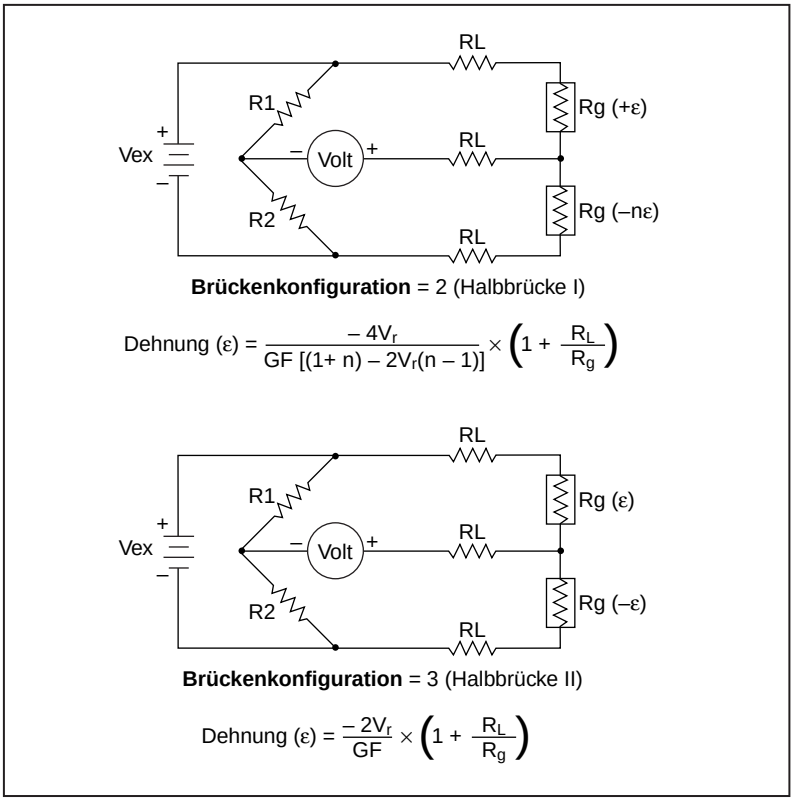
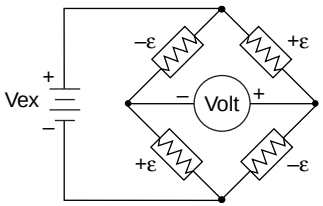
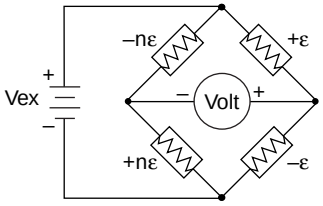


Abbildung 30-2. Meßbrückenschaltung für Dehnungsmessung (Halbbrücken-Konfiguration)



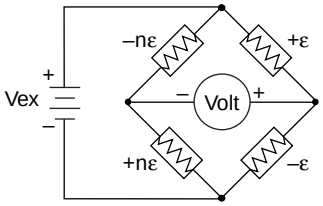
Brückenkonfiguration = 4 (Vollbrücke I)

$$\text{Dehnung } (\varepsilon) = \frac{-V_r}{GF}$$



Brückenkonfiguration = 5 (Vollbrücke II)

$$\text{Dehnung } (\varepsilon) = \frac{-2V_r}{GF(n + 1)}$$



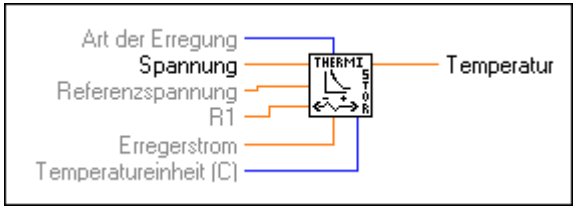
Brückenkonfiguration = 6 (Vollbrücke III)

$$\text{Dehnung } (\varepsilon) = \frac{-2V_r}{GF[(n + 1) - V_r(n - 1)]}$$

Abbildung 30-3. Meßbrückenschaltung für Dehnungsmessung (Vollbrücken-Konfiguration)

Thermistor-Meßwerte konvertieren

Konvertiert Thermistorspannungen in Temperatur. Dieses VI besitzt zwei verschiedene Betriebsmodi für spannungsangeregte und stromangeregte Thermistoren.



Dieses VI besitzt zwei Betriebsmodi zur Benutzung mit verschiedenen Arten von Thermistorkreisen. In Abbildung 30-4 wird gezeigt, wie der Thermistor mit einer Referenzspannung verbunden werden kann. Dies ist die in SCXI-1303, SCXI-1322, SCXI-1327 und SCXI-1328-Anschlußblocks benutzte Einstellung, die einen Onboard-Thermistor für die Kaltstellen-Kompensation (cold-junction compensation) benutzen.

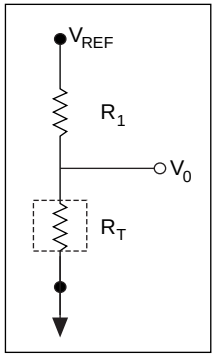


Abbildung 30-4. Schaltplan eines Thermistors in einem Spannungsteiler

In Abbildung 30-5 wird eine Schaltung gezeigt, in der der Thermistor durch eine ständige Stromquelle erregt wird. Ein Beispiel für diese Einstellung ist die Benutzung des Geräts DAQPad-MIO-16XE-50, das für eine ständige Stromausgabe sorgt. Das DAQPad-TB-52 verfügt über einen Thermistor für Cold-Junction Sensing.

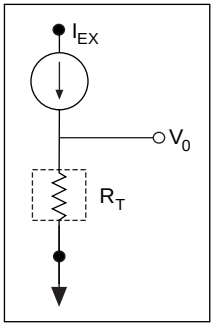


Abbildung 30-5. Schaltplan eines Thermistors mit Stromerregung

Für durch Spannung erregte Thermistoren wird im folgenden eine Gleichung gezeigt, in der der Thermistorwiderstand, R_T, mit den Eingangswerten in Bezug gesetzt wird:

$$R_T = R_1 \left(\frac{V_0}{V_{REF} - V_0} \right)$$

Wenn der Thermistor stromerregt ist, lautet die Gleichung

$$R_T = \frac{V_0}{I_{EX}}$$

Bei folgender Gleichung handelt es sich um die vom VI für die Konvertierung von Thermistorwiderstand in Temperatur benutzte Standardformel:

$$T_K = \frac{1}{a + b(lnR_T) + c(lnR_T)^3}$$

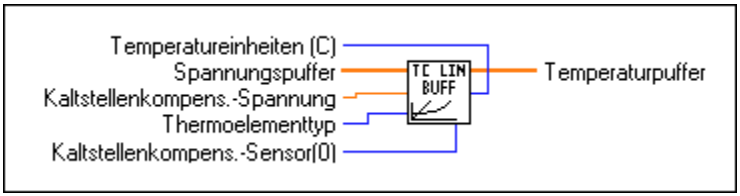
Die von diesem VI für *a*, *b* und *c* benutzten Werte sind nachstehend aufgeführt. Diese Werte sind für die Thermistoren der Anschlußblocks SCXI und DAQPad-TB-52 korrekt. Wenn Sie einen Thermistor mit anderen Werten für *a*, *b* und *c* benutzen (bitte sehen Sie in Ihrem Thermistor-Datenblatt nach), können Sie das VI-Diagramm bearbeiten und Ihre eigenen Werte für *a*, *b* und *c* einsetzen.

a = 1,295361E−3
b = 2,343159E−4
c = 1,018703E−7

Das VI produziert Temperaturen in Grad Celsius. Deshalb ist $T_C = T_K - 273{,}15$.

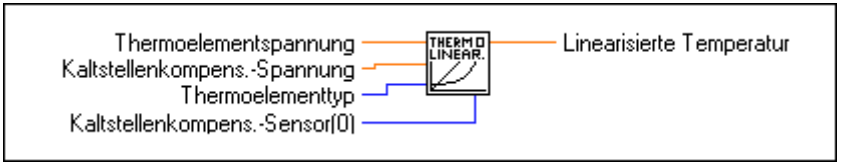
Thermoelement-Puffer konvertieren

Konvertiert einen von einem Thermoelement abgelesenen Spannungspuffer in einen Temperaturpufferwert in Grad Celsius.



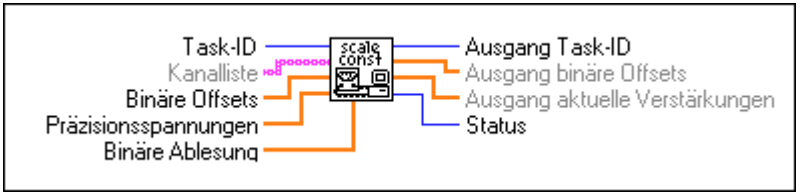
Thermoelement-Meßwerte konvertieren

Konvertiert eine von einem Thermoelement abgelesene Spannung in einen Temperaturwert in Grad Celsius.



Abstimmen der Skalierkonstante

Paßt die Skalierkonstanten an, die LabVIEW zum Ausgleich von Offset und nichtidealer Verstärkung benutzt, um Analog-Eingangsbinärdaten in Spannungsdaten zu konvertieren.



Um dieses VI richtig benutzen zu können, müssen Sie zuerst zwei Analog-Eingangsmessungen durchführen - eine Null-Offsetmessung und eine Messung mit bekannter Spannung.

Das Standard-Binäroffset für jeden Kanal der Gruppe ist 0. Um das tatsächliche Binäroffset für jeden Kanalpfad zu bestimmen, müssen Sie die Kanaleingänge erden und eine Binärmessung durchführen oder mehrfache Binärmessungen durchführen und den Durchschnitt errechnen, um gebrochene LSBs des Offsets zu erhalten.

Wenn Sie SCXI benutzen, müssen Sie die Eingänge der SCXI-Kanäle erden, um das Offset für den gesamten Signalpfad, einschließlich beider SCX-Module und des DAQ-Geräts, zu messen. Die Module SCXI-1100, SCXI-1122 und SCXI-1141 haben einen internen Schalter, den Sie zur Erdung der Verstärkereingänge benutzen können, ohne die Anschlüsse tatsächlich mit der Erde zu verbinden. Wenn Sie dieses Merkmal benutzen möchten, müssen Sie den Spezial-SCXI-String CALGND in Ihrem SCXI-Kanalstring benutzen, wie im Abschnitt *Verstärker-Offset* von Kapitel 21, *Gebräuchliche SCXI-Anwendungen*, im *Grundlagen der Datenerfassung mit LabVIEW* beschrieben. Benutzen Sie Mittlere oder Fortgeschrittene Analogeingangs-VIs, um anstelle von Spannungsdaten Binärdaten zu erhalten.

☞ **Hinweis** ***Wenn Ihr Gerät Dithering unterstützt, sollten Sie diese Möglichkeit an Ihrem DAQ-Gerät aktivieren, wenn Sie Mehrfach-Messungen vornehmen und aus diesen den Mittelwert errechnen.***

LabVIEW geht davon aus, daß die Verstärkungseinstellungen der DAQ-Geräte und die SCXI-Module ideal sind, wenn Binärmessungen zu Spannung skaliert werden, außer, wenn Sie das VI zur Bestimmung der tatsächlichen Verstärkungswerte benutzen. Wenden Sie für jeden Kanal eine bekannte Präzisionsspannung an, oder nehmen Sie an jedem Kanal Mehrfachmessungen vor, und errechnen Sie für jeden Kanal eine Durchschnittsmessung. Ihre Präzisionsspannung sollte ungefähr zehn Mal so genau sein wie die Auflösung Ihres DAQ-Geräts, um verwendungsfähige Ergebnisse zu liefern. Wenn Sie **Binärmessungen**, **Präzisionsspannungen** und **Binäroffsets** mit diesem VI verbinden, bestimmt LabVIEW die tatsächliche Verstärkung mit Hilfe folgender Formel:

$$\text{tats. Verstärk.} = \frac{\text{Spanungsauflösung} * (\text{Binär-Messung} - \text{Binär-Offset})}{\text{Präzisions-Spannung}}$$

In dieser Formel wird der Wert **Spannungsauflösung** und Volt pro LSB ausgedrückt; es ist ein Wert, der abhängig vom DAQ-Gerätetyp, der Polaritätseinstellung und der Einstellung des Eingabebereichs variiert. Die Spannungsauflösung für ein Gerät PCI-MIO-16E-1 im bipolaren Modus mit einem Eingabebereich von+5 bis –5 V ist 2,44 mV. Das VI gibt ein Array der tatsächlichen Verstärkungswerte zurück, die es für jeden Kanal speichert.

☞ **Hinweis** ***Wenn Sie Messungen vornehmen, um das Offset und die tatsächliche Verstärkung zu bestimmen, sollten Sie dieselben Eingabegrenz-Einstellungen und Taktfrequenzen wie für die Messung Ihrer Eingabesignale benutzen.***

LabVIEW benutzt folgende Gleichung, um Binärmessungen zu Spannung zu skalieren:

$$\text{Spannung} = \frac{\text{Spannungsauflösung} * (\text{Binär-Messung} - \text{Binär-Offset})}{\text{Verstärkung}}$$

Wenn Sie das AI Gruppenkonfigurations-VI ausführen, stellt es die Attribute aller Kanäle in der Gruppe auf Ihre Standardwerte ein, einschließlich des Binäroffsets und der Verstärkungswerte.

Wenn Sie die Skalierkonstanten für ein Subset der Kanäle in der Gruppe anpassen möchten, können Sie die **Kanalliste** verbinden. Wenn die **Kanalliste** unverbunden bleibt, paßt das VI die Skalierkonstanten für alle Kanäle in der Gruppe an. Das VI benutzt dieselbe Methode wie das AI Hardware-Konfigurations-VI, um Werte in den Eingabearrays **Binäroffsets**, **Präzisionsspannungen** und **Binärmessungen** anzuwenden. Das heißt, wenn Sie die **Kanalliste** mit diesem VI verbunden haben, wird das erste Element (bei Index 0) der Eingabearrays **(Binäroffsets, Präzisionsspannung** und **Binärablesungen)** auf die bei Index 0 der **Kanalliste** angegebenen Kanäle angewandt. Wenn Sie die **Kanalliste** unverbunden lassen, sind die ersten Werte der Eingabearrays für den ersten Kanal in der Gruppe anwendbar. Das VI wendet die Werte jedes Eingabearrays auf die Kanäle der **Kanalliste** oder so gesehen in der Gruppe an, bis das VI die Arrays erschöpft. Wenn Kanäle in der **Kanalliste** oder in der Gruppe unkonfiguriert bleiben, wendet das VI die Endwerte in den Arrays auf alle verbleibenden unkonfigurierten Kanäle an.

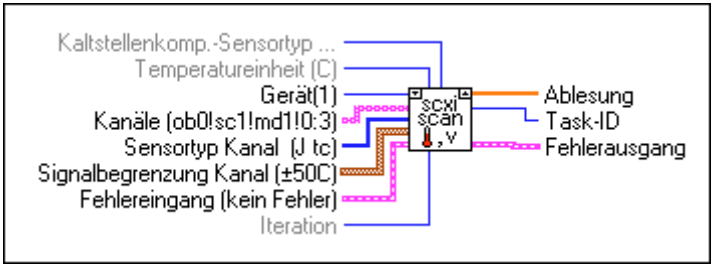
Wenn Sie nur die Kanaloffsets anpassen möchten und Sie davon ausgehen, daß die Verstärkungseinstellungen am DAQ-Gerät und den SCXI-Modulen ideal sind, verbinden Sie nur die **Binäroffsets** und lassen Sie die **Präzisionsspannungen** und **Binärmessungen** unverbunden.

Sie können dieses VI auch zum Abfragen der Binär-Offsetwerte und der tatsächlichen Verstärkungswerte für alle Kanäle in der Gruppe durch ausschließliches Verbinden mit **Task-ID** verwenden.

Sie können dieses VI auch für die Anpassung der Skalierkonstanten für einen Kanalpfad verwenden; alle Analogeingabe-VIs, die Spannungsdaten zurückgeben, benutzen die angepaßten Konstanten zur Skalierung. Sie können das VI AI Gruppenkonfiguration zum Zurücksetzen der Skalierkonstanten für jeden Kanal in der Gruppe auf Ihre Standardwerte (Null-Offset und ideale Verstärkung) verwenden.

SCXI-Temperaturabtastung

Dieses VI gibt eine einzelne Abtastung der Temperaturdaten von einer Liste von SCXI-Kanälen zurück. Das VI SCXI-Temperaturabtastung benutzt die Durchschnittsbildung zur Reduzierung von Rauschen von 60 Hz und 50 Hz, führt die Thermoelement-Linearisierung und die Offset-Kompensation für SCXI-1100-Module durch.



Teil III

Instrumenten-I/O-Funktionen und -VIs

Teil III, *Instrumenten-I/O-Funktionen und -VIs*, beschreibt Instrumententreiber, GPIB, serielle Anschlüsse, Instrumententreibervorlagen sowie VISA-VIs und -Funktionen von LabVIEW. In diesem Teil sind folgende Kapitel enthalten:

- Kapitel 31, *Einführung in LabVIEW Geräte-I/O-VIs*, stellt Instrumententreiber und GPIB, serielle Anschlüsse, Instrumententreibervorlagen und VISA-VIs und -Funktionen von LabVIEW vor.
- Kapitel 32, *Vorlage-VIs für Instrumententreiber*, beschreibt die Instrumententreibervorlage-VIs.
- Kapitel 33, *VISA-Referenzbibliothek*, beschreibt die VISA-Bibliotheksreferenz-Operationen und -attribute.
- Kapitel 34, *Traditionelle GPIB-Funktionen*, beschreibt die traditionellen GPIB-Funktionen.
- Kapitel 35, *GPIB-488.2-Funktionen*, beschreibt die IEEE 488.2-Funktionen (GPIB).
- Kapitel 36, *Serielle Anschluß-VIs*, beschreibt die VIs für serielle Anschlußoperationen.

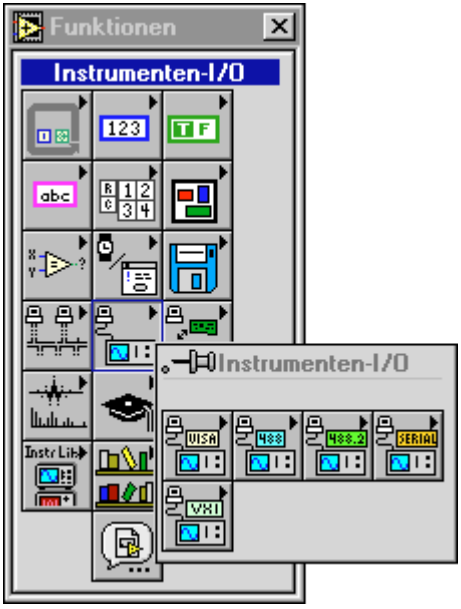
31

Einführung in LabVIEW Geräte-I/O-VIs

Dieses Kapitel beschreibt die LabVIEW Gerätetreiber und GPIB, serielle Anschlüsse, Gerätetreibervorlagen und VISA-VIs und Funktionen.

Die VIs Gerätetreiber können Sie in der Palette **Funktionen** Ihres Blockdiagramms in LabVIEW finden. Die VIs Gerätetreiber befinden sich im unteren Teil der Palette **Funktionen**.

Sie können auf die Palette **Instrumenten-I/O,** wie in nachstehender Abbildung gezeigt, durch Auswahl von **Funktionen»Instrumenten-I/O** zugreifen.



Die Palette **Instrumenten-I/O** besteht aus folgenden Unterpaletten:

- VISA
- Traditionelles GPIB
- GPIB 488.2
- Seriell

Sie können wertvolle Informationen über einzelne VIs online durch Benutzung des Hilfe-Fensters von LabVIEW (**Hilfe»Hilfe anzeigen**) erhalten. Wenn Sie den Cursor auf ein VI-Icon plazieren, erscheinen die Verbindungsdiagramme und Parameterbezeichnungen für dieses VI im Hilfe-Fenster. Sie können auch Informationen über Frontpanel-Bedienelemente oder Anzeigeelemente finden, indem Sie den Cursor bei offenem Hilfe-Fenster über das Bedien- oder Anzeigeelement plazieren. Weitere Informationen zum Fenster Hilfe von LabVIEW finden Sie im Abschnitt *So erhalten Sie Hilfe* in Kapitel 1, *Einführung in die Programmierung in G*, im *Referenzhandbuch zur Programmierung in G*.

Zusätzlich zum Fenster Hilfe sind bei LabVIEW weitere Online-Informationen verfügbar. Wählen Sie zum Zugriff auf diese Informationen **Hilfe»Online-Referenz** aus. Für die meisten Blockdiagramm-Objekte können Sie **Online-Referenz** aus dem Popup-Menü des Objekts auswählen, um auf die Online-Beschreibung zuzugreifen. Weitere Informationen über die Erstellung Ihrer eigenen Online-Referenzdateien finden Sie im Abschnitt *Erstellen Ihrer eigenen Hilfe-Dateien* in Kapitel 5, *Drucken und Dokumentieren von VIs*, im *Referenzhandbuch zur Programmierung in G*.

Überblick über Gerätetreiber

Bei einem LabVIEW -Gerätetreiber handelt es sich um einen Satz VIs, die ein programmierbares Gerät steuern. Jedes VI entspricht einer Programmoperation, wie der Konfiguration, Messung von, Schreiben auf oder dem Triggern des Geräts. LabVIEW Gerätetreiber vereinfachen die Gerätesteuerung und reduzieren die Testprogrammentwicklung durch die nicht mehr vorhandene Notwendigkeit, das Low-Level-Programmierprotokoll für jedes Gerät zu erlernen.

Die LabVIEW Gerätetreiber-Bibliothek von National Instruments enthält Gerätetreiber für eine Vielzahl programmierbarer Instrumente, einschließlich GPIB, VXI und seriell. Wenn sich ein Treiber für Ihr Gerät in der Bibliothek befindet, können Sie diesen wie vorhanden zur Steuerung Ihres Geräts benutzen. Gerätetreiber werden mit einem Blockdiagramm-

Quellcode verteilt, so daß Sie ihn, wenn nötig, für Ihre spezielle Anwendung anpassen können. Wenn für Ihr Gerät kein Treiber existiert, können Sie einen der folgenden Vorschläge befolgen:

- Benutzen Sie einen Treiber für ein ähnliches Gerät. Oft haben ähnliche Geräte vom selben Hersteller ähnliche, wenn nicht sogar, identische Gerätetreiber.
- Ändern Sie zur Erstellung eines neuen Treibers für Ihr Gerät die VIs Gerätetreibervorlagen.
- Benutzen Sie entweder die im Lieferumfang enthaltenen GPIB, VXI, seriellen oder VISA- I/O-Bibliotheken, um Befehle direkt an Ihr Gerät zu senden oder von diesem zu empfangen.
- In Kapitel 7, *Erste Schritte mit einem LabVIEW-Gerätetreiber*, im *LabVIEW Benutzerhandbuch* finden Sie Informationen, wie Sie mit der Benutzung der LabVIEW Gerätetreiber von National Instruments beginnen.

Gerätetreiber-Verteilung

LabVIEW Gerätetreiber werden in einer Vielzahl von Medien verteilt, einschließlich elektronisch über ein Mailbox-System (BBS) und das Internet und auf CD-ROM.

Sie können die neuesten Versionen der LabVIEW Gerätetreiber von einem der BBS-Systeme von National Instruments herunterladen, und, wenn Sie Internet-Zugriff haben, können Sie die neuesten Gerätetreiberdateien von der FTP-Site von National Instruments herunterladen. Hierzu wird auf die Abschnitte *Bulletin Board Support* und *FTP-Unterstützung* in Anhang D, *Kundenbetreuung* hingewiesen.

Gerätetreiber-Verteilung auf CD-ROM

Die gesamte Bibliothek der LabVIEW Gerätetreiber ist auf CD-ROM erhältlich. Die CD-ROM ist von National Instruments kostenlos erhältlich.

Sie können die neueste Gerätetreiberliste mittels eines Tonwahl-Telefons erhalten, indem Sie das automatisierte Faxsystem von National Instruments, Faxabrufdienst, unter der Nr. (512) 418 1111, USA, anrufen oder indem Sie National Instruments anrufen.

Gerätetreiber-Vorlage-VIs

Die LabVIEW Gerätetreiber-Vorlagen stellen die Grundlage der gesamten LabVIEW Gerätetreiber-Entwicklung dar. Die Vorlagen haben eine einfache, flexible Struktur und einen gebräuchlichen Satz an Gerätetreiber-VIs, die Sie zur Treiberentwicklung benutzen können. Die VIs erstellen ein Standardformat für alle LabVIEW Treiber, wobei jedes mit Anweisungen zur Änderung für ein bestimmtes Gerät ausgestattet ist.

Bei den LabVIEW Gerätetreiber-Vorlagen handelt es sich um Gerätetreiber-VIs, die gebräuchliche Operationen, wie Initialisierung, Eigentest, Fehlerabfrage usw. durchführen. Anstatt zur Durchführung dieser Aufgabe Ihre eigenen VIs zu entwickeln, sollten Sie die VIS Gerätetreiber-Vorlagen von LabVIEW benutzen, die bereits den LabVIEW Standards für Gerätetreiber entsprechen.

In Kapitel 32, *Vorlage-VIs für Instrumententreiber*, finden Sie weitere Informationen zu Gerätetreiber-Vorlagen-VIs.

Einführung zur VISA-Bibliothek

Bei VISA (Virtual Instrument Software Architecture; virtuelle Geräte-Softwarearchitektur) handelt es sich um eine einzelne Schnittstellen-Bibliotheken zur Steuerung der Geräte VXI-, GPIB-, RS-232- und anderer Gerätetypen. Die VISA-Bibliothek bietet einen Standardsatz an I/O-Routinen, die von allen LabVIEW Gerätetreibern benutzt werden. Durch die Benutzung der VISA-Funktionen können Sie ein einzelnes Gerätetreiber-VIs erstellen, das ein bestimmtes Gerätemodell über verschiedene I/O-Schnittstellen steuert.

Ein Geräte-Descriptorstring wird an die VISA Funktion open weitergeleitet, um auszuwählen, welche Art I/O zur Kommunikation mit dem Gerät benutzt wird. Sobald die Sitzung mit dem Gerät geöffnet ist, führen Funktionen wie VISA Read und VISA Write die Geräte-I/O-Aktivitäten auf generische Art und Weise so durch, so daß das Programm nicht an bestimmte GPIB- oder VXI-Funktionen gebunden ist. Ein solcher Gerätetreiber wird als schnittstellenunabhängig betrachtet und kann in verschiedenen Systemen benutzt werden.

Gerätetreiber, die die VISA-Funktionen benutzen, führen dem Gerät eigene und nicht der Kommunikationsschnittstelle eigene Aktivitäten aus. Dadurch werden mehr Gelegenheiten für die Benutzung der Gerätetreiber in vielen unterschiedlichen Situationen geschaffen.

Weitere Informationen zu VISA-Funktionen finden Sie in Kapitel 33, *VISA-Referenzbibliothek*.

Einführung in GPIB

Der General Purpose Interface Bus (GPIB; Allzweck-Schnittstellenbus) ist eine Verknüpfung oder ein Schnittstellensystem, durch welches miteinander verbundene elektronische Geräte kommunizieren.

Traditionelle GPIB-Funktionen von LabVIEW

Diese traditionellen GPIB-Funktionen sind sowohl mit IEEE 488- als auch mit IEEE 488.2-Geräten kompatibel und sind für die meisten Anwendungen ausreichend. Für komplexere Anwendungen, wie die Benutzung mehrerer Geräte und mehr als einer GPIB-Schnittstelle, können Sie die GPIB IEE 488.2-Funktionen benutzen.

Weitere Informationen zu den LabVIEW Traditionelle GPIB-Funktionen finden Sie in Kapitel 34, *Traditionelle GPIB-Funktionen*.

GPIB 488.2-Funktionen

Die Benutzung von GPIB 488.2-Funktionen zusammen mit IEEE 488.2-kompatiblen Geräten verbessert die Vorhersagbarkeit des Geräts und des Softwareverhaltens und vermindert die Programmierunterschiede zwischen Geräten verschiedener Hersteller.

Die neuesten Revisionen vieler GPIB-Karten von National Instruments sind mit den IEEE 488.2-Spezifikationen für Controller voll kompatibel. Das LabVIEW-Paket enthält auch Funktionen, die IEEE 488.2 verwenden. Durch die Benutzung dieser Funktionen hält sich Ihre Programmierschnittstelle strikt an den IEEE 488.2-Standard für Befehls- und Datenfolgen.

Die GPIB 488.2-Funktionen enthalten dieselbe grundlegende Funktionalität wie die traditionellen GPIB-Funktionen und beinhalten folgende Verbesserungen und Zusätze:

- Sie spezifizieren die GPIB-Geräteadresse anstelle eines Strings mit einem Integer. Des weiteren geben Sie die Busnummer mit einer zusätzlichen numerischen Steuerung an, die den Umgang mit mehrfachen GPIB-Schnittstellen vereinfacht.
- Sie können den GPIB-Status, Fehler und/oder Bytezählung direkt vom Anschlußfeld einer jeden GPIB 488.2-Funktion bestimmen. Sie

müssen zur Erlangung von Fehler- und anderer Informationen nicht mehr die GPIB-Statusfunktion benutzen.

- Die Funktion FindLstn implementiert das Protokoll IEEE 488.2 Alle Listener finden. Sie können diese Funktion am Anfang einer Anwendung benutzen, um zu bestimmen, welche Geräte am Bus vorhanden sind, ohne ihre Adresse zu kennen.
- Die Funktion GPIB Verschiedene ist noch verfügbar, aber in den meisten Fällen nicht mehr notwendig. IEEE 488.2 spezifiziert Routinen für die meisten GPIB-Anwendungsbedürfnisse, die als Funktionen implementiert sind. Sie können jedoch die Funktion GPIB Verschiedene sowie andere GPIB-Funktionen, wenn erforderlich, mit den GPIB 488.2-Funktionen mischen.
- Es sind GPIB 488.2-Funktionen mit Low-Level- und High-Level-Funktionalität vorhanden, um jeder GPIB-Anwendung zu entsprechen. Verwenden Sie die Low-Level-Funktionen in Non-Controller-Situationen oder wenn Sie zusätzliche Flexibilität benötigen.
- Obwohl Sie mit diesen Funktionen einen IEEE 488.2-kompatiblen Controller benutzen müssen, können Sie sowohl die IEEE 488.1-Geräte als auch die IEEE 488.2-Geräte steuern. Die GPIB 488.2-Funktionen sind in fünf Funktionskategorien aufgeteilt: Einzelgerät, Mehrfachgerät, Busmanagement, Low-Level und allgemein.

Einzelgerät-Funktionen

Die Einzelgerät-Funktionen führen GPIB-I/O- und Steueroperationen mit einem einzelnen GPIB-Gerät durch. Im allgemeinen akzeptiert jede Funktion eine Einzelgerätadresse als einen seiner Eingänge.

Weitere Informationen zu Einzelgeräte-Funktionen finden Sie in Kapitel 35, *GPIB-488.2-Funktionen*.

Mehrfachgeräte-Funktionen

Die Mehrfachgeräte-Funktionen führen GPIB-I/O- und andere Steueroperationen mit mehreren GPIB-Geräten gleichzeitig aus. Im allgemeinen akzeptiert jede Funktion ein Array von Adressen als einen seiner Eingänge.

Weitere Informationen zu Mehrfachgeräte-Funktionen finden Sie in Kapitel 35, *GPIB-488.2-Funktionen*.

Bus-Verwaltungsfunktionen

Die Busmanagement-Funktionen führen Funktionen im gesamten System durch oder berichten über den Status des gesamten Systems.

Weitere Informationen zu Busmanagement-Funktionen finden Sie in Kapitel 35, *GPIB-488.2-Funktionen*.

Low-Level-Funktionen

Mit den Low-Level-Funktionen können Sie ein spezifischeres, detailliertes Programm erstellen als mit High-Level-Funktionen. Benutzen Sie Low-Level-Funktionen für ungewöhnliche Situationen oder für Situationen, die eine zusätzliche Flexibilität erfordern.

Weitere Informationen zu Low-Level-Funktionen finden Sie in Kapitel 35, *GPIB-488.2-Funktionen*.

Allgemeine Funktionen

Die allgemeinen Funktionen sind in besonderen Situationen nützlich.

Weitere Informationen zu allgemeinen Funktionen finden Sie in Kapitel 35, *GPIB-488.2-Funktionen*.

Überblick über serielle Anschluß-VIs

Die seriellen Anschluß-VIs konfigurieren den seriellen Anschluß Ihres Computers und führen mit Hilfe dieses Anschlusses I/O durch.

Weitere Informationen zu seriellen Anschluß-Funktionen finden Sie in Kapitel 36, *Serielle Anschluß-VIs*.

32

Vorlage-VIs für Instrumententreiber

Dieses Kapitel beschreibt die Instrumententreiber-Vorlage-VIs. Diese VIs befinden sich in `examples\instr\insttmpl.llb`.

Einführung in die Instrumententreiber-Vorlage-VIs

Die LabVIEW Instrumententreiber-Vorlagen bilden die Grundlage für alle LabVIEW Instrumententreiber-Entwicklungen. Die Vorlagen besitzen eine einfache, flexible Struktur und einen allgemeinen Satz von Instrumententreiber-VIs zur Treiberentwicklung. Die Vorlagen richten ein Standardformat für alle LabVIEW Treiber ein, und jede verfügt über Anweisungen zum Abändern für ein bestimmtes Instrument. Die LabVIEW Instrumententreiber-Vorlagen enthalten die folgenden 11 vordefinierten Vorlagekomponenten-VIs:

- PREFIX initialisieren
- PREFIX initialisieren (VXI, Reg-gestützt)
- PREFIX schließen
- PREFIX rücksetzen
- PREFIX-Selbsttest
- PREFIX-Fehlerabfrage
- PREFIX-Fehlerabfrage (Mehrfache)
- PREFIX-Fehlermeldung
- PREFIX-Versionsabfrage
- PREFIX meldungsgestützte Vorlage
- PREFIX registergestützte Vorlage

Die Vorlagen enthalten die folgenden Unterstützungs-VIs.

- PREFIX Utility-Säuberung initialisieren
- PREFIX Utility-Instrumenten-Standardsetup

Sie enthalten außerdem einen PREFIX VI-Baum, einen VI-Beispiel-Baum.

Anstatt Ihre eigenen VIs zur Bewältigung dieser Aufgaben zu entwickeln, können Sie die LabVIEW Instrumententreiber-Vorlage-VIs einsetzen, die bereits mit den LabVIEW Standards für Instrumententreiber übereinstimmen. Die Vorlage-VIs sind IEEE 488.2-kompatibel und arbeiten nach nur minimalen Änderungen mit IEEE 488.2-Instrumenten. Für nicht-IEEE 488.2-Instrumente sind die Vorlagen-VIs als eine Shell oder als Muster zu verwenden, in denen Sie, wo nötig, die entsprechenden instrumentenspezifischen Befehle austauschen. Nach dem Abändern besitzen Sie einen Treiber auf Basisebene, der alle Vorlagen-Instrumententreiber-VIs für Ihr spezielles Instrument umsetzt.

Außerdem ähneln die von den Vorlagen-VIs entwickelten LabVIEW Instrumententreiber anderen Instrumententreibern in der Bibliothek. Dadurch verfügen Sie bei Ihrer Arbeit mit mehrfachen Instrumententreibern über eine größere Vertrautheit und besitzen ein besseres Verständnis dafür.

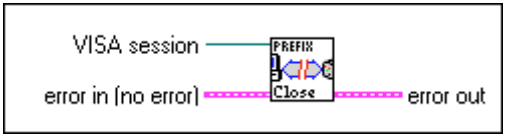
Beschreibungen der Instrumententreiber-Vorlage-VIs

Die folgenden Instrumententreiber-Vorlage-VIs sind verfügbar.

☞ **Hinweis** ***Folgen Sie den Anweisungen auf dem Frontpanel des Vorlagen-VIs, um Ihr eigenes Instrumententreiber-VI zu entwickeln.***

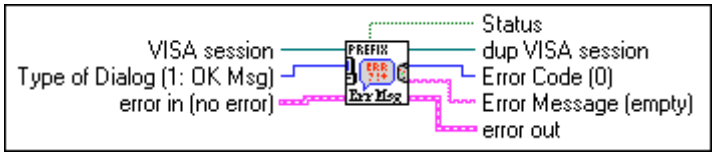
PREFIX schließen

Alle LabVIEW Instrumententreiber sollten ein VI Schließen beinhalten. Das Schließen-VI ist das bei der Steuerung eines Instruments zuletzt aufgerufene VI. Es beendet die Software-Verbindung mit dem Instrument und entfernt die Zuweisungen von Systemressourcen. Daneben können Sie auch entscheiden, das Instrument in Leerlauf zu versetzen. Wenn Sie z.B. einen Schaltertreiber entwickeln, können Sie beim Schließen des Instrumententreibers die Verbindungen von allen Schaltern auflösen.



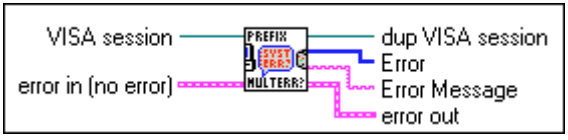
PREFIX-Fehlermeldung

Das VI PREFIX-Fehlermeldung bildet eine Vorlage zum Erstellen eines Fehlermeldungs-VIs für Ihr bestimmtes Instrument. Es übersetzt die Fehlerstatusinformationen, die ein LabVIEW Instrumententreiber-VI ausgibt, in einen für den Benutzer lesbaren String.



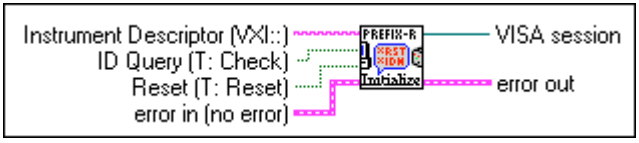
PREFIX-Fehlerabfrage, Fehlerabfrage (Mehrfache) und Fehlermeldung

Für den Fall, daß ein Instrument über eine Fehlerabfragefunktion verfügt, besitzt der LabVIEW Instrumententreiber Fehlerabfrage- und Fehlermeldung-VIs. Das Fehlerabfrage-VI fragt das Instrument ab und gibt die instrumentenspezifischen Fehlerinformationen aus. Das Fehlermeldungs-VI übersetzt die von einem LabVIEW Instrumententreiber-VI ausgegebenen Fehlerstatusinformationen in einen für den Benutzer lesbaren String.



PREFIX initialisieren und PREFIX initialisieren (VXI, Register-basierend)

Das VI Initialisieren ist das beim Zugriff auf einen Instrumententreiber zuerst aufgerufene VI. Es konfiguriert die Kommunikationsoberfläche, verwaltet Handles und sendet einen Standardbefehl zum Instrument. Normalerweise konfiguriert das Standardsetup die Instrumentenoperation für den Rest des Treibers (einschließlich der Umschaltkopfzeilen Ein oder Aus unter Verwendung von langen oder kurzen Formen für Abfragen). Nach erfolgreicher Operation gibt das VI Initialisieren eine **VISA-Session** aus, die das Instrument in allen nachfolgenden Instrumententreiber-VIs adressiert. Das VI Initialisieren ist eine Vorlage für meldungsgestützte Instrumente, wohingegen das VI Initialisieren (VXI, reg.-gestützt) für registergestützte Instrumente bestimmt ist.



Das VI besitzt als einen Eingang einen **Instrumenten-Deskriptor**-String. Von der Syntax dieser Eingabe ausgehend konfiguriert dieses VI die I/O-Schnittstelle und erzeugt ein Instrumenten-Handle für alle anderen Instrumententreiber-VIs. Die folgende Tabelle zeigt die Grammatik für den **Instrumenten-Deskriptor**. Optionale Parameter sind in eckigen Klammern angegeben ([]).

Schnittstelle	Syntax
GPIB	`GPIB[Platine]::Primäradresse[::Sekundäradresse][::INSTR]`
VXI	`VXI::VXI logische Adresse[::INSTR]`
GPIB-VXI	`GPIB-VXI[Platine][::GPIB-VXI-Primäradresse]::VXI logische Adresse[::INSTR]`
Seriell	`ASRL[Platine][::INSTR]`

Das GPIB-Schlüsselwort wird mit GPIB-Instrumenten verwendet. Das VXI-Schlüsselwort wird sowohl für eingebettete als auch MXI-Bus-Controller verwendet. Das GPIB-VXI-Schlüsselwort wird für einen National Instruments GPIB-VXI-Controller verwendet.

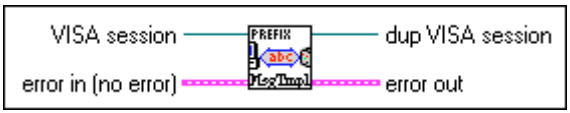
Die folgende Tabelle zeigt die Standardwerte für optionale Parameter.

Optionaler Parameter	Standardwerte
Platine	0
Sekundäradresse	keine
GPIB-VXI-Primäradresse	1

Darüber hinaus kann das VI Initialisieren ausgewählte ID-Abfrage- und Rücksetzoperationen ausführen. Mit anderen Worten, Sie können die ID-Abfrage deaktivieren, wenn Sie versuchen, den Treiber mit einem ähnlichen, doch unterschiedlichen Instrument zu verwenden, ohne den Treiberquellcode zu ändern. Außerdem können Sie die Rücksetzoperation aktivieren und deaktivieren. Diese Funktion ist beim Debugging nützlich, wo das Rücksetzen das Instrument aus dem Zustand nehmen würde, den Sie testen möchten.

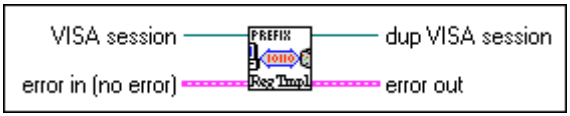
PREFIX meldungsgestützte Vorlage und registergestützte Vorlage

Die VIs PREFIX meldungsgestützte Vorlage und PREFIX registergestützte Vorlage bilden den Ausgangspunkt zur Entwicklung Ihrer eigenen Instrumententreiber-VIs. Die Vorlagen-VIs verfügen über alle erforderlichen Instrumententreiber-Bedienelemente sowie Anweisungen zur Änderung für ein bestimmtes Instrument.



PREFIX registergestützte Vorlage

Das VI PREFIX registergestützte Vorlage ist eine Vorlage zur Erstellung eines registergestützten VIs für Ihr spezielles Instrument.



PREFIX Rücksetzen

Alle LabVIEW Instrumententreiber besitzen ein Rücksetzen-VI, welches das Instrument in einen Standardzustand setzt. Der Standardzustand, in den das VI Rücksetzen das Instrument versetzt, sollte in den Hilfe-Informationen dokumentiert werden. Bei einem IEEE488.2-Instrument sendet dieses VI den Befehlsstring `*RST` an das Instrument. Wenn das Instrument vom VI Initialisieren aus zurückgesetzt wird, wird dieses VI aufgerufen. Außerdem kann das Rücksetzen-VI auch getrennt aufgerufen werden. Wenn das Instrument das Rücksetzen nicht ausführen kann, sollte das Rücksetzen-VI den verbalen String **`Rücksetzen nicht unterstützt (Reset Not Supported)`** ausgeben.

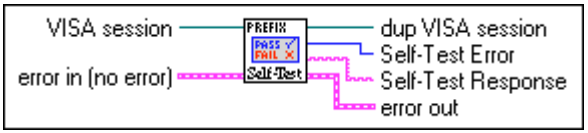
PREFIX Versionsabfrage

LabVIEW Instrumententreiber besitzen ein Versionsabfrage-VI. Dieses VI gibt folgendes aus:

- Die Version des Instrumententreibers.
- Die Firmware-Version des verwendeten Instruments. (Sollte die Version der Firmware des Instruments nicht abgefragt werden können, sollte das VI Versionsabfrage den verbalen String **`Firmware-Version nicht unterstützt (Firmware Revision Not Supported)`** ausgeben.

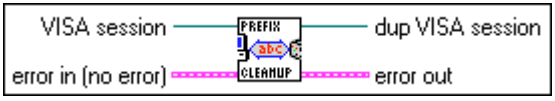
PREFIX Selbsttest

Wenn ein Instrument über die Selbsttest-Funktion verfügt, sollte der LabVIEW Instrumententreiber ein Selbsttest-VI enthalten, welches das Instrument anweist, einen Selbsttest auszuführen und dessen Ergebnisse auszugeben. Sollte das Instrument keinen Selbsttest ausführen können, gibt das Selbsttest-VI den verbalen String **`Selbsttest nicht unterstützt (Self-Test Not Supported)`** aus.



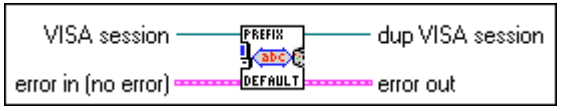
PREFIX Utility Säuberung initialisieren

Schließt eine offene VISA-Session, wenn während der Initialisierung ein Fehler auftritt. Dieses VI sollte nur vom VI Initialisieren aus aufgerufen werden.



PREFIX Dienstprogramm Instrumenten-Standardsetup

Sendet jedes Mal, wenn eine neue **VISA-Session** geöffnet oder das Instrument zurückgesetzt wird, einen Standardbefehlsstring an das Instrument. Verwenden Sie dieses VI als ein SubVI der VIs Initialisieren und Rücksetzen.



PREFIX VI-Baum

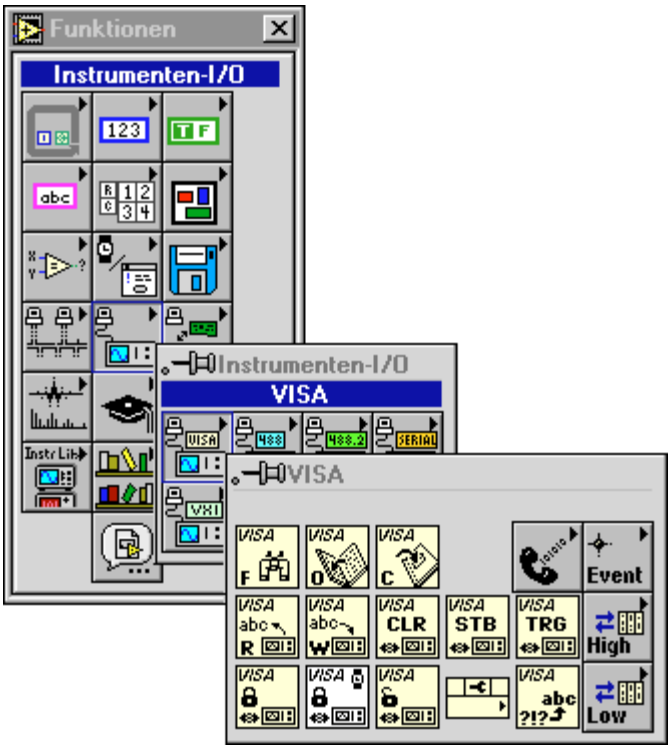
Bei dem VI-Baum-VI handelt es sich um kein ausführbares VI. Es zeigt die funktionale Struktur des Instrumententreibers. Es enthält das Erste-Schritte-VI, Anwendungs-VIs und alle Komponenten-VIs.

33

VISA-Referenzbibliothek

Dieses Kapitel beschreibt die VISA-Referenzbibliothek-Operationen und -Attribute.

Die folgende Abbildung zeigt die **VISA**-Palette, die über das Menü **Funktionen**»**Instrumenten-I/O**»**VISA** aufgerufen wird.



Die VISA-Palette umfaßt die folgenden Unterpaletten:

- Ereignisbehandlung-Funktionen
- High-Level Event Access
- Low-Level Registry Access
- Serielle Funktionen

Operationen

Dieser Abschnitt beschreibt die VISA-Referenzbibliothek-Operationen.

VISA Referenzbibliothek-Parameter

Die Mehrzahl der VISA-Bibliothek-Operationen verwendet die folgenden Parameter:

- **VISA-Session** ist ein einmaliger logischer Identifikator, der zur Kommunikation mit einer Ressource verwendet wird. Er wird von der Funktion VISA Open erstellt und mit einer Ressource verbunden. Danach wird er von anderen VISA-Funktionen für den Zugriff auf eine Ressource und deren Attribute verwendet. VISA-Session-Duplikat ist eine Kopie von VISA-Session, die von den VISA-Funktionen verteilt wird. Indem VISA-Session in Funktionen ein- und ausgegeben wird, kann die Datenflußprogrammierung durch Aneinanderreihen von Funktionen vereinfacht werden. Das ähnelt den Datei-Refnum-Duplikaten, die von den Datei-I/O-Funktionen verwendet werden.

  VISA-Session ist standardmäßig auf die Klasse **Instr** gesetzt. Der Klassentyp kann im Bearbeiten-Modus durch Aufrufen vom Popup-Menü des Bedienelements der VISA-Session und Auswählen einer anderen Klasse geändert werden. Zur Zeit werden die folgenden Klassen unterstützt:

  – Instrument
  – GPIB-Instrument
  – VXI/GPIB-VXI/VME RBD-Instrument
  – VXI/GPIB-VXI MBD-Instrument
  – Serielles Instrument

- Allgemeines Ereignis
- Service-Request-Ereignis (Wartungsanforderung)
- Trigger-Ereignis
- VXI-Signal-Ereignis
- VXI/VME-Interrupt-Ereignis
- Ressourcen-Manager
- PXI-Instrument
- VXI/GPIB-VXI/VME MemAcc

☞ **Hinweis** ***Die Klassen Allgemeines Ereignis, Service-Request-Ereignis, Trigger-Ereignis, VXI-Signal-Ereignis, VXI/VME-Interrupt-Ereignis und Ressourcen-Manager können nur mit der Funktion VISA Close (= Schließen) und dem VISA Eigenschaften-Knoten als eine VISA-Session eingegeben werden.***

VISA-Funktionen variieren in der Klasse der **VISA-Session**, die mit ihnen verbunden werden können. Die gültigen Klassen für jede Funktion sind in der Dokumentation aufgeführt. Die Funktionen auf den Paletten **High Level Register Access** und **Low Level Register Access**, z.B., akzeptieren keine VISA-Sessions der Klasse GPIB-Instrument oder Serielles Instrument. Wenn eine **VISA-Session** mit einer Funktion verbunden wird, die die Klasse der Session nicht unterstützt oder wenn zwei VISA-Sessions unterschiedlicher Klassen verbunden werden, wird Ihr Diagramm zerstört und ein Fehler als ein Klassenkonflikt berichtet.

- **Fehlereingang**- und **Fehlerausgang**-Anschlüsse bestehen aus den Fehlerclustern in jeder VISA-Funktion. Ein Fehlercluster enthält drei Felder. Das Statusfeld ist ein Boolescher Wert, der TRUE ist, wenn ein Fehler eintritt, und FALSE, wenn kein Fehler auftritt. Das **Codefeld** ist ein VISA-Fehlercode-Wert, wenn während einer VISA-Funktion ein Fehler auftritt. Die VISA-Bibliotheksreferenz-Fehlercodes sind in Anhang A, *Fehlercodes*, aufgelistet. Das **Quellenfeld** ist ein String, der beschreibt, wo der Fehler aufgetreten ist. Durch Verbinden vom **Fehlerausgang** von jeder Funktion mit dem **Fehlereingang** der nächsten Funktion wird die erste Fehlerbedingung aufgezeichnet und bis zum Ende des Diagramms durchgegeben, wo sie an nur einer Stelle berichtet wird.

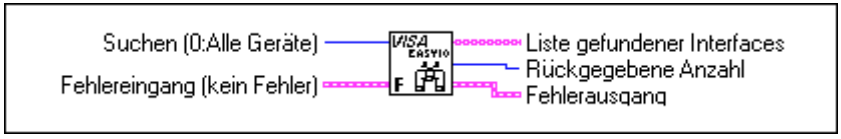
Beschreibungen der VISA-Operationen

Diese Funktionen erscheinen auf der **VISA**-Hauptpalette. Die für diese Funktionen gültigen Klassen sind Instrument (Standardeinstellung), GPIB-Instrument, Serielles Instrument, VXI/GPIB-VXI/VME RBD-Instrument und VXI/GPIB-VXI MBD-Instrument.

☞ **Hinweis** ***Die folgenden Leichte VISA-VIs stellen eine einfache Oberfläche zu den von Ihnen verwendeten Funktionen bereit. Wenn die Optimierung der Leistungsfähigkeit Ihrer Anwendung wichtig ist, sollten Sie die VISA-Primitives, die sich ebenfalls auf der Palette befinden, einsetzen.***

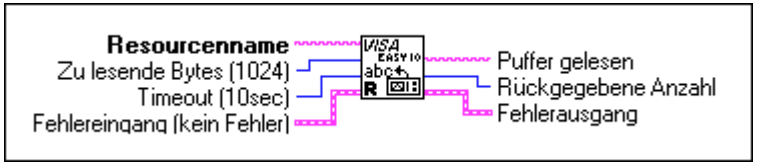
Easy VISA Find Resources (Einfach VISA Ressourcen finden)

Findet alle für die Kommunikation verfügbaren VXI-, Seriellen und GPIB-Ressourcen.



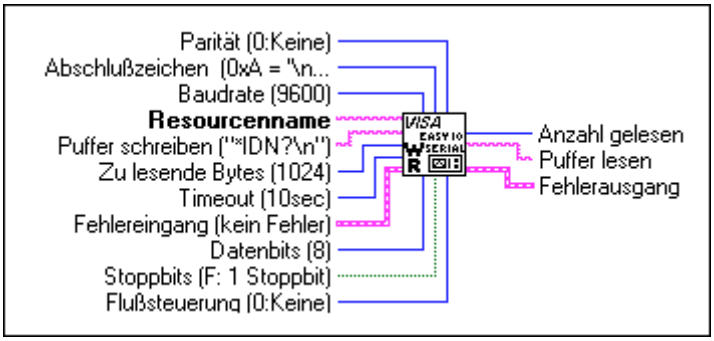
Easy VISA Read (Einfach VISA lesen)

Liest Daten von der durch Ressourcenname angegebenen Ressource. Die maximal zu lesende Byteanzahl wird durch Anzahl der Byte festgelegt.



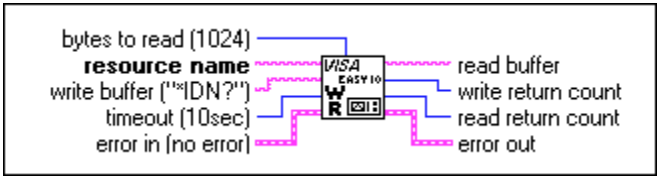
Easy VISA Serial Write and Read (Einfach VISA seriell schreiben und lesen)

Schreibt einen Befehlsstring an das angegebene serielle Gerät und liest dann die Antwortdaten. Mit Erhalt des vorgegebenen Abschlußzeichens oder der in dem Parameter Bytes zu lesen festgelegten Byteanzahl - je nachdem, was zuerst eintritt - wird das Lesen beendet. Wenn für das Schreiben zu dem Instrument ein Abschlußzeichen erforderlich ist, muß es in den Schreibpufferstring eingeschlossen werden.



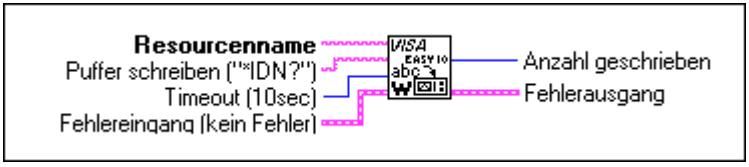
Easy VISA Write and Read (Einfach VISA schreiben und lesen)

Spricht das angegebene Gerät mit einem Befehlsstring an und liest dann die Anwortdaten.



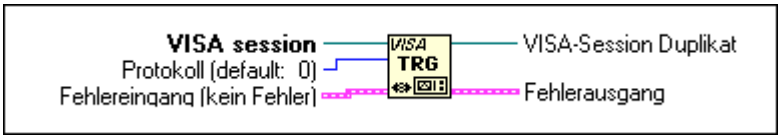
Easy VISA Write (Einfach VISA schreiben)

Spricht das angegebene Gerät mit einem Befehlsstring an.



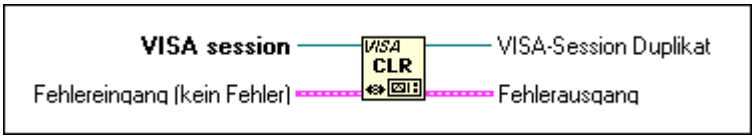
VISA Assert Trigger (Trigger durchsetzen)

Macht vom Typ der Schnittstelle abhängig einen Software- oder Hardware-Trigger geltend.



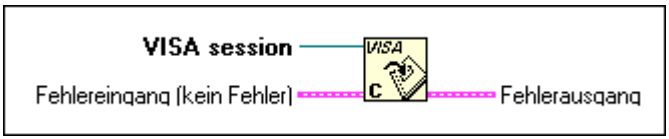
VISA Clear (Zurücksetzen)

Führt eine Zurücksetzung des Geräts gemäß IEEE488.1 durch. Bei VXI ist dies der Befehl Word Serial Clear; bei GPIB-Systemen ist dies der Befehl Selected Device Clear. Bei Seriellen sendet diese Funktion den String `*CLS In` (clear status input).



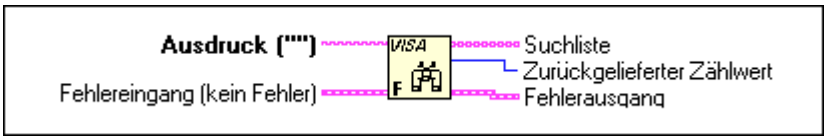
VISA Close (schließen)

Schließt eine vorgegebene Geräte-Session oder ein Ereignisobjekt. VISA Close akzeptiert alle verfügbaren Klassen. Schauen Sie die verfügbaren Klassen in der Auflistung weiter vorn im Abschnitt *VISA Referenzbibliothek-Parameter* nach.



VISA Find Resource (Ressource finden)

Fordert das System auf, die mit einer vorgegebenen Schnittstelle verknüpften Geräte zu finden.



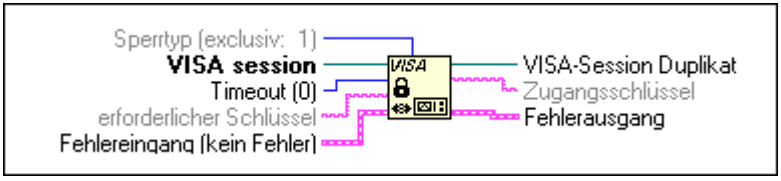
Die folgenden Tabellen enthalten die Beschreibungen der Ausdrucksparameter für das VI Find Resource.

Instrumentenressourcen	Ausdruck
GPIB	`GPIB[0-9]*::?*INSTR`
VXI	`VXI?*INSTR`
GPIB-VXI	`GPIB-VXI?*INSTR`
GPIB und GPIB-VXI	`GPIB?*INSTR`
Alle VXI	`?*VXI[0-9]*::?*INSTR`
Seriell	`ASRL[0-9]*::?*INSTR`
Alle	`?*INSTR`

Speicherressourcen	Ausdruck
VXI	`VXI?*MEMACC`
GPIB-VXI	`GPIB-VXI?*MEMACC`
Alle VXI	`?*VXI[0-9]*::?*MEMACC`
Alle	`?*MEMACC`

VISA Lock (sperren)

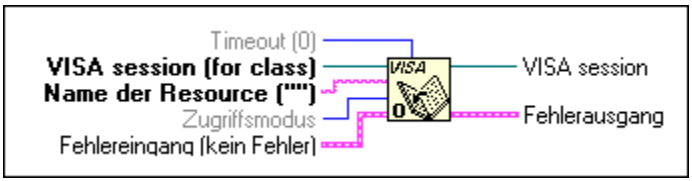
Richtet gesperrten Zugriff auf die angegebene Ressource ein.



Weitere Informationen über VISA locking und shared locking finden Sie in Kapitel 8, *Tutorial für LabVIEW VISA*, im *LabVIEW Benutzerhandbuch*.

VISA Open (öffnen)

Öffnet eine Session mit dem angegebenen Gerät und gibt einen Session-Identifikator aus, mit dem jede andere Operation dieses Geräts aufgerufen werden kann.



Die folgende Tabelle zeigt die Grammatik für den Adressenstring. Optionale Stringsegmente sind in eckigen Klammern angegeben ([ ]).

Schnittstelle	Syntax
VXI	`VXI[Karte]::VXI logische Adresse[::INSTR]`
GPIB-VXI	`GPIB-VXI[Karte]::VXI logische Adresse[::INSTR]`
GPIB	`GPIB[Karte]::Erstadresse[::Zweitadresse][::INSTR]`
ASRL	`ASRL[Karte][::INSTR]`
VXI	`VXI[Karte]::MEMACC`
GPIB-VXI	`GPIB-VXI[Karte]::MEMACC`

Das VXI-Schlüsselwort wird für VXI-Instrumente entweder über eingebettete oder MXI-Bus-Controller verwendet. Das GPIB-VXI-Schlüsselwort wird für einen GPIB-VXI-Controller verwendet. Das GPIB-Schlüsselwort kann zur Einrichtung der Kommunikation mit einem GPIB-Gerät verwendet werden. Das ASRL-Schlüsselwort wird zur Einrichtung der Kommunikation mit einem asynchronen seriellen Gerät, wie z.B. RS-232, verwendet.

Die folgende Tabelle zeigt den Standardwert für optionale Stringsegmente.

Optionale Stringsegmente	Standardwert
Karte	0
Zweitadresse	keine

Die folgende Tabelle zeigt Beispiele für Adressenstrings.

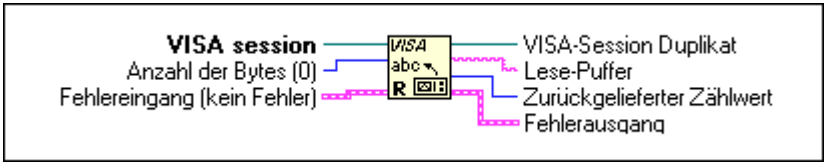
Adressenstring	Beschreibung
`VXI0::1::INSTR`	Ein VXI-Gerät unter der logischen Adresse 1 an der VXI-Schnittstelle VXI0.
`GPIB-VXI::9::INSTR`	Ein VXI-Gerät unter der logischen Adresse 9 in einem GPIB-VXI-gesteuerten System.
`GPIB::1::0::INSTR`	Ein GPIB-Gerät an einer GPIB-Schnittstelle 0 unter der Primäradresse 1 mit der Sekundäradresse 0.
`ASRL1::INSTR`	Ein an die Schnittstelle ASRL1 angeschlossenes Gerät.
`VXI::MEMACC`	Registerzugriff auf die VXI-Schnittstelle von der Platine aus.
`GPIB-VXI1::MEMACC`	Registerzugriff auf die GPIB-VXI-Schnittstellennummer 1 auf der Platinenebene.

Bei dem Parameter für den Zugriffsmodus wird der Wert `VI_EXCLUSIVE_LOCK (1)` zur Schaffung einer exklusiven Sperrung unmittelbar nach Öffnen einer Session verwendet; kann eine Sperrung nicht geschaffen werden, wird die Session geschlossen und ein Fehler ausgegeben.

Der Wert `VI_LOAD_CONFIG (4)` wird zur Konfiguration der Attribute auf Werte, die durch externe Konfigurationsdienstprogramme wie T&M Explorer (unter Windows 95/NT) oder VISAconf (unter Windows 3.*x*, Solaris 2 und HP-UX) festgelegt werden, verwendet.

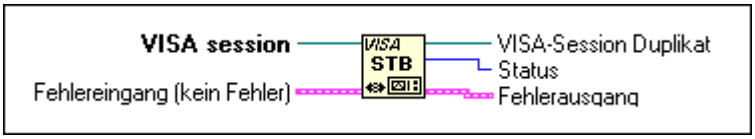
VISA Read (lesen)

Liest von einem Gerät Daten. Ob die Daten synchron oder asynchron übertragen werden, hängt von der Plattform ab.



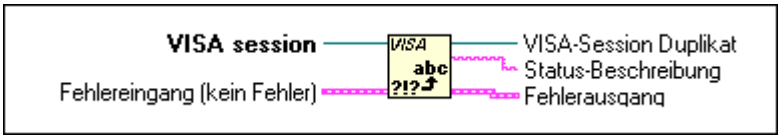
VISA Read STB (STB lesen)

Liest den Service-Anforderung-**Status** eines meldungsgestützten Geräts. Bei einer IEEE-488.2-Schnittstelle wird die Meldung von Abfragegeräten gelesen. Bei anderen Schnittstellentypen wird eine Meldung in Reaktion auf eine Service-Anforderung zum Einholen der Statusinformationen gesendet. Wenn die Statusinformation nur ein Byte lang ist, wird das höchstwertige Byte mit einem Wert von Null ausgegeben.



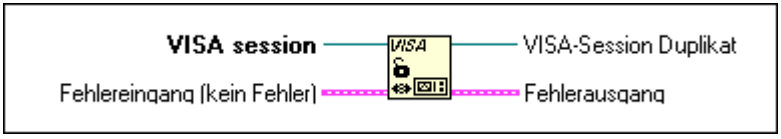
VISA Status Description (Statusbeschreibung)

Holt einen für Benutzer lesbaren String ein, der den Statuscode vom **Fehlereingang** darstellt.



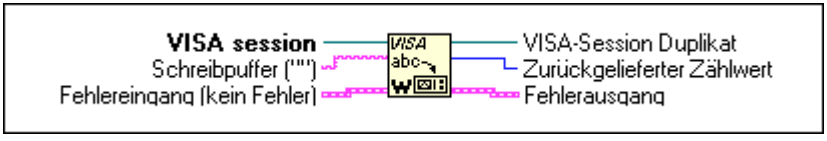
VISA Unlock (freigeben)

Gibt die zuvor mittels der Funktion VISA Lock eingerichtete Sperrung auf.



VISA Write (schreiben)

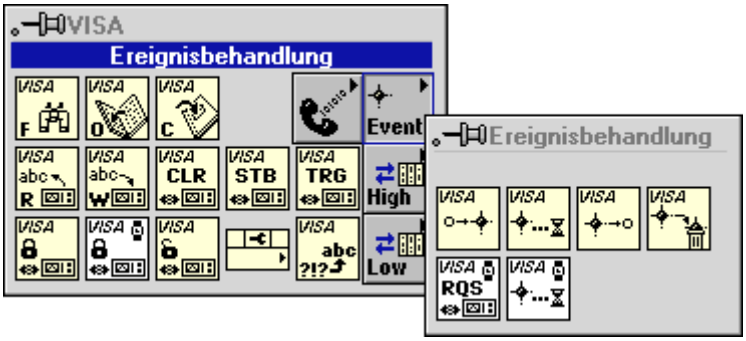
Schreibt Daten zum Gerät. Ob die Daten synchron oder asynchron übertragen werden, hängt von der Plattform ab.



Ereignisbehandlungsfunktionen

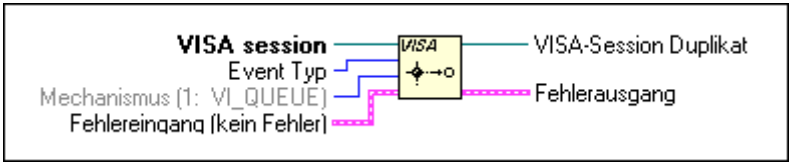
Dieser Abschnitt beschreibt die Ereignisbehandlungsfunktionen. Gültige Klassen für diese Funktionen sind Instr (Standardeinstellung), GPIB Instr, Serial Instr, VXI/GPIB-VXI/VME RBD Instr und VXI/GPIB-VXI MBD Instr.

Die VISA-Ereignisbehandlungsfunktionen befinden sich auf der **VISA**-Palette, die über das Menü **Funktionen»Instrumenten-I/O»VISA** aufgerufen wird.



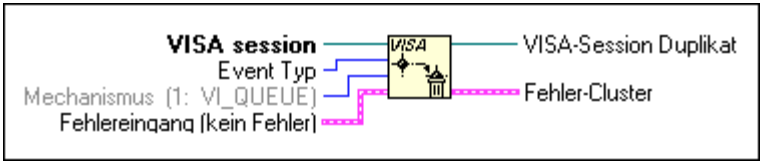
VISA Disable Event (Ereignis deaktivieren)

Deaktiviert die Betreuung eines Ereignisses. Diese Operation verhindert das Einreihen von neuen Ereignis-Occurrences. Bereits eingereihte Ereignis-Occurrences gehen jedoch nicht verloren; verwenden Sie das VI VISA Ereignis verwerfen, wenn eingereihte Ereignisse ausrangiert werden sollen.



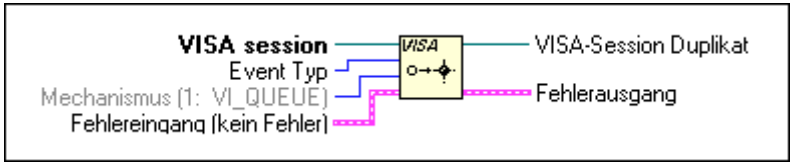
VISA Discard Events (Ereignisse verwerfen)

Sondert alle anhängigen Occurrences der angegebenen Ereignistypen und -mechanismen aus der angegebenen Session aus.



VISA Enable Event (Ereignis aktivieren)

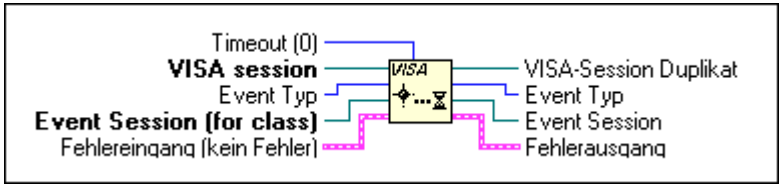
Schaltet die Benachrichtigung eines vorgegebenen Ereignisses ein.



☞ **Hinweis** ***Vor Verwenden des VIs VISA Wait on Event muß das VI VISA Enable Event für die jeweilige Session aufgerufen werden.***

VISA Wait On Event (Auf Ereignis warten)

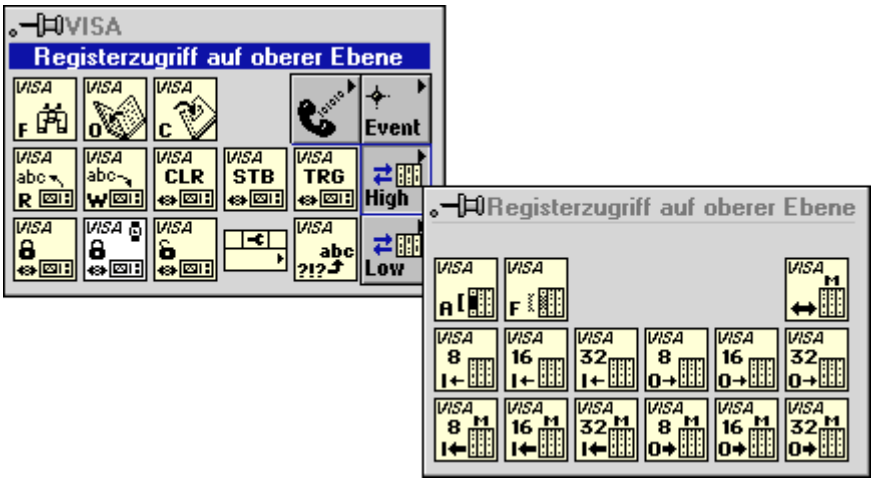
Stellt die Ausführung eines Anwendungs-Threads zurück und wartet einen maximalen Zeitraum lang, wie in **Timeout** festgelegt, auf ein Ereignis vom **Ereignistyp**. Kontextdefinitionen finden Sie in den einzelnen Ereignisbeschreibungen. Wenn es sich bei dem festgelegten **Ereignistyp** um Alle Ereignisse handelt, wartet die Operation auf ein beliebiges, für die gegebene Session aktiviertes Ereignis.



☞ **Hinweis** ***Vor Verwenden des VIs VISA Wait On Event muß für die angegebene Session das VI VISA Enable Event aufgerufen werden.***

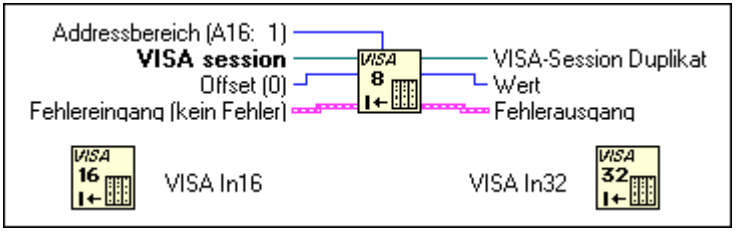
High-Level-Register-Access-Funktionen

Dieser Abschnitt beschreibt die VISA High-Level-Register-Access-Funktionen. Gültige Klassen für diese Funktionen sind Instr (Standardeinstellung), VXI/GPIB-VXI/VME RBD Instr, VXI/GPIB-VXI MBD Instr und VXI/GPIB-VXI/VME MemAcc. Sie greifen auf die VISA High-Level-Access-Funktionen zu, indem Sie das Popup-Menü des High-Level-Icons auf der **VISA**-Palette aufrufen.



VISA In8 / In16 / In32

Liest aus dem angegebenen Speicherplatz (zugewiesener Grundspeicher plus Zusatzspeicher) Daten in 8 Bits, 16 Bits bzw. 32 Bits. Für diese Funktionen muß vor dem Aufrufen die Funktion VISA Map Address (VISA-Adresse zuordnen) nicht aufgerufen werden.

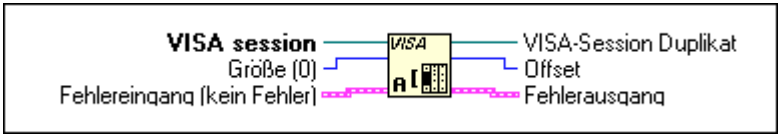


Die folgende Tabelle listet die gültigen Einträge zum Festlegen des Adreßbereichs auf.

Wert des Adreßbereichs	Beschreibung
(1)	Adressiert den Adreßbereich A16 des VXI/MXI-Bus'.
(2)	Adressiert den Adreßbereich A24 des VXI/MXI-Bus'.
(3)	Adressiert den Adreßbereich A32 des VXI/MXI-Bus'.

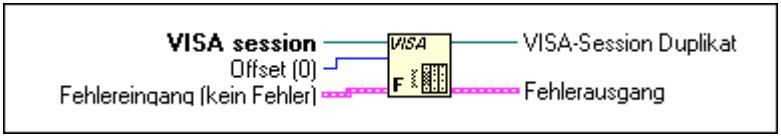
VISA Memory Allocation (Speicherzuweisung)

Gibt einen Offset in die Region eines Gerätes, die für diese Session zugewiesen wurde. Der Speicherplatz kann entweder auf dem Gerät selbst oder auf dem Systemspeicher des Computers zugewiesen werden. Wenn sich das Gerät, auf das sich eine bestimmte VISA Session bezieht, auf der lokalen Schnittstellenkarte befindet, kann der Speicherplatz entweder auf dem Gerät selbst oder auf dem Systemspeicher des Computers zugewiesen werden. Auf die Speicherregion, die der von dieser Funktion ausgegebene **Offset** anzeigt, kann mit den High-Level-Funktionen VISA Move In8 / Move In16 / Move In32 und VISA Move Out8 / Move Out16 / Move Out32 zugegriffen werden, oder sie kann mit Hilfe der Funktion VISA Map Address zugewiesen werden.



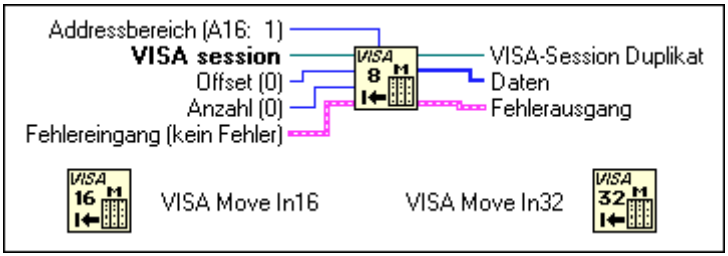
VISA Memory Free (Speicher räumen)

Macht Speicher frei, der vorher von der Funktion VISA Memory Allocation zugewiesen wurde. Wenn der **Offset** mit Hilfe der Funktion VISA Map Address zugewiesen wurde, muß die Zuweisung aufgehoben werden, ehe der Speicher freigemacht werden kann.



VISA Move In8 / Move In16 / Move In32

Verlagert einen Datenblock in Dateneinheiten von 8 Bit, 16 Bit bzw. 32 Bit vom Gerätespeicher zum lokalen Speicher. Die Funktion VISA Move In*XX* verwendet den angegebenen Adreßbereich zum Lesen von Daten in 8 Bit, 16 Bit bzw. 32 Bit vom angegebenen Offset. Für diese Funktionen muß die Funktion VISA Map Address nicht vor ihrem Aufrufen aufgerufen werden.

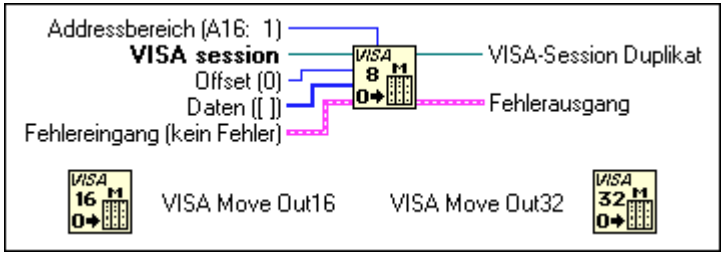


Die folgende Tabelle listet die gültigen Einträge zum Festlegen Adreßbereichs auf.

Wert	Beschreibung
(1)	Adressiert den Adreßbereich A16 des VXI/MXI-Bus'.
(2)	Adressiert den Adreßbereich A24 des VXI/MXI-Bus'.
(3)	Adressiert den Adreßbereich A32 des VXI/MXI-Bus'.

VISA Move Out8 / Move Out16 / Move Out32

Verlagert Datenblocks aus dem lokalen Speicher an die vorgegebene Adresse und den vorgegebenen Offset und verwendet den Adreßbereich, um Daten in 8 Bit, 16 Bit bzw. 32 Bit zu dem vorgegebenen Offset zu schreiben. Für diese Funktionen muß die Funktion VISA Map Address nicht vor ihrem Aufrufen aufgerufen werden.

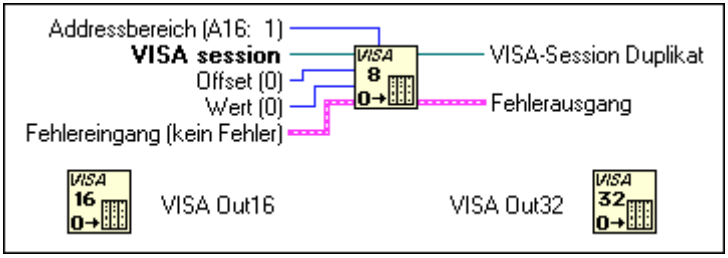


Die folgende Tabelle listet die gültigen Einträge zum Festlegen des Adreßbereichs auf.

Wert	Beschreibung
(1)	Adressiert den Adreßbereich A16 des VXI/MXI-Bus'.
(2)	Adressiert den Adreßbereich A24 des VXI/MXI-Bus'.
(3)	Adressiert den Adreßbereich A32 des VXI/MXI-Bus'.

VISA Out8 / Out16 / Out32

Schreibt Daten in 8 Bit, 16 Bit bzw. 32 Bit in den angegebenen Adreßbereich (zugewiesener Grundspeicher plus Offset). Für diese Funktionen muß die Funktion VISA Map Address nicht vor ihrem Aufrufen aufgerufen werden.

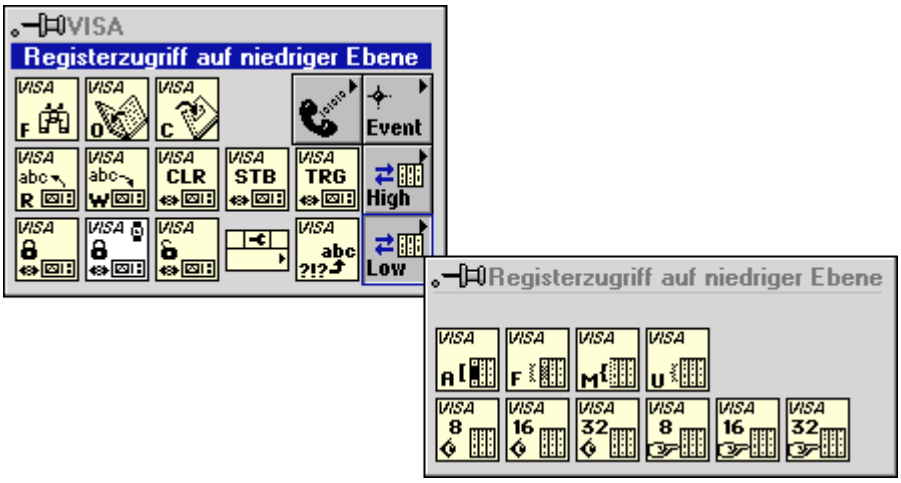


Die folgende Tabelle listet die gültigen Einträge zum Festlegen des Adreßbereichs auf.

Wert	Beschreibung
(1)	Adressiert den Adreßbereich A16 des VXI/MXI-Bus'.
(2)	Adressiert den Adreßbereich A24 des VXI/MXI-Bus'.
(3)	Adressiert den Adreßbereich A32 des VXI/MXI-Bus'.

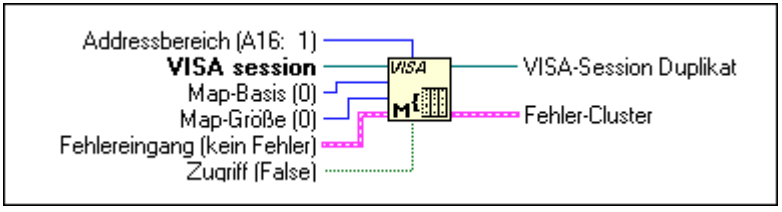
Low-Level-Register-Access-Funktionen

Dieser Abschnitt beschreibt die Funktionen VISA Low Level Register Access. Gültige Klassen für diese Funktionen sind Instr (Standardeinstellung), VXI/GPIB-VXIVXE RBD Instr, VXI/GPIB-VXI MBD Instr und VXI/GPIB-VXI/VME MemAcc. Rufen Sie das Popup-Menü vom Low-Level-Icon auf der **VISA**-Palette auf, um auf die Low-Level-Register-Access-Funktionen zuzugreifen.



VISA Map Address (Adresse zuweisen)

Weist einen vorgegebenen Speicherplatz zu. Der Platz, der zugewiesen wird, hängt von dem in **VISA-Session** und dem **Adreßbereich**-Parameter angegebenen Schnittstellentyp ab. Nach der Zuweisung des Fensters folgt VISA dem zugewiesenen Fenster. Durch dieses Verhalten ist es vorgegeben, daß VISA nur ein Fenster für jede VISA Session zuweisen kann.



Die folgende Tabelle listet die gültigen Einträge zum Festlegen des Adreßbereichs auf.

Wert	Beschreibung
(1)	Adressiert den Adreßbereich A16 des VXI/MXI-Bus'.
(2)	Adressiert den Adreßbereich A24 des VXI/MXI-Bus'.
(3)	Adressiert den Adreßbereich A32 des VXI/MXI-Bus'.

VISA Memory Allocation (Speicherzuweisung)

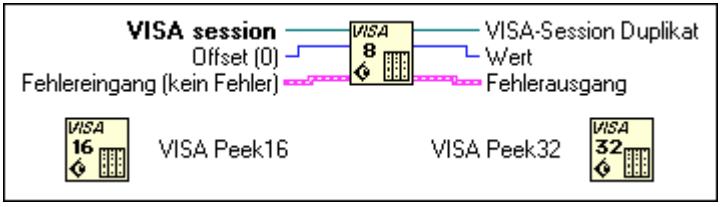
Für Informationen über die Funktion VISA Memory Allocation lesen Sie bitte den Abschnitt *High-Level-Register-Access-Funktionen* in diesem Kapitel.

VISA Memory Free (Speicher räumen)

Für Informationen über die Funktion VISA Memory Free lesen Sie bitte den Abschnitt *High-Level-Register-Access-Funktionen* in diesem Kapitel.

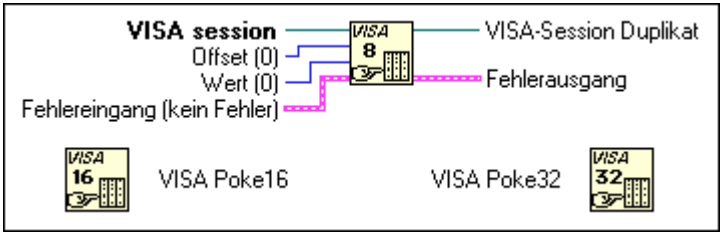
VISA Peek8 / Peek16 / Peek32

Liest einen 8-, 16- bzw. 32-Bit-Wert von der in **Offset** festgelegten Speicherstelle. Bei der Adresse muß es sich um eine gültige Speicheradresse handeln, die mit Hilfe eines vorangegangenen Aufrufs der Funktion VISA Map Address im aktuellen Prozeß zugewiesen wurde.



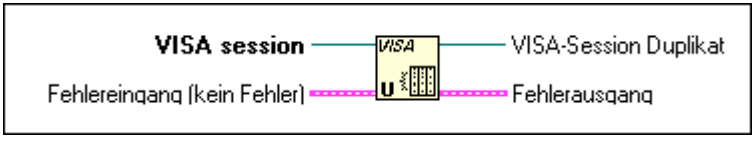
VISA Poke8 / Poke16 / Poke32

Schreibt einen 8-, 16- bzw. 32-Bit-Wert an die vorgegebenen Adresse und speichert den Inhalt des Werts an der Adresse, auf die **Offset** zeigt. Bei der Adresse muß es sich um eine gültige Speicheradresse handeln, die mit Hilfe eines vorangegangenen Aufrufs der Funktion VISA Map Address im aktuellen Prozeß zugewiesen wurde.



VISA Unmap Address (Adreßaufhebung)

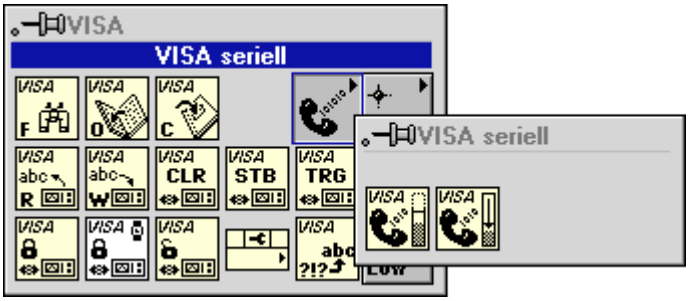
Hebt den Speicherplatz auf, der vorher von VISA Map Address zugewiesen wurde.



VISA Serial Functions (Serielle Funktionen)

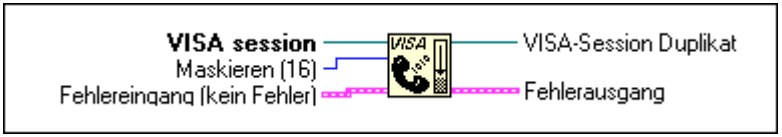
Dieser Abschnitt beschreibt die VISA-Funktionen, die sich auf serielle Schnittstellen beziehen. Gültige Klassen für diese Funktionen sind Instr (Standardeinstellung) und Serial Instr.

Sie greifen auf die Funktionen VISA Serial zu, indem Sie das Popup-Menü des Low-Level-Icons auf der **VISA**-Palette aufrufen.



Flush Serial Buffer (Seriellen Puffer leeren)

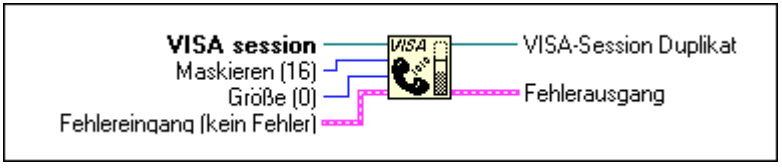
Leert den seriellen Puffer.



Leeren des Empfangspuffers (16) verwirft den Inhalt, wogegen Leeren des Ausgangspuffers (32) abwartet, daß eventuell im Übertragungspuffer befindlicher Inhalt zu dem Gerät gesendet wird. Um eventuell im Übertragungspuffer befindliche Daten zu verwerfen, muß Ausgabepuffermaske verwerfen (128) verwendet werden. Zum Leeren von mehr als einem Puffer gleichzeitig sind die Puffermasken durch einen O-Ring zu kombinieren.

Set Serial Buffer Size (Größe des seriellen Puffers einstellen)

Stellt die Größe des seriellen Puffers ein.

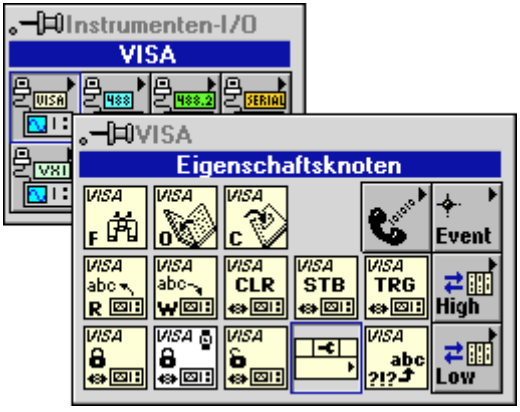


Gültige Werte von **Maske** sind Serieller Empfangspuffer (16) und Serieller Übertragungspuffer (32). Zum Einstellen der Größe von beiden Puffern sind die Puffermasken mit einem O-Ring zu kombinieren.

VISA Eigenschaftsknoten

Dieser Abschnitt beschreibt die VISA Bibliotheksattribute. Der VISA Eigenschaftsknoten erhält und/oder setzt die angezeigten Attribute. Der Knoten ist erweiterbar; die Bewertung erfolgt von oben nach unten, bis ein Fehler oder die abschließende Bewertung auftritt.

Auf den Eigenschaftsknoten greifen Sie zu, indem Sie im Menü **Funktionen»Instrumenten I/O»VISA** auswählen. Wählen Sie danach das Icon Eigenschaftsknoten aus der untersten Reihe auf der **VISA**-Palette aus.



Der VISA Eigenschaftsknoten zeigt nur die Attribute der verbundenen Klasse der Session an. Die Klasse eines VISA Eigenschaftsknotens kann solange verändert werden, wie er nicht mit einer **VISA-Session** verbunden wurde. Nach dem Verbinden einer **VISA-Session** mit einem VISA Eigenschaftsknoten paßt er sich an die Klasse der Session an, und alle angezeigten Attribute, die für diese Klasse nicht gültig sind, werden ungültig (angezeigt, indem das Attributobjekt schwarz wird).

Da der Eigenschaftsknoten in LabVIEW in anderen Zusammenhängen eingesetzt werden kann, kann sich der Eigenschaftsknoten u.U. auf die Standardeinstellungen einer anderen als einer VISA-Klasse einstellen, wenn er allein auf einem Diagramm plaziert wird. Wenn er mit einer **VISA-Klasse** verbunden wird, wandelt er sich in eine VISA-Klasse.

Beschreibungen der VISA-Eigenschaftsknoten

Die folgenden Kategorien von VISA-Eigenschaftsknoten sind verfügbar.

Schneller Datenkanal

Legt die folgenden Informationen fest:

- Kanalnummer
- Datenübertragung durch Kanalpaare
- Aktivierendes Signal
- Normalmodus oder Streaming-Modus für Übertragungen

Allgemeine Einstellungen

Bestimmen die folgenden Eigenschaften:

- Maximale Ereignisqueue-Länge
- Einmaliger VISA-Ressourcenname
- Sperrstatus der Ressource
- Zeitbegrenzungswert für den Zugriff auf das Gerät
- Triggermechanismus für Kommunikation
- Informationen zur Dokumentation der Funktionalität in Ihrer VISA-Anwendung

GPIB-Einstellungen

Legen die folgenden Informationen fest:

- Was die primären und sekundären Adressen eines GPIB-Gerätes sind
- Ob das GPIB-Gerät vor jeder Übertragung erneut adressiert werden muß
- Ob die Adressierung des GPIB-Geräts nach jeder Schreib- und Leseoperation aufgehoben wird

Schnittstellen-Informationen

Liefern Informationen über den Typ der VISA-Schnittstelle, die Platinennummer der Schnittstelle und die Platinennummer des Stammgeräts.

Meldungsgestützte Einstellungen

Legen die folgenden Aspekte von VISA meldungsgestützten Kommunikationen fest:

- Das I/O-Protokoll (GPIB, Seriell, VXI)
- Ob ein END-Bezeichner in Schreiboperationen gesendet werden muß
- Ob ein END-Bezeichner in Leseoperationen zu ignorieren ist
- Ob Leseoperationen mit einem Sonderzeichen abzuschließen sind

Modem-Leitungseinstellungen

Bestimmen den aktuellen Zustand der folgenden, bei Modemkommunikationen verwendeten Signale: CTS, DCD, DSR, DTR, RI und RTS.

PXI-Ressourcen

Legen den Adreßtyp, die Grundadresse und Adreßgröße von Geräten in den Steckplätzen BAR0 bis BAR5 fest.

PXI-Einstellungen

Legen die folgenden Informationen fest: Gerätenummer, Funktionsnummer, Herstellerinformationen zum Untersystem und den Modellcode des Untersystems.

Registergestützte Einstellungen

Bestimmen die folgenden Aspekte von VISA registergestützten Kommunikationen:

- Identifikation des Geräteherstellers
- Gerätemodell
- physikalische Lage des Gerätesteckplatzes
- Anzahl der Elemente in Blockverlagerungsoperationen an sowohl Quell- als auch Speicheradressen
- **(Windows)** Grundadresse, Adreßgröße und Zugriff auf Speicherplatz

Serielle Einstellungen

Legen folgendes fest: Byte-Anzahl an der seriellen Schnittstelle, Baudrate, Datenbits, Parität, Stoppbits, Flußregelung und Abschlußmethode von Lese- und Schreiboperationen.

Versions-Informationen

Liefern Informationen über die Version und den Herstellernamen der VISA-Implementierung sowie die Version der VISA-Spezifikation.

VME/VXE-Einstellungen

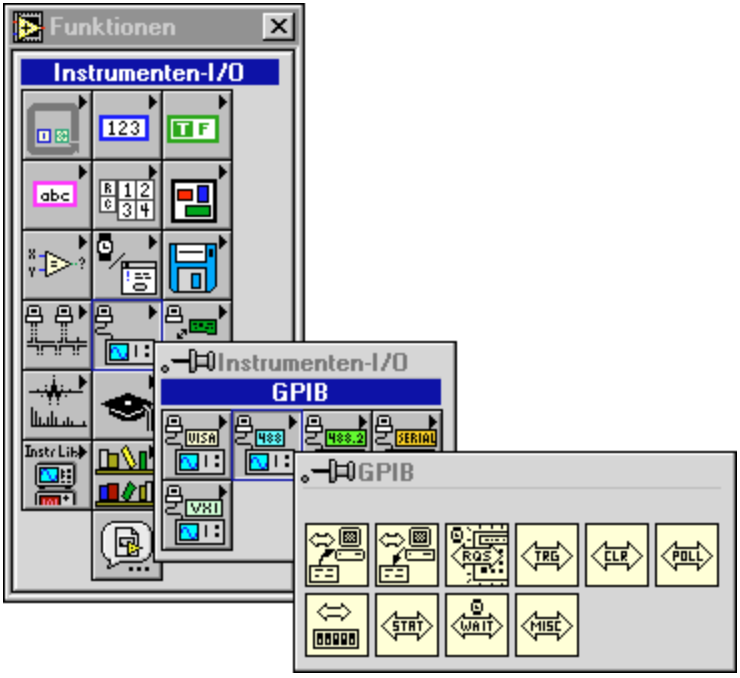
Bestimmen die notwendigen Adressen, die Zugriffsrechte, den Speicherplatz und die notwendige Byte-Reihenfolge für VXI-Kommunikationen.

34

Traditionelle GPIB-Funktionen

Dieses Kapitel beschreibt die Traditionellen GPIB-Funktionen.

Die folgende Abbildung zeigt die Palette **Traditionelle GPIB-Funktionen**, auf die über das Menü **Funktionen**» **Instrumenten- I/O»GPIB** zugegriffen werden kann.



Beispiele, wie die Traditionellen GPIB-Funktionen verwendet werden, finden Sie in `examples\instr\smplgpib.llb`.

Parameter der Traditionellen GPIB-Funktionen

Die Mehrzahl der Traditionellen GPIB-Funktionen verwenden die folgenden Parameter:

- **Adreßstring** enthält die Adresse des GPIB-Geräts, mit der die Funktion kommuniziert. In **Adreßstring** können sowohl die primäre als auch sekundäre Adresse in der Form von *Primäre+Sekundäre* eingegeben werden. Sowohl *Primäre* und *Sekundäre* sind Dezimalwerte, wenn also *Primäre* 2 ist und *Sekundäre* 3 ist, ist **Adreßstring** 2+3.

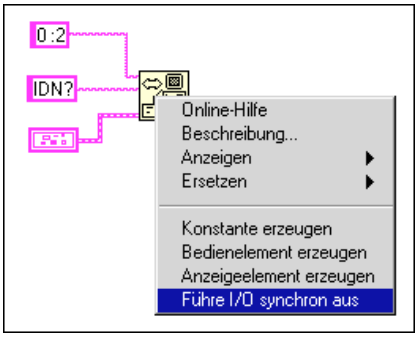
  Wenn keine Adresse angegeben wird, führen diese Funktionen keine Adressierung durch, bevor sie versuchen einen String zu lesen oder zu schreiben. Sie gehen davon aus, daß diese Befehle auf einem anderen Weg gesendet wurden oder ein anderer Controller beauftragt und daher für die Adressierung verantwortlich ist. Wenn der Controller das Gerät adressieren soll, dies aber nicht vor Ablauf der Zeitbegrenzung durchführt, enden die Funktionen mit GPIB-Fehler 6 (Timeout) und setzen Bit 14 in **Status**. Wenn der GPIB nicht der beauftragte Controller ist, ist kein **Adress-String** anzugeben.

  Wenn mehrere von LabVIEW verwendbare GPIB-Controller vorhanden sind, legt ein Präfix zum **Adress-String** in der Form `ID:address` (oder ID:, wenn keine Adresse erforderlich ist) den Controller fest, den eine bestimmte Funktion verwendet. Ist keine Controller-ID vorhanden, legen die Funktionen Controller (oder Bus) 0 zugrunde.
- **Status** ist ein 16-Bit-Array aus Booleschen Werten, in dem jedes Bit einen GPIB-Controller-Status beschreibt. Tritt ein Fehler auf, wird Bit 15 gesetzt. Das Feld **Fehlercode** des Clusters **Fehlerausgang** ist nur dann ein GPIB-Fehlercode, wenn Bit 15 vom **Status** gesetzt ist.
- Die Anschlüsse **Fehlereingang** und **Fehlerausgang** umfassen in jeder herkömmlichen GPIB-Funktion die Fehlerclusters. Der Fehlercluster enthält drei Felder. Das Statusfeld ist ein Boolescher Wert, der TRUE ist, wenn ein Fehler eintritt, FALSE, wenn kein Fehler auftritt. Das **Codefeld** ist ein GPIB-Fehlercodewert, wenn während der GPIB-Funktion ein Fehler eintritt. Das **Quellfeld** ist ein String zur Beschreibung, wo der Fehler aufgetreten ist. Wenn das **Statusfeld** des Parameters **Fehlereingang** an einer Funktion gesetzt ist, wird die Funktion nicht ausgeführt und derselbe Fehlercluster ausgegeben. Durch Verbinden des **Fehlerausgangs** von jeder Funktion mit dem **Fehlereingang** der nächsten Funktion wird die Fehlerbedingung aufgezeichnet und bis zum Ende des Diagramms durchgegeben, wo sie dann an einer einzigen Stelle berichtet wird.

Traditionelles Verhalten von GPIB-Funktionen

Die Funktionen GPIB Lesen und GPIB Schreiben verlassen das Gerät im adressierten Zustand, wenn sie mit der Ausführung fertig sind. Wenn Ihre Geräte das Arbeiten im adressierten Zustand nicht tolerieren können, müssen Sie die Funktion GPIB Versch. zum Senden der entsprechenden Adreßaufhebungsmeldung einsetzen oder die NI-488.2-Software zur automatischen Adreßaufhebung für alle Geräte am GPIB konfigurieren.

Die traditionellen Funktionen GPIB Lesen und GPIB Schreiben können asynchron arbeiten. Das bedeutet, daß LabVIEW-Aktivitäten weiter arbeiten können, während diese GPIB-Funktionen arbeiten.
Wenn asynchrone Ausführung eingestellt ist, erscheint ein kleines Armbanduhr-Icon als Teil der Funktion-Icons. Ein Popup-Objekt auf den traditionellen Funktionen GPIB Lesen und GPIB Schreiben erlaubt das Umschalten des Verhaltens zwischen synchroner und asynchroner Arbeitsweise.

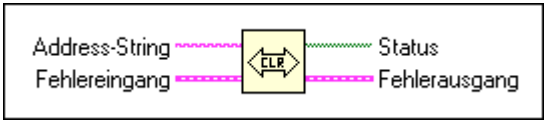


Beschreibungen der traditionellen GPIB-Funktionen

Die folgenden traditionellen GPIB-Funktionen sind verfügbar.

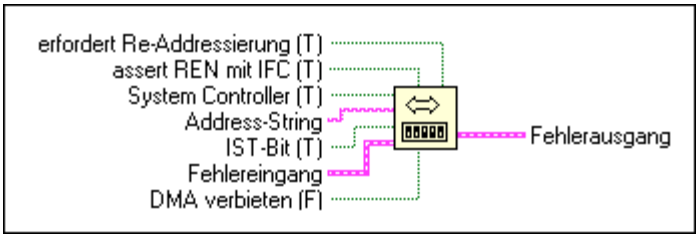
GPIB Zurücksetzen

Sendet entweder Selected Device Clear (SDC = Ausgewähltes Gerät zurücksetzen) oder Device Clear (DCL = Gerät zurücksetzen).



GPIB Initialisierung

Konfiguriert am **Adress-String** die GPIB-Schnittstelle.



GPIB Versch.

Führt die von **Command-String** vorgegebene GPIB-Operation aus. Verwenden Sie diese Low-Level-Funktion, wenn die vorher beschriebenen High-Level-Funktionen nicht angemessen sind.

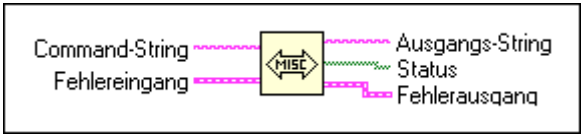


Tabelle 34-1. Command-Strings für Gerätefunktionen

Gerätefunktionen	Beschreibung
loc Adresse	Lokal schalten.
off Adresse	Gerät offline schalten.
pct Adresse	Steuerung übergeben.
ppc Byte-Adresse	Parallelabfrage konfigurieren (aktivieren oder deaktivieren).

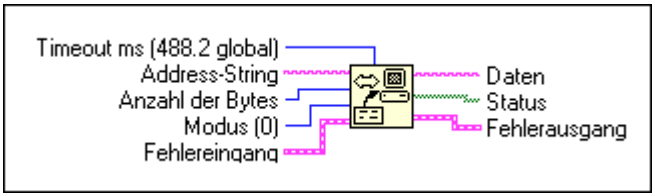
Tabelle 34-2. Command-Strings für Controller-Funktionen

Controller-Funktionen	Beschreibung
cac 0/1	Aktiver Controller werden.
cmd string	IEEE 488-Befehle senden.
dma 0/1	DMA-Modus oder programmierten I/O-Modus einstellen.
gts 0/1	Vom aktiven Controller in Bereitschaftszustand übergehen.
ist 0/1	Einzelnes Statusbit setzen.
llo	Lokale Fixierung.
loc	Controller in lokalen Zustand versetzen.
off	Controller offline schalten.
ppc byte	Parallelabfrage konfigurieren (aktivieren oder deaktivieren).
ppu	Parallelabfrage aller Geräte dekonfigurieren.
rpp	Parallelabfrage durchführen.
rsc 0/1	Systemsteuerung anfordern oder freigeben.
rsv byte	Service anfordern und/oder Statusbyte der seriellen Abfrage setzen.
sic	Interface clear senden und Fernsteuerung (Remote) aktivieren setzen.
sre 0/1	Fernsteuerung (Remote) aktivieren setzen oder zurücksetzen.

Legen Sie den GPIB-Controller für diese Funktionen fest, indem Sie einen **Command-String** in der Form ID: *xxx* verwenden, wobei ID der GPIB-Controller (Busnummer) ist und *xxx* der dreistellige Befehl und seine entsprechenden Argumente, falls vorhanden. Wenn keine Controller-ID angegeben wird, geht LabVIEW von 0 aus.

GPIB Lesen

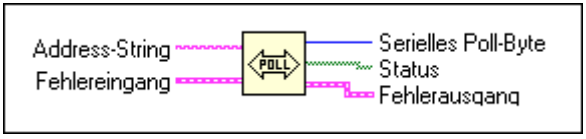
Liest die Byteanzahl vom **Byte-Count** des GPIB-Geräts am **Adress-String**.



Verwenden Sie die Funktion Timeout setzen zum Ändern des Standardwertes (den globalen 488.2-Timeout) von **Timeout ms**. Anfänglich setzt sich **Timeout ms** standardmäßig auf 10000. Weitere Informationen finden Sie im Abschnitt Beschreibung der Funktion Timeout setzen in Kapitel 35, *GPIB-488.2-Funktionen*.

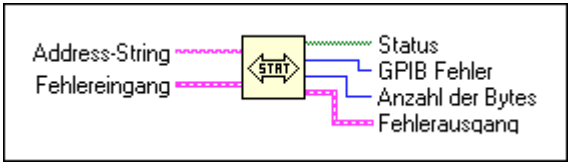
GPIB Serielle Abfrage

Führt eine serielle Abfrage vom durch **Adress-String** gekennzeichneten Gerät durch.



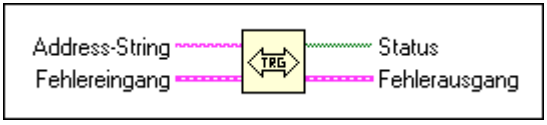
GPIB Status

Zeigt nach der vorangegangenen GPIB-Operation den Status des durch **Adress-String** angezeigten GPIB-Controllers an.



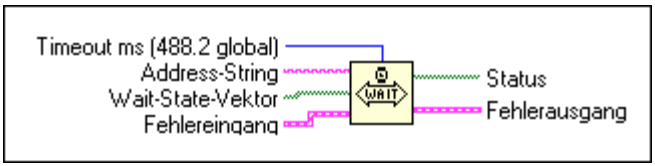
GPIB Trigger

Sendet GET (Group Execute Trigger) an das durch **Adress-String** angezeigte Gerät.



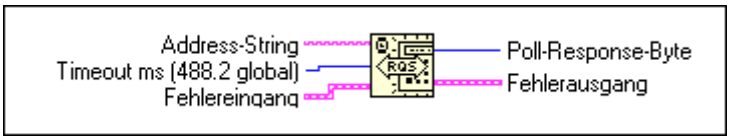
GPIB Warten

Wartet auf den/die durch **Wait-State-Vector** angezeigten Zustand/Zustände an dem Gerät, das durch **Adress-String** angezeigt wird.



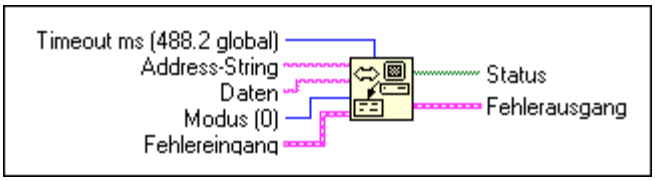
Wartet auf GPIB RQS

Wartet auf das durch **Adress-String** angezeigte Gerät**,** um RQS geltend zu machen.



GPIB Schreiben

Schreibt **Daten** zu dem durch **Adress-String** angezeigten GPIB-Gerät.



GPIB-Geräte- und -Controller-Funktionen

Dieser Abschnitt behandelt die in der Beschreibung der Funktion GPIB Versch. aufgelisteten Funktionen. Die Gerätefunktionen senden Konfigurationsinformationen an das angegebene Gerät. Die GPIB-Controller-Funktionen konfigurieren den Controller oder senden IEEE 488-Befehle, auf die alle Geräte antworten. Beachten Sie, daß es von den Befehlen **ppc** und **loc** sowohl Geräte- als auch Controllerversionen gibt. Die Syntax und der Einsatz der Befehle unterscheiden sich in jeder Version etwas.

Sie können diese Funktionen mit allen GPIB-Controllern einsetzen, die durch LabVIEW zugänglich sind, außer wenn in der Beschreibung der Funktion weiter unten anders angegeben. Ein ECMD-Fehler (17) ergibt sich, wenn eine Funktion mit einem GPIB-Controller ohne die angegebene Fähigkeit ausgeführt wird. Die Funktionssyntax ist streng festgelegt. Jede Funktion erkennt nur Kleinbuchstaben an und erlaubt nur eine Leerstelle zwischen dem Funktionsnamen und den Argumenten.

Gerätefunktionen

loc – Go to local (lokal schalten)

Syntax **loc** *Adresse*

loc bewegt Geräte vorübergehend von einem Programm-Fernsteuerungsmodus zu einem lokalen Modus

Adresse ist die GPIB-Adresse des Gerätes. Dieses Argument zeigt sowohl die primäre als auch die sekundäre Adresse an, wenn Sie die Form *primäre + sekundäre* verwenden, wobei *primäre* und *sekundäre* Dezimalwerte der primären und sekundären Adressen sind. Z.B., wenn *primäre* 2 ist und *sekundäre* 3, dann ist die *Adresse* 2 + 3.

loc sendet die Mitteilung Go To Local (GTL) an das GPIB-Gerät.

off – Take device offline (Gerät offline schalten)

Syntax **off** *Adresse*

off schaltet das Gerät an der angegebenen GPIB-Adresse offline. Dies ist nur notwendig, wenn das Gerät gemeinsam mit einer anderen Anwendung, die die NI-488-GPIB-Bibliothek verwendet, genutzt wird.

Adresse ist die GPIB-Adresse des Gerätes. Dieses Argument zeigt sowohl die primäre als auch die sekundäre Adresse an, wenn Sie die Form *primäre + sekundäre* verwenden, wobei *primäre* und *sekundäre* Dezimalwerte der primären und sekundären Adressen sind. Z.B., wenn *primäre* 2 ist und *sekundäre* 3, dann ist die *Adresse* 2 + 3.

pct – Pass control (Steuerung übergeben)

Syntax **pct** *Adresse*

pct übergibt die CIC-Autorität (Controller-in-Charge = beauftragter Controller) an das Gerät an der angegebenen Adresse. Der GPIB-Controller geht automatisch in Leerlauf über. Dabei geht die Funktion davon aus, daß das Gerät, an das **pct** die Steuerung übergibt, Controller-Fähigkeiten besitzt.

Adresse ist die GPIB-Adresse des Gerätes. Dieses Argument zeigt sowohl die primäre als auch die sekundäre Adresse an, wenn Sie die Form *primäre + sekundäre* verwenden, wobei *primäre* und *sekundäre* Dezimalwerte der primären und sekundären Adressen sind. Z.B., wenn *primäre* 2 ist und *sekundäre* 3, dann ist die *Adresse* 2 + 3.

pct sendet die folgenden Befehlsfolgen:

1. Sende-Adresse des Geräts (Talk-Adresse)
2. Sekundäre Adresse des Geräts, falls zutreffend
3. Steuerung übernehmen (TCT)

ppc – Parallel poll configure (Parallelabfrage konfigurieren)

Syntax **ppc** *Byte-Adresse*

ppc versetzt das Gerät in die Lage, auf Parallelabfragen zu antworten.

Byte ist 0 oder ein gültiger PPE-Befehl (Parallel Poll Enable). Wenn *Byte* 0 ist, wird das PPD-Byte 0x70 (Parallel Poll Disable = Parallelabfrage deaktiviert) gesendet, um das Gerät zu deaktivieren, auf Parallelabfragen zu antworten. Jede der 16 PPE-Mitteilungen wählt eine GPIB-Datenleitung (DIO1 bis DIO8) und Sinn (1 oder 0), die das Gerät verwenden muß, wenn es während einer Parallelabfrage auf die IDY-Mitteilung (Identify = Identifizieren) antwortet. Das Gerät vergleicht den **ist**-Sinn und steuert die DIO-Leitung mit TRUE oder FALSE an.

Adresse ist die GPIB-Adresse des Gerätes. Dieses Argument zeigt sowohl die primäre als auch die sekundäre Adresse an, wenn Sie die Form *primäre + sekundäre* verwenden, wobei *primäre* und *sekundäre* Dezimalwerte der primären und sekundären Adressen sind. Z.B., wenn *primäre* 2 ist und *sekundäre* 3, dann ist die *Adresse* 2 + 3.

Controller-Funktionen

cac – Become active Controller (Aktiver Controller werden)

Syntax **cac** 0 (übernimmt die Steuerung synchron)

cac 1 (übernimmt die Steuerung sofort)

cac übernimmt die Steuerung entweder synchron oder unmittelbar (und in einigen Fällen asynchron). Im allgemeinen braucht die Funktion **cac** nicht eingesetzt zu werden, weil andere Funktionen wie **cmd** und **rpp** die Steuerung automatisch übernehmen.

Wenn versucht wird, die Steuerung während eines stattfindenden Daten-Handshakes synchron zu übernehmen, schiebt die Funktion die Steuerungsübernahme solange auf, bis der Handshake abgeschlossen wurde. Wenn kein Handshake stattfindet, führt die Funktion die Steuerungsaktion sofort aus. Die synchrone Steuerungsübernahme ist nicht garantiert, wenn eine Lese- oder Schreiboperation mit einem Timeout oder anderem Fehler abschließt.

Die Steuerungsübernahme sollte asynchron erfolgen, wenn es unmöglich ist, die Steuerung synchron zu erlangen (z.B., nach einem Timeout-Fehler).

Der ECIC-Fehler ergibt sich, wenn der GPIB-Controller nicht der CIC (beauftragte Controller) ist.

cmd – Send IEEE 488 commands (IEEE 488-Befehle senden)

Syntax **cmd** *String*

cmd sendet GPIB-Befehlsmeldungen. Diese Befehlsmeldungen umfassen Geräte-Sende- und -Empfängeradressen, sekundäre Adressen, Konfigurationsmitteilungen für serielle und Parallelabfrage und Mitteilungen zum Rücksetzen und Triggern des Gerätes.

cmd wird nicht zur Übertragung von Programmieranweisungen an das Gerät eingesetzt. Die Funktionen GPIB Lesen und GPIB Schreiben übertragen Programmieranweisungen und andere, vom Gerät abhängige Informationen.

String enthält die vom Controller gesendeten Befehlsbytes. Diese Bytes werden in **cmd** *String* mit ASCII-Zeichen dargestellt. Wenn nichtanzeigbare Zeichen gesendet werden müssen, können Sie in String-Bedienelement und String-Konstante die Backslash-Codes

aktivieren oder eine Formatierfunktion zur Auflistung der Befehle im hexadezimalen Format verwenden.

dma – Set DMA mode or programmed I/O mode (DMA-Modus oder programmierten I/O-Modus einstellen)

Syntax **dma** 0 (programmierte I/O verwenden)

dma 1 (DMA verwenden)

dma zeigt an, ob die Datenübertragung DMA verwendet.

Einige GPIB-Platinen verfügen nicht über DMA-Fähigkeit. Wenn Sie versuchen, **dma** 1 auszuführen, wird fehlende DMA-Fähigkeit durch den GPIB-Fehler 11 angezeigt.

gts – Go from active Controller to standby (Vom aktiven Controller in Wartezustand wechseln)

Syntax **gts** 0 (kein Schatten-Handshaking)

gts 1 (Schatten-Handshaking)

Beschreibung:

gts versetzt den GPIB-Controller in den Controller-Wartezustand und gibt das ATN-Signal auf, wenn es der aktive Controller ist. Normalerweise ist der GPIB-Controller in die Datenübertragung einbezogen. **gts** erlaubt GPIB-Geräten, Daten ohne Beteiligung des GPIB-Controllers zu übertragen.

Wenn Schatten-Handshaking aktiviert ist, ist der GPIB-Controller in der GPIB-Übertragung zwar als Listener (Empfänger) beteiligt, akzeptiert jedoch keine Daten. Wenn er die END-Mitteilung erkennt, behauptet der GPIB-Controller NRFD (Not Ready For Data = Für Daten nicht bereit), um einen Handshake-Verzögerungs-Status zu erzeugen.

Wenn Schatten-Handshaking nicht aktiviert ist, führt der GPIB-Controller weder Schatten-Handshaking noch eine Handshake-Verzögerung aus.

Wenn Sie die Option Schatten-Handshake aktivieren, ist der GPIB-Controller in einem Daten-Handshake als Listener ohne eigentliches Lesen der Daten beteiligt. Er überwacht die Übertragung auf END-Mitteilungen und stoppt nachfolgende Übertragungen. Dieser Mechanismus erlaubt dem GPIB-Controller die Steuerung von nachfolgenden Operationen wie **cmd** oder **rpp** synchron zu übernehmen.

Nach dem Senden des Befehls **gts** sollte vor dem Auslösen eines weiteren GPIB-Befehls immer die END-Mitteilung abgewartet werden. Dies kann mit der Funktion GPIB Warten erfolgen.

Wenn der GPIB-Controller nicht der CIC ist, entsteht der ECIC-Fehler.

ist – Set individual status bit (Einzelnes Statusbit setzen)

Syntax **ist** 0 (Einzelner Statusbit ist zurückgesetzt)

ist 1 (Einzelner Statusbit ist gesetzt)

ist setzt den Sinn des einzelnen Statusbits (**ist = einzelner Status**).

ist wird eingesetzt, wenn der GPIB-Controller nicht der CIC ist, doch in einer Parallelabfrage beteiligt ist, die von einem Gerät geleitet wird, das der aktive Controller ist. Der CIC leitet eine Parallelabfrage durch Aufrechterhalten der EOI- und ATN-Signale, die die IDY-Mitteilung (Identifizieren) senden. Während diese Mitteilung aktiv ist, antwortet jedes zur Teilnahme konfigurierte Gerät durch Aufrechterhalten einer vorgegebenen GPIB-Datenleitung, entweder als TRUE oder FALSE, was vom Wert seines lokalen **ist**-Bits abhängig ist. Sie können z.B. den GPIB-Controller anweisen, die Datenleitung DIO3 mit TRUE anzusteuern, wenn **ist** 1 ist, und FALSE, wenn **ist** 0 ist. Sie können ihn auch umgekehrt anweisen, DIO3 mit TRUE anzusteuern, wenn **ist** 0 ist, und FALSE, wenn **ist** 1 ist.

Die für jedes Gerät aktuelle PPE-Mitteilung (Parallelabfrage aktivieren) bestimmt die Beziehung von dem **ist**-Wert, der angesteuerten Leitung und dem Sinn, mit dem die Leitung angesteuert wird. Der Controller ist in der Lage, diese Mitteilung entweder über **ppc** lokal oder über einen Befehl vom CIC remote zu empfangen. Wenn die PPE-Mitteilung läuft, wechselt **ist** den Sinn, mit dem der GPIB-Controller die Leitung während der Parallelabfrage ansteuert, und der GPIB-Controller kann eine One-Bit-, vom Gerät abhängige Mitteilung an den Controller vermitteln.

llo – Local lockout (lokale Fixierung)

Syntax **llo**

llo versetzt alle Geräte in den lokalen Fixierstatuts. Diese Aktion verhindert gewöhnlich die Erkennung von Eingaben vom Frontpanel des Geräts.

llo sendet den LLO-Befehl (Local Lockout = lokale Fixierung).

loc – Place Controller in local state (Controller in lokalen Status versetzen)

Syntax **loc**

loc versetzt den GPIB-Controller durch Senden der RTL-Mitteilung (Return To Local = Zurück zu lokal) in einen lokalen Status, wenn es nicht in den Fernsteuermodus eingesperrt ist (durch das LOK-Bit vom Status angezeigt). Verwenden Sie **loc** zur Simulation eines RTL-Schalters auf dem Frontpanel beim Einsatz eines Computers zur Simulation eines Geräts.

off – Take controller offline (Controller offline schalten)

Syntax **off**

off schaltet den Controller offline. Dies ist nur notwendig, wenn der Controller mit einer weiteren Anwendung gemeinsam benutzt wird, die die NI-488-Bibliothek verwendet.

ppc – Parallel poll configure (enable and disable) [Parallelabfrage konfigurieren (aktivieren und deaktivieren)]

Syntax **ppc** *Byte*

ppc konfiguriert den GPIB-Controller zur Teilnahme an einer Parallelabfrage durch Setzen seiner LPE-Mitteilung (Local Poll Enable = lokale Abfrage aktivieren) auf den Wert von *Byte*. Wenn der Wert von *Byte* 0 ist, dekonfiguriert sich der GPIB-Controller selbst.

Jede der 16 PPE-Mitteilungen (Parallel Poll Enable = Parallelabfrage aktivieren) wählt die GPIB-Datenleitung (DIO1 bis DIO8) und den Sinn (1 oder 0), die das Gerät beim Anworten auf die IDY-Mitteilung (Identifizieren) während einer Parallelabfrage verwenden muß. Das Gerät interpretiert die zugewiesene Mitteilung und den aktuellen Wert des einzelnen Statusbits (**ist**), um herauszufinden, ob die ausgewählte Leitung mit TRUE oder FALSE angesteuert ist. Wenn z.B. PPE=0x64 ist, ist DIO5 mit TRUE angesteuert, wenn **ist** 0 ist, und FALSE, wenn **ist** 1 ist. Wenn PPE=0x68 ist, ist die DIO1-PPE-Mitteilung in Kraft. Sie müssen wissen, welche PPE- und PPD-Mitteilungen gesendet werden und herausfinden, was die Antworten anzeigen.

ppu – Parallel poll unconfigure (Parallelabfrage dekonfigurieren)

Syntax **ppu**

ppu deaktiviert das Antworten von allen Geräten auf Parallelabfragen.

ppu sendet den PPU-Befehl (Parallel Poll Unconfigure = Parallelabfrage dekonfigurieren)

rpp – Conduct parallel poll (Parallelabfrage leiten)

Syntax **rpp**

rpp leitet eine Parallelabfrage von vorher konfigurierten Geräten durch Aufrechterhalten der ATN- und EOI-Signale, welche die IDY-Mitteilung senden.

rpp plaziert die Parallelabfrageantwort als ASCII-Zeichen in den Ausgabestring.

rsc – Release or request system control (Systemsteuerung freigeben oder anfordern)

Syntax **rsc** 0 (Systemsteuerung freigeben)

rsc 1 (Systemsteuerung anfordern)

rsc dient zum Freigeben oder Anfordern der Fähigkeit vom GPIB-Controller, die IFC- (Interface Clear = Schnittstelle zurücksetzen) und REN-Mitteilungen (Remote Enable = Remote aktivieren) mit Hilfe der sic- und sre- Funktionen an die GPIB-Geräte, zusenden. Damit der GPIB-Controller auf von einem weiteren Controller gesendeten IFC antwortet, darf der GPIB-Controller nicht der System-Controller sein.

In den meisten Anwendungen ist der GPIB-Controller immer der System-Controller. Setzen Sie **rse** nur ein, wenn der Computer nicht für die gesamte Dauer der Programmausführung der System-Controller ist.

rsv – Request service and/or set the serial poll status byte (Service anfordern und/oder den Statusbyte serielle Abfrage setzen)

Syntax **rsv** *Byte*

rsv setzt das serielle Abfragestatusbyte des GPIB-Controllers auf *Byte*. Wenn der 0x40-Bit in *Byte* gesetzt ist, fordert der GPIB-Controller auch Service vom Controller an, indem er die GPIB-RSQ-Leitung aufrechterhält. Wenn z.B. die GPIB-RSQ-Leitung aufrechterhalten werden soll, ist das ASCII-Zeichen @ zu senden, in dem das 0x40-Bit gesetzt ist.

Verwenden Sie **rsv** zur Anforderung von Service vom Controller mit Hilfe des SRQ-Signals (Service Request Signal = Service-Anforderungs-Signal) und zur Bereitstellung eines vom System abhängigen Statusbytes, wenn der Controller den GPIB-Anschluß seriell abfragt.

sic – Send interface clear (Interface clear senden)

Syntax **sic**

sic veranlaßt den Controller, das IFC-Signal für wenigstens 100 msec aufrechtzuerhalten, wenn der Controller über die System-Controller-Autorität verfügt. Die Aktion initialisiert den GPIB und macht den Controller-Anschluß zum CIC. **sic** wird im allgemeinen eingesetzt, wenn das Gerät der CIC werden oder eine Busfehlerbedingung beseitigt werden soll.

Das IFC-Signal setzt nur die GPIB-Funktionen der Bus-Geräte zurück; es setzt die internen Gerätefunktionen nicht zurück. Die Befehle DCL (Device Clear = Gerät zurücksetzen) und SDC (Selected Device Clear = Ausgewähltes Gerät zurücksetzen) setzen die Gerätefunktionen zurück. Schlagen Sie in der Dokumentation Ihres Geräts nach, um die Wirkungen dieser Mitteilungen herauszufinden.

sre – Unassert or assert remote enable (Remote aktivieren abbrechen oder aufrechterhalten)

Syntax **sre** 0 (Remote aktivieren abbrechen)

sre 1 (Remote aktivieren aufrechterhalten)

sre dient zum Abbrechen oder Aufrechterhalten der GPIB-REN-Leitung. Die Geräte überwachen REN, wenn sie zwischen lokalem und Remote-Betriebsmodus wählen. Ein Gerät tritt erst mit Eingang seiner Empfangsadresse wirklich in den Remote-Modus ein.

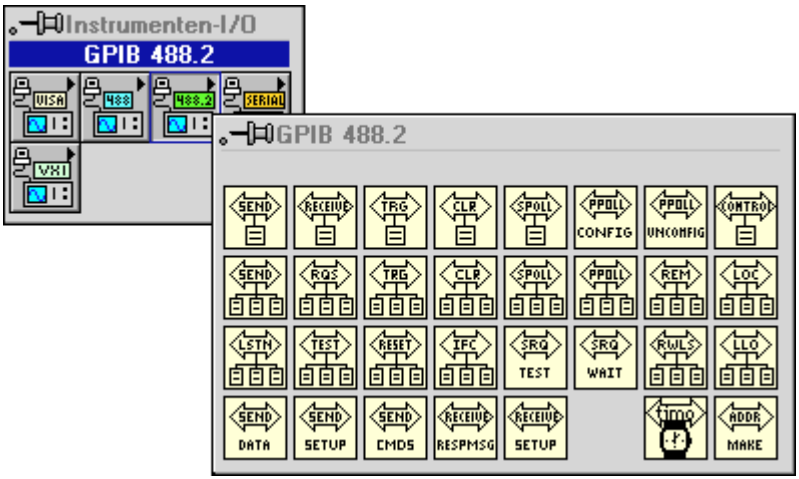
Der ESAC-Fehler tritt auf, wenn der Controller nicht der System-Controller ist.

35

GPIB-488.2-Funktionen

Dieses Kapitel beschreibt die IEEE- 488.2-(GPIB)-Funktionen.

Die folgende Abbildung zeigt die **GPIB-488.2**-Palette, auf die über das Menü **Funktionen»Instrumenten-I/O»GPIB-488.2** zugegriffen werden kann.



Beispiele für die Verwendung der GPIB-488.2-Funktionen finden Sie in `examples\instr\smplgpib.llb`.

Parameter der allgemeinen GPIB-488.2-Funktionen

Die Mehrzahl der GPIB-488.2-Funktionen verwenden die folgenden Parameter:

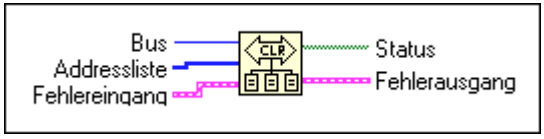
- **Adresse** enthält die primäre Adresse des GPIB-Gerätes, mit dem die Funktion kommuniziert. Wenn eine sekundäre Adresse erforderlich ist, ist die Funktion Adr erstellen einzusetzen, um die primäre und sekundäre Adresse in das richtige Format zu bringen. Wenn nicht anders angegeben, bestehen **Adresse** und **Adressliste** aus den Datentypen Integer bzw. Array aus Integers.
- Die standardmäßige primäre Adresse der GPIB-Platine lautet 0, ohne sekundäre Adresse. Sie wird als System-Controller angegeben. Der Standardwert von Timeout der Funktion beträgt 10 Sekunden. Sollte einer dieser Parameter geändert werden, müssen Sie das mit Ihrer GPIB-Platine gelieferte Konfigurationsdienstprogramm einsetzen. Außerdem können die Funktionen GPIB init und Timeout setzen zum Setzen der primären Adresse und zum Ändern des Timeout-Standardwertes während der Ausführungszeit eingesetzt werden; diese Funktionen beeinflussen die Schnittstelle jedoch nur, wenn sie mit LabVIEW verwendet werden. Weitere Informationen finden Sie in der mit Ihrer Hardware-Schnittstelle gelieferten Dokumentation.
- **Bus** bezieht sich auf die GPIB-Bus-Nummer. Wenn in Ihrem Computer nur eine Schnittstelle vorhanden ist, lautet die Bus-Standardnummer 0. Für Informationen über zusätzliche GPIB-Schnittstellen lesen Sie bitte die mit Ihrer GPIB-Schnittstelle gelieferten Software-Installationsanweisungen.
- **Anzahl der Bytes** bezieht sich auf die Anzahl von Bytes, die über den GPIB gegeben werden.
- **Status** ist ein Boolesches Array, in dem jedes Bit einen Zustand des GPIB-Controllers beschreibt. Wenn ein Fehler auftritt, setzen die GPIB-Funktionen Bit 15. **GPIB-Fehler** ist nur gültig, wenn Bit 15 vom **Status** gesetzt ist.
- **Fehlereingang; Fehlerausgang**. Lesen Sie in Kapitel 34, *Traditionelle GPIB-Funktionen*, den Abschnitt *Parameter der Traditionellen GPIB-Funktionen*.

Beschreibungen der GPIB-488.2-Funktionen (Funktionen für einzelne Geräte)

Einzelgerätfunktionen führen GPIB-I/O- und Steuerungsoperationen mit einem einzelnen GPIB-Gerät aus. Im allgemeinen akzeptiert jede Funktion die Adresse eines einzelnen Geräts als eine ihrer Eingaben.

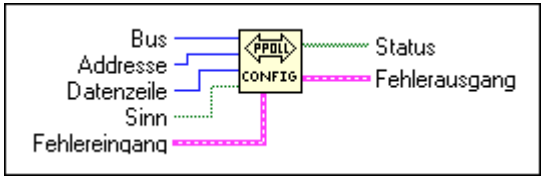
DevClear

Setzt ein einzelnes Gerät zurück. Um die SDC-Mitteilung (Selected Device Clear = ausgewähltes Gerät zurücksetzen) an mehrere GPIB-Geräte zu senden, verwenden Sie die Funktion DevClearList.



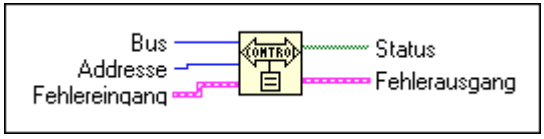
PPollKonfig

Konfiguriert ein Gerät für Parallelabfragen.



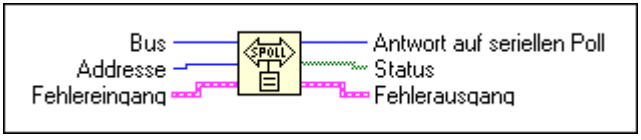
PassControl

Übergibt die Steuerung an ein anderes Gerät mit Controller-Fähigkeiten.



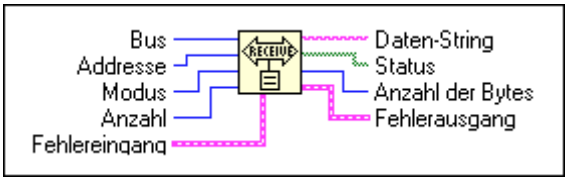
Status lesen

Fragt ein einzelnes Gerät seriell nach seinem Statusbyte ab.



Empfangen

Liest von einem GPIB-Gerät Datenbytes.

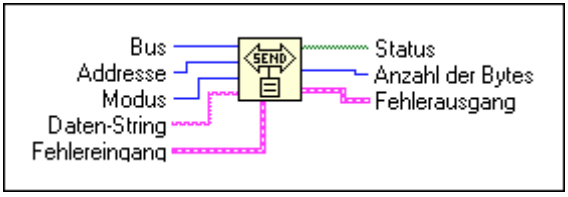


Die Funktion Empfangen endet, wenn sie eine der folgenden Aktionen ausführt:

- liest die angeforderte Anzahl von Bytes
- entdeckt einen Fehler
- überschreitet die Zeitbegrenzung
- entdeckt die END-Mitteilung (EOI aufrechterhalten)
- entdeckt das EOS-Zeichen (unter der Voraussetzung, daß der an **Modus** eingegebene Wert diese Option aktiviert hat)

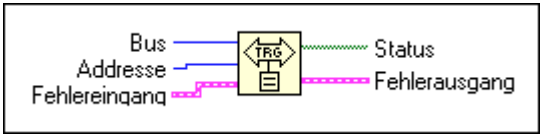
Senden

Sendet Datenbytes an ein einzelnes GPIB-Gerät.



Trigger

Triggert ein einzelnes Gerät. Um eine einzelne Mitteilung, die mehrere GPIB-Geräte triggert, zu senden, ist die Funktion TriggerListe zu verwenden.

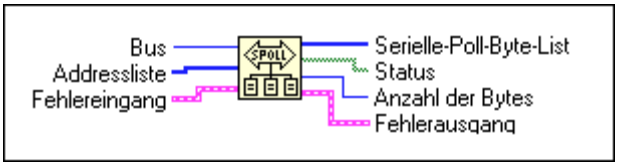


Beschreibungen der GPIB-488.2-Funktionen für mehrere Geräte

Die Funktionen für mehrere Geräte führen GPIB-I/O- und Steuerungsoperationen mit verschiedenen GPIB-Geräten gleichzeitig aus. Im allgemeinen akzeptiert jede Funktion ein Array aus Adressen als eine ihrer Eingaben.

AllSPoll (Alle seriell abfragen)

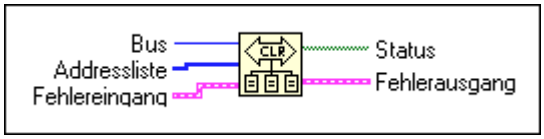
Fragt alle Geräte seriell ab.



Obwohl die Funktion AllSPoll im allgemeinen ausreicht, um eine beliebige Anzahl von GPIB-Geräten seriell abzufragen, sollte die Funktion Status lesen für die serielle Abfrage von nur einem GPIB-Gerät verwendet werden.

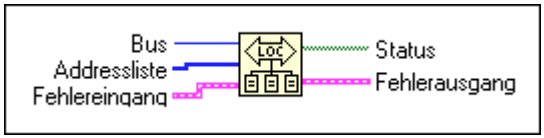
DevClearList

Setzt mehrere Geräte gleichzeitig zurück.



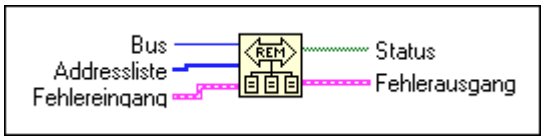
Lokale Variable aktivieren

Aktiviert den lokalen Modus für mehrere Geräte.



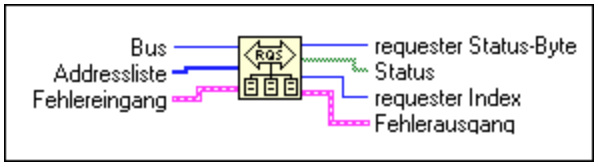
Remote aktivieren

Aktiviert die Remote-Programmierung mehrerer GPIB-Geräte.



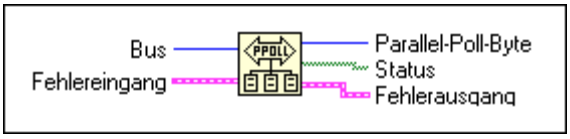
FindRQS

Findet das Service anfordernde Gerät heraus.



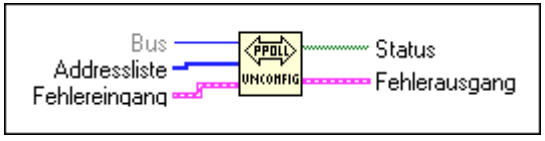
PPoll

Führt eine Parallelabfrage durch.



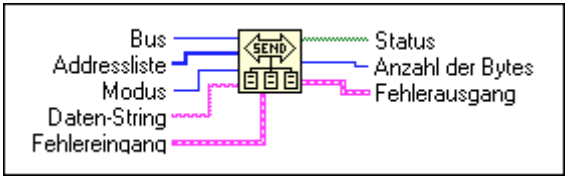
PPollUnkonfig

Unkonfiguriert Geräte für Parallelabfragen. Die Funktion unkonfiguriert die GPIB-Geräte, deren Adressen in dem **Adressliste**-Array für Parallelabfragen enthalten sind, d.h., daß sie an Abfragen nicht länger teilnehmen.



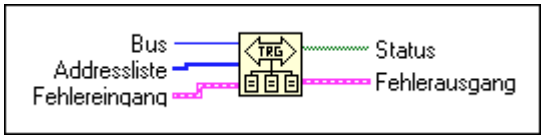
Liste senden

Sendet Datenbytes an mehrere GPIB-Geräte. Diese Funktion ähnelt der Funktion Senden, außer daß die Funktion Liste senden an mehrere Listener in einer Übertragung sendet.



TriggerListe

Triggert mehrere Geräte gleichzeitig.

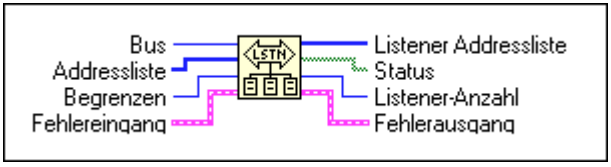


Beschreibungen der GPIB-488.2-Bus-Verwaltungsfunktionen

Die Bus-Verwaltungsfunktionen führen systemweite Funktionen aus oder berichten den Status systemweit.

FindLstn

Findet alle Listener auf dem GPIB. Diese Funktion wird normalerweise zum Entdecken der Anwesenheit von Geräten an bestimmten Adressen eingesetzt, da die meisten GPIB-Geräte über die Hören-Fähigkeit verfügen. Wenn sie entdeckt werden, können Sie gewöhnlich die Geräte fragen, um deren Identität herauszufinden.

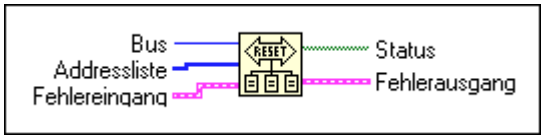


ResetSys

Führt die Initialisierung des Busses, des Mitteilungsaustausches und der Geräte durch. Zuerst erhält die Funktion REN (Remote Enable = Fernbetrieb aktivieren), von IFC (Interface Clear) gefolgt, aufrecht, wodurch die Adressierung aller Geräte aufgehoben wird und die GPIB-Platine (der System-Controller) zum CIC (Controller-in-Charge = beauftragter Controller) gemacht wird.

Zweitens sendet die Funktion die DCL-Mitteilung (Device Clear = Gerät zurücksetzen). Dadurch wird sichergestellt, daß alle IEEE-488.2-kompatiblen Geräte die RST-Mitteilung (Zurücksetzen) empfangen können.

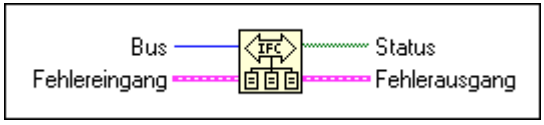
Drittens sendet die Funktion die *RST-Mitteilung an alle Geräte, deren Adresse in dem **Adressliste**-Array enthalten sind. Diese Mitteilung initialisiert vom Gerät abhängige Funktionen in jedem Gerät.



IFC senden

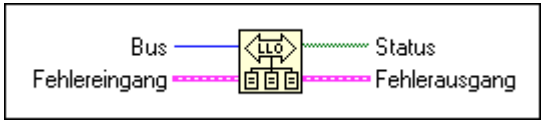
Setzt die GPIB-Funktionen durch IFC (Interface Clear = Schnittstelle zurücksetzen). Wenn die IFC-Mitteilung an die GPIB-Geräte ausgegeben wird, kehren die Schnittstellenfunktionen aller angeschlossenen Geräte in ihre Ausgangszustände zurück.

Diese Funktion sollte als Teil der GPIB-Initialisierung eingesetzt werden. Sie zwingt die GPIB-Platine in die Rolle des GPIB-Controllers und sichert ab, daß die Adressierungen der angeschlossenen Geräten völlig aufgehoben werden und sich die Schnittstellenfunktionen der Geräte im Leerlaufzustand befinden.



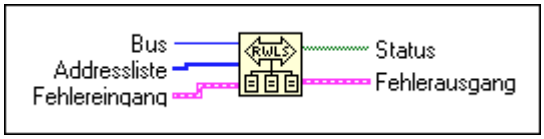
LLO senden

Sendet die LLO-Mitteilung (Local Lockout = lokale Fixierung) an alle Geräte. Wenn die Funktion die Mitteilung GPIB Local Lockout sendet, kann kein Gerät unabhängig den lokalen oder Remote-Status wählen. Während der Local Lockout in Kraft ist, kann einzig der Controller den lokalen oder Remote-Zustand der Geräte durch Senden der entsprechenden GPIB-Mitteilungen ändern. Die Funktion LLO senden sollte nur in ungewöhnlichen lokalen oder Remote-Situationen eingesetzt werden, besonders in solchen, in dem alle Geräte in den lokalen Programmierstatus gesperrt werden müssen. Verwenden Sie die Funktion RWLS setzen, wenn Sie die Geräte mit dem Lockout-Status in den Remote-Modus versetzen möchten.



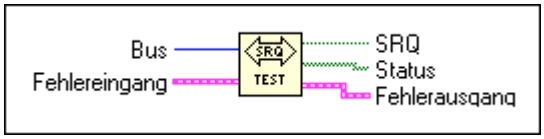
RWLS setzen

Versetzt bestimmte Geräte in den RWLS-Status (Remote With Lockout State = Remote-mit-Lockout-Status. Die Funktion sendet REN (Remote Enable = Remote aktivieren) an die in **Adressliste** aufgelisteten GPIB-Geräte. Daneben versetzt sie auch alle Geräte in den Lockout-Status, wodurch deren unabhängiges Zurückfallen in den lokalen Programmiermodus ohne Eingreifen des Controllers verhindert wird.



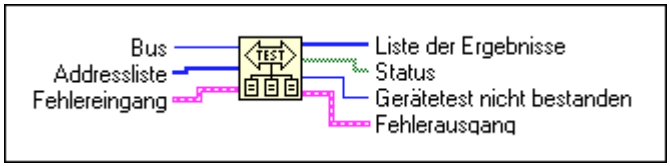
TestSRQ

Legt den aktuellen Status der SRQ-Leitung fest. Das Format dieser Funktion ähnelt dem der Funktion WaitSRQ, nur daß WaitSRQ sich selbst suspendiert, während es auf das Auftreten von SRQ wartet und TestSRQ sofort in den aktuellen SRQ-Status ausgibt.



TestSys

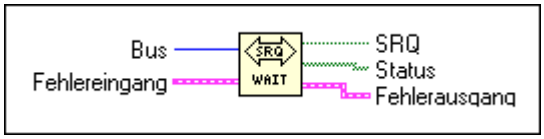
Weist mehrere Geräte an, IEEE-488.2-Selbsttests auszuführen.



WaitSRQ

Wartet, bis ein Gerät eine Service-Anforderung aufstellt. Die Funktion suspendiert die Ausführung, bis ein an den GPIB angeschlossenes GPIB-Gerät die SRQ-Leitung (Service Request = Service-Anforderung) aufstellt.

Das Format dieser Funktion ähnelt dem der Funktion TestSRQ, nur daß TestSRQ den SRQ-Status sofort ausgibt, wogegen WaitSRQ das Programm für die Dauer der Timeout-Periode (jedoch nicht länger) suspendiert, während es auf das Auftreten einer SRQ wartet.

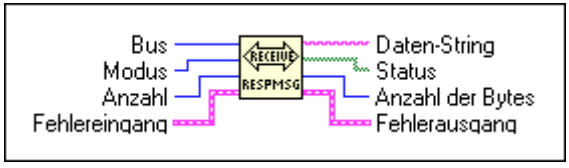


Beschreibungen der GPIB-488.2-Low-Level-I/O-Funktionen

Die Low-Level-Funktionen erlauben die Erstellung eines spezifischeren, detaillierteren Programms als die High-Level-Funktionen. Sie können die Low-Level-Funktionen für ungewöhnliche Situationen oder für Situationen, die zusätzliche Flexibilität erfordern, verwenden.

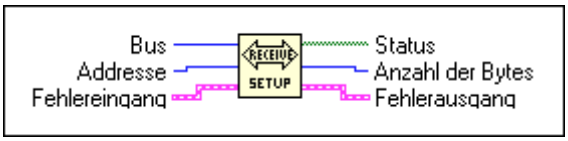
RcvRespMsg

Liest Datenbytes von früher adressierten Geräten. Diese Funktion geht davon aus, daß eine andere Funktion, wie ReceiveSetup, Empfangen oder SendCmds, die GPIB-Talker und -Listener bereits adressiert hat. Verwenden Sie die Funktion RcvRespMsg speziell zum Überspringen des Adressierungsschrittes der GPIB-Verwaltung. Normalerweise wird die Funktion Empfangen eingesetzt, um die gesamte Folge der Adressierungen auszuführen und danach die Datenbytes zu empfangen.



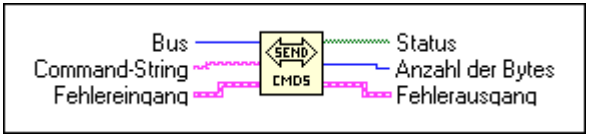
ReceiveSetup

Bereitet ein Gerät auf das Senden von Datenbytes und die GPIB-Platine auf das Lesen von Datenbytes vor. Nach Aufruf dieser Funktion kann eine Funktion wie RcvRespMsg zur Übertragung der Daten vom Talker (Sender) eingesetzt werden. Auf diese Weise wird die Notwendigkeit der Neuadressierung zwischen Leseblöcken ausgeschaltet.



SendCmds

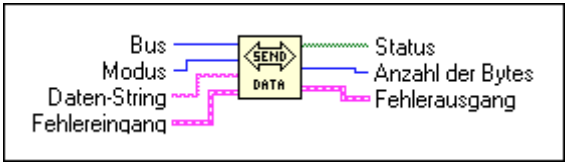
Sendet GPIB-Befehlsbytes.



Normalerweise braucht die Funktion SendCmds für GPIB-Operationen nicht eingesetzt zu werden. Sie wird eingesetzt, wenn spezialisierte Befehlsfolgen, die nicht von anderen Funktionen bereitgestellt werden, über den GPIB gesendet werden müssen.

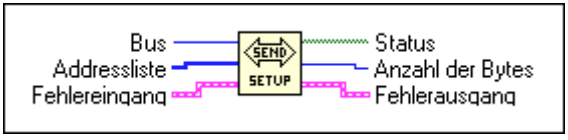
Datenbytes senden

Sendet Datenbytes an früher adressierte Geräte.



Setup senden

Bereitet bestimmte Geräte auf den Empfang von Datenbytes vor. Normalerweise wird ein Aufruf an diese Funktion von einem Aufruf einer Funktion wie Datenbytes senden gefolgt, um die Daten tatsächlich an die Listener zu übertragen. Diese Folge eliminiert die Notwendigkeit der Neuadressierung von Geräten zwischen gesendeten Blöcken.

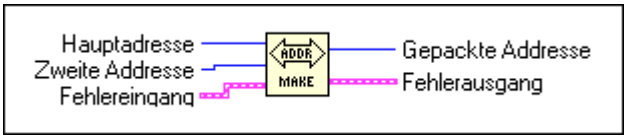


Beschreibungen der allgemeinen GPIB-488.2-Funktionen

Die allgemeinen Funktionen sind für spezielle Situationen nützlich.

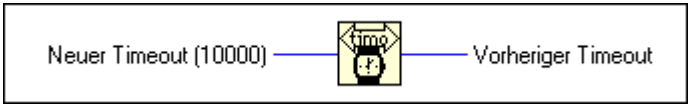
Adr. erstellen

Kombiniert **primäre Adresse** und **sekundäre Adresse** in eine speziell formatierte **Gepackte Adresse** für Geräte, die sowohl eine primäre als auch eine sekundäre GPIB-Adresse benötigen.



Timeout setzen

Verändert die globale Timeout-Periode für alle GPIB-488.2-Funktionen. Die Funktion setzt außerdem die Standard-Timeout-Periode für alle GPIB-Funktionen.

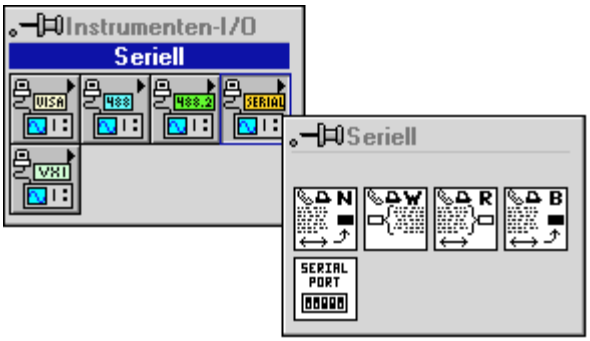


36

Serielle Anschluß-VIs

Dieses Kapitel beschreibt die VIs für Operationen mit seriellen Anschlüssen.

Die folgende Abbildung zeigt die **Seriell**-Palette, die über das Menü **Funktionen**»**Instrumenten-I/O**»**Seriell** aufgerufen werden kann.



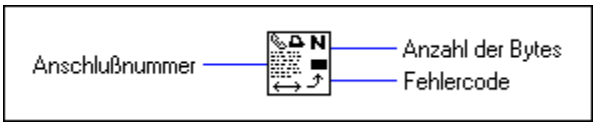
Beispiele für die Verwendung der VIs Serieller Anschluß finden Sie in `examples\instr\smplserl.llb`.

Beschreibungen der Seriellen Anschluß-VIs

Die folgenden Seriellen Anschluß-VIs sind verfügbar.

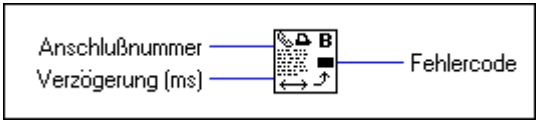
Bytes am seriellen Anschluß

Gibt an **Anzahl der Bytes** die Anzahl der Bytes im Eingangspuffer des durch **Anschlußnummer** vorgegebenen seriellen Anschlusses aus.



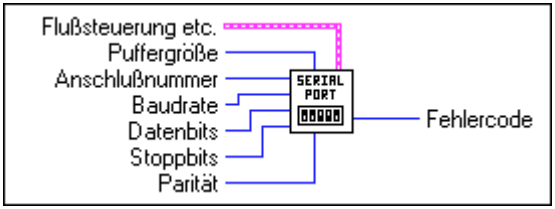
Serieller Anschluß - Pause

Sendet an den durch **Anschlußnummer** vorgegebenen Ausgangsanschluß eine Pause von mindestens der vom Eingang **Verzögerung** angeforderten Dauer.



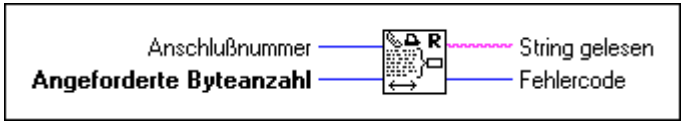
Seriellen Anschluß initialisieren

Initialisiert den ausgewählten seriellen Anschluß mit den vorgegebenen Einstellungen.



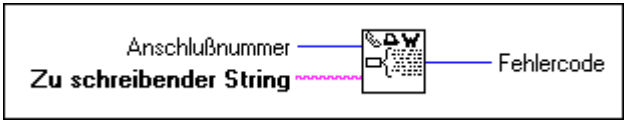
Von seriellem Anschluß lesen

Liest von dem durch **Anschlußnummer** angezeigten seriellen Anschluß die durch **angeforderte Byteanzahl** festgelegte Anzahl von Zeichen.



Auf seriellen Anschluß schreiben

Schreibt die Daten in **zu schreibender String** auf den in **Anschlußnummer** angegebenen seriellen Anschluß.



Teil IV

Analyse-VIs

Teil IV, *Analyse-VIs*, beschreibt die Analyse-VIs. In diesem Teil sind folgende Kapitel enthalten:

- Kapitel 37, *Einführung in die Analyse unter LabVIEW*, stellt die LabVIEW Analyse-VIs vor. Des weiteren ist eine Beschreibung enthalten, wie VIs organisiert sind, sowie Anweisungen zum Zugriff auf die VIs und zum Aufrufen der Online-Hilfe und eine Beschreibung der Analyse-VI-Fehlerberichte.
- Kapitel 38, *Signalerzeugungs-VIs*, beschreibt die VIs, die eindimensionale Arrays mit spezifischen Signalverlaufsmustern erzeugen.
- Kapitel 39, *VIs zur digitalen Signalverarbeitung*, beschreibt die VIs, die ein erfaßtes oder simuliertes Signal verarbeiten und analysieren. Die Digital-Signalverarbeitungs-VIs führen Frequenzbereichsanalysen, Zeitbereichsanalysen und andere Transformationen durch, wie die Fourier,- Hartley- und Hilbert-Transformationen.
- Kapitel 40, *Messungs-VIs*, beschreibt die Messungs-VIs, die darauf ausgelegt sind, DFT-gestützte und FFT-gestützte Analysen mit Signalerfassung für Frequenzmeßanwendungen, wie in typischen Frequenzmeßinstrumenten anzutreffen, wie dynamische Signalanalysierer, durchzuführen.
- Kapitel 41, *Filter-VIs*, beschreibt die VIs, die IIR-. FIR- und nichtlinerare Filter implementieren.
- Kapitel 42, *Fenster-VIs*, beschreibt die VIs, die Glättungsfenster implementieren.
- Kapitel 43, *Kurvenanpassungs-VIs*, beschreibt die VIs, die Kurvenanpassung oder Regressionsanalysen durchführen.

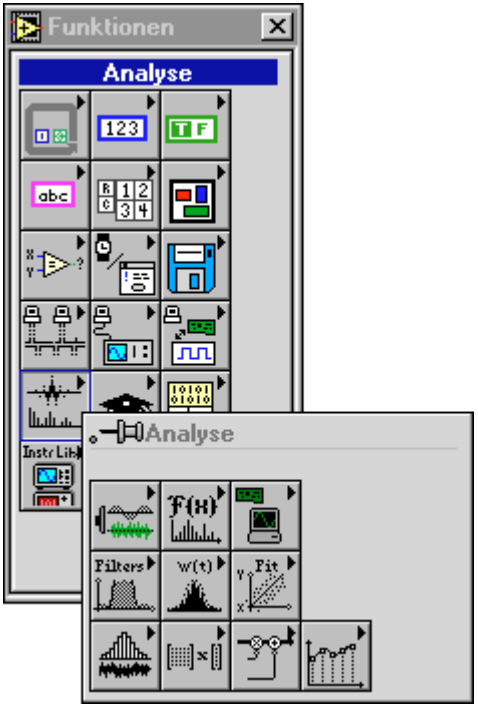
- Kapitel 44, *Wahrscheinlichkeits- und Statistik-VIs*, beschreibt die VIs, die Wahrscheintlichkeitsanalysen, beschreibende Statistik, Varianzanalysen und Interpolationsfunktionen durchführen.
- Kapitel 45, *Lineare-Algebra-VIs*, beschreibt die VIs, die mit Echt- und Komplexmatrizes verbundene Berechnungen und Analysen durchführen.
- Kapitel 46, *Arrayoperationen-VIs*, beschreibt die VIs, die häufige, ein- und zweidimensionale numerische Array-Operationen durchführen.
- Kapitel 47, *Zusätzliche Numerische Methoden-VIs*, beschreibt die VIs, die numerische Methoden zur Wurzelfindung, numerischen Integration und zur Peak-Detektion (Spitzenwertfindung) benutzen.

37

Einführung in die Analyse unter LabVIEW

Dieses Kapitel stellt die Analyse-VIs von LabVIEW vor, liefert eine Beschreibung ihrer Organisation, Anweisungen zum Zugriff auf die VIs und die auf Online-Hilfe sowie eine Beschreibung der Fehler-Berichterstattung von Analyse-VIs.

Sie können auf die **Analyse**-Palette vom Blockdiagramm-Fenster aus zugreifen, indem Sie im Menü **Funktionen»Analyse** auswählen; fahren Sie fort, indem Sie in den hierarchischen Menüs das gewünschte VI auswählen. Plazieren Sie das entsprechende VI-Icon auf dem Blockdiagramm und verbinden Sie es.



Full Development System

Die Analyse-VI-Grundbibliothek stellt einen Teilsatz der erweiterten Analyse-VI-Bibliothekdar, die im Full Development System zur Verfügung steht. Die Analyse-VI-Grundbibliothek umfaßt VIs zur statistischen Analyse, linearen Algebra und numerischen Analyse. Die erweiterte Analyse-VI-Bibliothek umfaßt weitere VIs in diesen Gebieten sowie VIs zur Signalerzeugung, Zeit- und Frequenzbereich-Algorithmen, Fensterung-Routinen, digitale Filter, Bewertungen und Regressionen.

Wenn die VIs der Grundbibliothek für Ihre Zwecke nicht ausreichen, können erweiterte LabVIEW-Analyse-Bibliotheken zum G-Grund-Paket hinzugefügt werden. Nach der Aufrüstung stehen Ihnen alle Analyse-Werkzeuge des Full Development Systems zur Verfügung.

Weitere Informationen über den Zugriff auf eine bestimmte Funktion oder VI-Palette finden Sie in den Kapiteln 38 bis 47, die jede Analyse-Unterpalette besprechen.

Überblick über die Analyse-VIs

Die Analyse-VIs von LabVIEW verarbeiten effektiv in digitaler Form dargestellte Informationsblöcke. Sie decken die folgenden Hauptbereiche der Verarbeitung ab:

- Pattern-Erzeugung
- Digitale Signalverarbeitung
- Auf Messungen beruhende Analyse
- Digitale Filterung
- Fensterglättung
- Wahrscheinlichkeit und statistische Analyse
- Kurvenanpassung
- Lineare Algebra
- Numerische Analyse

Die Analyse-VIs führen mit dem Zentralprozessor (CPU) und einem Fließkomma-Koprozessor (FPU) numerische Operationen aus. Viele der VIs nutzen den Vorteil der Simultanverarbeitungsfähigkeiten der CPU und FPU und verringern dadurch die Ausführungszeit von Datenanalyseaufgaben.

Die Analyse-VIs nutzen vor Ort vorhandene Datenverarbeitungsalgorithmen. Das heißt, daß die Algorithmen den Daten minimalen Speicherplatz zuweisen und die Daten innerhalb dieses Speicherplatzes bearbeiten. Durch die Verarbeitung vor Ort können größere Datenblöcke verarbeitet werden. Die einzige Speichereinschränkung für diese VIs besteht in der Größe des in Ihrem Computer verfügbaren Arbeitsspeichers (RAM). Anweisungen zum Konfigurieren der Speicherplatzzuweisungen für LabVIEW finden Sie im *LabVIEW Benutzerhandbuch.*

Die Analyse-VIs sind leistungsfähig genug, so daß Fachleute anspruchsvolle Analyseanwendungen schnell und effektiv erstellen können. Gleichzeitig sind sie einfach genug, so daß Neulinge Daten analysieren können, ohne spezialisierte Programmierer in DSP, digitaler Filterung, Statistik oder numerischer Analyse zu sein.

Organisation der Analyse-VIs

Nach der Installation erscheinen die zehn VI-Bibliotheken in der **Funktionenpalette**. Die **Analyse**-Palette umfaßt die folgenden Unterpaletten:

- **Signalerzeugung** enthält VIs, die digitale Pattern und Kurvenformen erzeugen.
- **Digitale Signalverarbeitung** enthält VIs, die Frequenzbereichstransformationen und -analysen, Zeitbereichsanalyse und andere Transformationen wie die Hartley- und Hilbert-Transformationen ausführen.
- **Messung** enthält VIs, die an Messungen orientierte Funktionen wie einseitige Spektren, skalierte Fensterung und Spitzenkraft- und Frequenzeinschätzungen ausführen.
- **Filter** enthält VIs, die IIR-, FIR-, und nichtlineare, digitale Filterungsfunktionen ausführen.
- **Fenster** enthält VIs, die Datenfensterung ausführen.
- **Wahrscheinlichkeit und Statistik** enthält VIs, die beschreibende statistische Funktionen, wie das Herausfinden des Durchschnitts oder der Standardabweichung eines Datensatzes, sowie inferentielle statistische Funktionen zur Wahrscheinlichkeit und Varianzanalyse (ANOVA) ausführen.
- **Kurvenanpassung** enthält VIs, die Kurvenanpassungs-Funktionen und Interpolationen ausführen.
- **Lineare Algebra** enthält VIs, die algebraische Funktionen an realen und komplexen Vektoren und Matrizen ausführen.

- **Array-Operationen** enthält VIs, die ein- und zweidimensionale numerische Array-Operationen wie lineare Bewertung und Skalierung ausführen.
- **Zusätzliche Num. Methoden** enthält VIs, die numerische Methoden zum Wurzelziehen, zur numerischen Integration und Spitzenwertfindung verwenden.

Die Ordner und VIs können Ihren Bedürfnissen und Anwendungen entsprechend umorganisiert werden. Sie können auch die ursprüngliche Struktur durch Entfernen der VIs von der Festplatte und Neuinstallieren von den Vertriebsdisketten wiederherstellen.

Festlegungen zur Schreibweise und Bezeichnung

Zum leichteren Erkennen der Parameter- und Operationstypen verwendet dieser Abschnitt des Handbuches die folgenden Festlegungen zur Schreibweise und Bezeichnung, soweit in einer VI-Beschreibung nicht anders angegeben. Obwohl es ein paar Skalarfunktionen und -operationen gibt, verarbeiten die meisten Analyse-VIs große Datenblöcke in der Form von 1D-Arrays (oder Vektoren) und 2D-Arrays (oder Matrizen).

Normale Kleinbuchstaben stellen Skalare oder Konstanten dar. z. B.

a,
π,
$b = 1{,}234$.

Großbuchstaben stellen Arrays dar. z. B.

X,
A,
$Y = a\,X + b$.

X und Y bezeichnen im allgemeinen 1D-Arrays, und A, B und C stellen Matrizen dar.

Array-Indizes beginnen in LabVIEW mit Null. Der Index des ersten Elements in dem Array ist unabhängig von dessen Dimensionen Null. Die folgende Zahlenfolge stellt ein 1D-Array X mit n-Elementen dar.

$$X = \{x_0, x_1, x_2, ..., x_{n-1}\}$$

Die folgende Skalargröße stellt das *i*-te. Element der Folge *X* dar.

$$x_i, \quad 0 \le i < n$$

Das erste Element der Folge ist x_0 und das letzte Element in der Folge ist x_{n-1}, bei einer Gesamtzahl von *n*-Elementen.

Die folgende Zahlenfolge stellt ein zweidimensionales Array mit *n*-Zeilen und *m*-Spalten dar.

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & \cdots & a_{0m-1} \\ a_{10} & a_{11} & a_{12} & \cdots & a_{1m-1} \\ a_{20} & a_{21} & a_{22} & \cdots & a_{2m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-10} & a_{n-11} & a_{n-2} & \cdots & a_{n-1m-1} \end{bmatrix}$$

Die Gesamtzahl der Elemente in dem 2D-Array ist das Produkt von *n* und *m*. Der erste Index korrespondiert mit der Zeilennummer und der zweite mit der Spaltennummer. Die folgende Skalargröße stellt das Element in der *i*-ten. Zeile und der *j*-ten. Spalte dar.

$$a_{ij}, 0 \le i < n \text{ und } 0 \le j < m$$

Das erste Element in *A* ist $a_{0\,0}$, und das letzte Element ist $a_{n-1\,m-1}$.

Soweit nicht anders angegeben verwendet dieses Handbuch die folgende vereinfachte Schreibweise für Array-Operationen.

Das Setzen eines Array-Elements zu einer Skalarkonstante wird wie folgt dargestellt:

$$X = a,$$

was der untenstehenden Folge entspricht

$$X = \{a, a, a, \ldots, a\}$$

und benutzt wird anstelle von

$$x_i = a, \quad \text{für } i = 0, 1, 2, \ldots, n-1.$$

Das Multiplizieren der Array-Elemente mit einer Skalarkonstanten wird wie folgt dargestellt:

$$Y = aX,$$

was der untenstehenden Folge entspricht

$$Y = \{ax_0, ax_1, ax_2, \ldots, ax_{n-1}\}$$

und benutzt wird anstelle von

$$y_i = ax_i, \quad \text{für } i = 0, 1, 2, \ldots, n-1.$$

Ähnlich wird das Multiplizieren eines 2D-Arrays mit einer Skalarkonstanten wie folgt dargestellt:

$$B = k\,A,$$

was der untenstehenden Folge entspricht

$$B = \begin{bmatrix} ka_{00} & ka_{01} & ka_{02} & \ldots & ka_{0m-1} \\ ka_{10} & ka_{11} & ka_{12} & \ldots & ka_{1m-1} \\ ka_{20} & ka_{21} & ka_{22} & \ldots & ka_{2m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ ka_{n-10} & ka_{n-11} & ka_{n-12} & \ldots & ka_{n-1m-1} \end{bmatrix}$$

und benutzt wird anstelle von

$$b_{ij} = ka_{ij}, \text{ für } i = 0, 1, 2, \ldots, n-1 \text{ und } j = 0, 1, 2, \ldots, m-1.$$

Leere Arrays sind in LabVIEW möglich. Ein Array ohne Elemente ist ein leeres Array und wird wie folgt dargestellt:

$$\text{Leer} = \text{NULL} = \varnothing = \{\ \}.$$

Operationen mit leeren Arrays führen im allgemeinen zu leeren Ausgabearrays oder nichtdefinierten Ergebnissen.

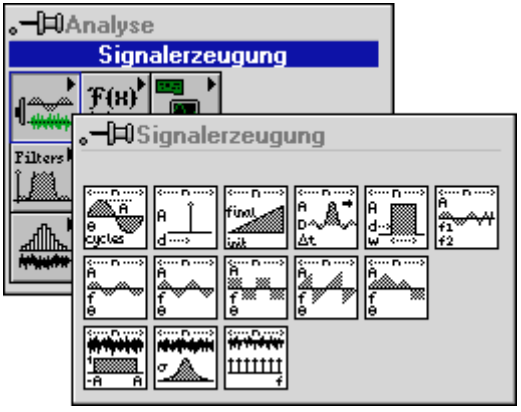
38

Signalerzeugungs-VIs

Dieses Kapitel beschreibt die VIs, die eindimensionale Arrays mit bestimmten Kurvenformen-Mustern erzeugen.

Diese VIs können mit den in Kapitel 4, *Numerische Funktionen*, besprochenen arithmetischen Funktionen eingesetzt werden, um anspruchsvollere Kurvenformen zu erzeugen. Wenn Sie z.B. einen amplitudenmodulierten Impuls erzeugen möchten, müssen Sie ein Impuls-Muster mit einem sinusförmigen Muster multiplizieren.

Auf die **Signalerzeugung**s-Palette kann über das Menü **Funktionen» Analyse»Signalerzeugung** zugegriffen werden. Die folgende Abbildung zeigt die auf der Palette **Signalerzeugung** verfügbaren Optionen.



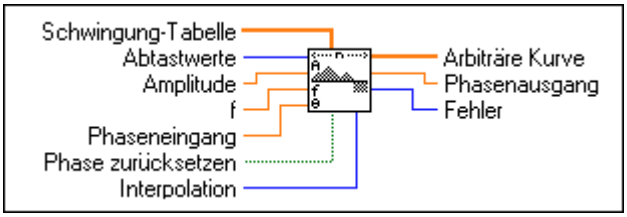
Beispiele der Verwendung dieser Signalerzeugungs-VIs finden Sie in `\analysis\sigxmpl.llb`.

Beschreibungen der Signalerzeugungs-VIs

Die folgenden Signalerzeugungs-VIs sind verfügbar.

Arbiträre Kurve

Erzeugt ein Array, das eine arbiträre Kurve enthält.



Wenn die Folge *y* eine **Arbiträre Kurve** darstellt, dann erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$y[i] = a * \text{arb}(\text{phase}[i]), \text{ mit } i = 0, 1, 2, \ldots, n-1,$$

wobei *a* die **Amplitude**, *n* die Anzahl der **Abtastwerte** ist

$$\text{arb}(\text{phase}[i]) = \text{WT}((\text{phase}[i] \text{ modulo } 360)*m/360)$$

und *m* die Größe des Arrays **Schwingung-Tabelle** ist.

Wenn **Interpolation** = 0 (keine Interpolation), dann *WT*(*x*) = **Schwingung-Tabelle**[int(*x*)].

Wenn **Interpolation** = 1 (lineare Interpolation), dann ist *WT*(*x*) gleich dem linear interpolierten Wert von **Schwingung-Tabelle**[int(*x*)] und **Schwingung-Tabelle**[(int(*x*)+1) modulo *m*].

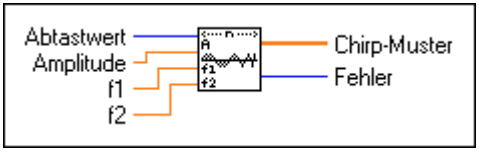
phase[*i*] = initial_phase + **f***360.0**i*, wobei **f** die Frequenz in normierten Einheiten von Zyklen/Abtastwert ist; initial_phase **Phaseneingang** ist, wenn **Phase zurücksetzen** TRUE ist, oder initial_phase der **Phasenausgang** von der vorangegangenen Ausführung dieses VIs ist, wenn **Phase zurücksetzen** FALSE ist.

Das VI ist ablaufinvariant, damit es zur Simulation einer fortlaufenden Erfassung von einem arbiträren Kurvenfunktionsgenerator eingesetzt werden kann. Wenn das Eingabebedienelement **Phase zurücksetzen** FALSE ist, erzeugen nachfolgende Aufrufe an eine bestimmte Instanz dieses VIs das Ausgabearray **Arbiträre Kurve**, das die nächsten Abtastwerte der arbiträren Kurve enthält.

Phasenausgang ist auf phase[*n*] eingestellt, und dieses ablaufinvariante VI nimmt diesen Wert als seinen neuen **Phaseneingang**, wenn bei der nächsten Ausführung dieses VIs **Phase zurücksetzen** FALSE ist.

Chirp-Muster

Erzeugt ein Array, das ein Chirp-Muster enthält.



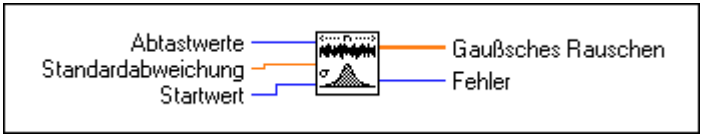
Wenn die Folge *Y* das **Chirp-Muster** darstellt, erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$y_i = A^* \sin((a/2\ i + b)\ i), \quad \text{für } i = 0, 1, 2, ..., n-1,$$

wobei *A* die **Amplitude** ist, $a = 2\pi(\mathbf{f2}-\mathbf{f1})/n$, $b = 2\pi\mathbf{f1}$, **f1** die Anfangsfrequenz in normierten Einheiten von Zyklen/Abtastwert ist, **f2** die Endfrequenz in normierten Einheiten von Zyklen/Abtastwert ist und *n* die Anzahl von **Abtastwert** ist.

Gaußsches weißes Rauschen

Erzeugt ein pseudozufälliges Muster mit Gaußscher Verteilung, dessen statistisches Profil $(\mu, \sigma) = (0, s)$ ist, wobei *s* der absolute Wert der angegebenen **Standardabweichung** ist.



Zur Erzeugung des Musters verwendet das VI eine modifizierte Version des Zufallszahl-Erzeugungsalgorithmus' mit sehr langen Zyklen, die auf dem zentralen Grenzwertsatz basiert. Angenommen, daß die Wahrscheinlichkeitsdichtefunktion *f*(*x*) von **Gaußsches Rauschen** mit der Gaußschen Verteilung wie folgt ist:

$$f(x) = \frac{1}{\sqrt{2\pi}s} e^{\left(-\frac{1}{2}\right)\left(\frac{x}{s}\right)^2},$$

wobei *s* der absolute Wert der angegebenen **Standardabweichung** ist, und daß die erwarteten Werte, $E\{\bullet\}$, mit folgender Gleichung berechnet werden können:

$$E(x) = \int_{-\infty}^{\infty} x f(x) dx,$$

dann sind der erwartete Mittelwert, μ, und der erwartete Wert der Standardabweichung, σ, der pseudozufälligen Folge:

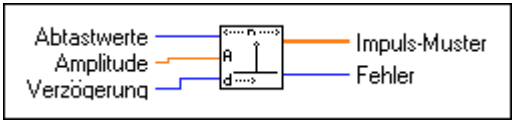
$$\mu = E\{x\} = 0,$$

$$\sigma = [E\{(x-\mu)^2\}]^{1/2} = s.$$

Die pseudozufällige Folge erzeugt ungefähr 2^{90} Abtastwerte, ehe sich das Muster wiederholt.

Impuls-Muster

Erzeugt ein Array, das ein Impuls-muster enthält.



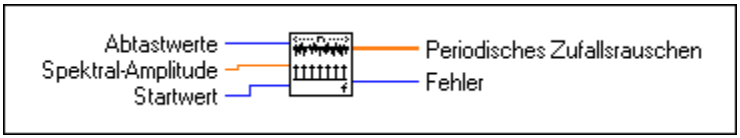
Wenn **Impuls-Muster** durch die Folge X dargestellt ist, erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$x_i = \begin{cases} a & \text{wenn } i = d \quad \text{for } i = 0, 1, 2, ..., n-1 \\ 0 & \text{anderweitig} \end{cases}$$

wobei *a* die **Amplitude**, *d* die **Verzögerung** und *n* die Anzahl der **Abtastwerte** ist.

Periodisches Zufallsrauschen

Erzeugt ein Array, das periodisches Zufallsrauschen (PRN) enthält.



Das Ausgabe-Array enthält alle Frequenzen, die mit einer Integralzahl von Zyklen in der angeforderten Anzahl von **Abtastwerte** dargestellt werden können. Jede Frequenzdomänen-Komponente hat einen Betrag aus **Spektral-Amplitude** und Zufallsphase.

Sie können sich ein Ausgabe-Array aus PRN als eine Summierung von Sinuslinien-Signalen mit denselben Amplituden, doch mit zufälligen Phasen vorstellen. Die Einheit von **Spektral-Amplitude** ist dieselbe wie die Ausgabe **Periodisches Zufallsrauschen** und ist ein lineares Maß der Amplitude, ähnlich anderen Signalerzeugungs-VIs.

Das VI erzeugt dieselbe periodische Zufallsfolge für einen eingegebenen positiven **Startwert**. Das VI gibt keinen neuen Startwert in den Zufallsphasengenerator ein, wenn **Startwert** negativ ist.

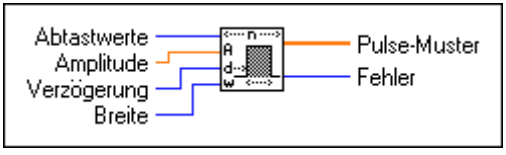
Die Ausgabefolge ist durch eine Amplitude von **spektral amplitude** * $\frac{\textbf{samples}}{2}$ begrenzt.

PRN kann zur Berechnung des Frequenzgangs eines linearen Systems von der Aufzeichnung eines einzelnen Zeitpunkts verwendet werden, anstatt den Mittelwert des Frequenzgangs aus Aufzeichnungen von mehreren Zeitpunkten zu bilden, wie es für Quellen mit unperiodischem Zufallsrauschen erforderlich ist.

PRN braucht vor dem Ausführen einer Spektralanalyse nicht gefenstert zu werden; PRN führt die Fensterung selbst aus und hat daher keine spektrale Streuung, weil PRN nur integralzyklische Sinuslinien enthält.

Pulse-Muster

Erzeugt ein Array, das ein Pulse-Muster enthält.



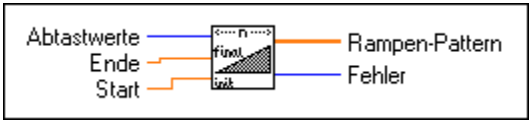
Wenn die Folge *X* das **Pulse-Muster** darstellt, erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$x_i = \begin{cases} a & \text{wenn } d \le i < (d + w) \quad \text{für } i = 0, 1, 2, \ldots, n-1. \\ 0,0 & \text{anderweitig} \end{cases}$$

wobei *a* die **Amplitude**, *d* die **Verzögerung**, *w* die **Breite** und *n* die Anzahl der **Abtastwerte** ist.

Rampen-Pattern

Erzeugt ein Array, das ein Rampen-Pattern enthält.



Wenn die Folge *X* das **Rampen-Pattern** darstellt, erzeugt das VI das Muster (Pattern) entsprechend der Gleichung:

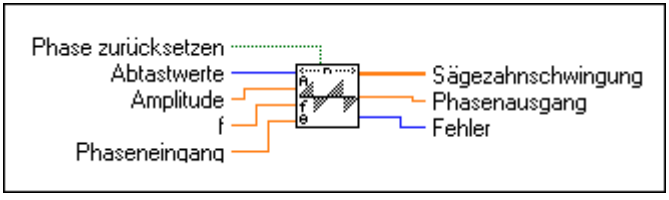
$$x_i = x_0 + i\Delta x \quad \text{für } i = 0, 1, 2, \ldots, n-1,$$

wobei $\Delta x = \frac{x_{n-1} - x_0}{n - 1}$, x_{n-1} **Ende**, x_0 **Start** und *n* die Anzahl der **Abtastwerte** ist.

Das VI legt der Beziehung zwischen **Start** und **Ende** keine Bedingungen auf. Dadurch kann es Aufwärtsrampen-Muster und Abwärtsrampen-Muster erzeugen.

Sägezahnschwingung

Erzeugt ein Array, das eine Sägezahnschwingung enthält.



Wenn *Y* die **Sägezahnschwingung** darstellt, erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$y[i] = a * \text{sawtooth}(\text{phase}[i]), \text{ für } i = 0, 1, 2, ..., n–1,$$

wobei *a* die **Amplitude** ist, *n* die Anzahl der **Abtastwerte** ist,

$$\text{sawtooth (phase[i])} = \begin{cases} \frac{p}{180} & 0 \le p < 180 \\ \frac{p}{180} - 2{,}0 & 180 \le p < 360 \end{cases}$$

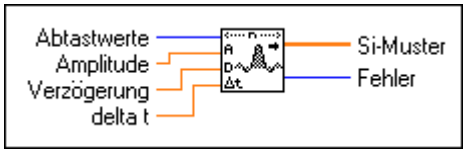
p = phase[*i*] modulo 360.0, phase[*i*] = initial_phase + **f***360.0**i*, **f** die Frequenz in normierten Einheiten von Zyklen/Abtastwert ist; initial_phase der **Phaseneingang** ist, wenn **Phase zurücksetzen** TRUE ist, oder initial_phase der **Phasenausgang** von der vorangegangenen Ausführung dieses VIs ist, wenn **Phase zurücksetzen** FALSE ist.

Das VI ist ablaufinvariant, damit es zur Simulation einer fortlaufenden Erfassung von einem Sägezahnschwingungs-Funktionsgenerator eingesetzt werden kann. Wenn das Eingabebedienelement **Phase zurücksetzen** FALSE ist, erzeugen nachfolgende Aufrufe an eine bestimmte Instanz dieses VIs das Ausgabearray **Sägezahnschwingung**, das die nächsten **Abtastwerte** der Sägezahnschwingung enthält.

Phasenausgang ist auf phase[*n*] eingestellt und, wenn **Phase zurücksetzen** FALSE ist, verwendet dieses ablaufinvariante VI bei der nächsten Ausführung diese Werte als seinen neuen **Phaseneingang**.

Si-Muster

Erzeugt ein Array, das ein Si-Muster enthält.



Wenn die Folge Y das **Si-Muster** darstellt, erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$y_i = \text{asinc}(i\Delta t - d), \text{ für } i = 0, 1, 2, \ldots, n–1,$$

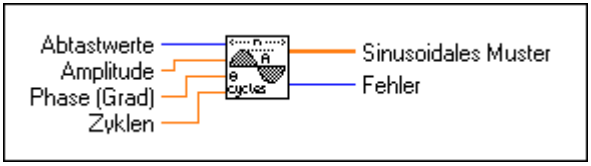
wobei $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$, *a* die **Amplitude** ist, Δt das Abtastintervall **delta t** ist, *d* die **Verzögerung** ist und *n* die Anzahl der **Abtastwerte** ist.

Die Hauptkeule der Si-Funktion, Si(x), ist der Teil der Si-Kurve, der im Bereich $-1 \le x \le 1$ eingeschlossen ist.

Wenn $|x| = 1$ ist, ist der Si(x) = 0,0; und der Spitzenwert der Si-Funktion tritt auf, wenn $x = 0$. Mit der l'Hôpital-Regel kann gezeigt werden, daß Si(0) = 1 und ihr Spitzenwert ist. Damit ist die Hauptfläche der Bereich der Si-Kurve, der von dem ersten Satz mit Nullen auf der linken und rechten Seite des Si-Wertes eingefaßt ist.

Sinusmuster

Erzeugt ein Array, das ein sinusförmiges Muster enthält.



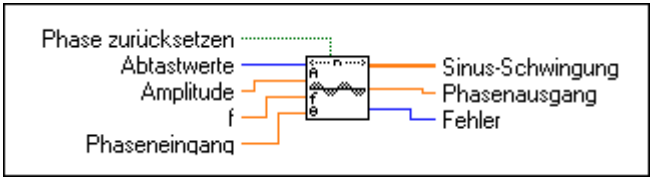
Wenn die Folge Y das **Sinusoidale Muster** darstellt, erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$y_i = a \sin(x_i) \text{ , für } i = 0, 1, 2, \ldots, n-1,$$

wobei $x_i = \frac{2\pi i\ k}{n} + \frac{\pi\phi_0}{180}$, a die **Amplitude** ist, k die Anzahl der Zyklen in dem Muster ist, ϕ_0 die initiale **Phase (Grad)** ist und n die Anzahl der **Abtastwerte** ist.

Sinus-Schwingung

Erzeugt ein Array, das eine Sinuswelle enthält.



Wenn die Folge Y die **Sinus-Schwingung** darstellt, erzeugt das VI ein Muster entsprechend der folgenden Gleichung:

$$y_i = a * \sin(\text{phase}[i]), \text{ für } i = 0, 1, 2, \ldots, n-1,$$

wobei a die **Amplitude** und phase[i] = initial_phase + **f***360.0*i ist; **f** die Frequenz in normierten Einheiten von Zyklen/Abtastwert ist; initial_phase der **Phaseneingang** ist, wenn

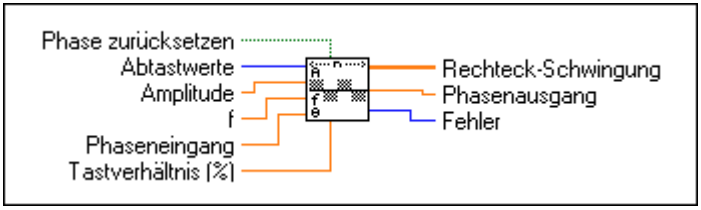
Phase zurücksetzen TRUE ist, oder initial_phase der **Phasenausgang** der vorangegangenen Ausführung dieser Instanz des VIs ist, wenn Phase zurücksetzen FALSE ist.

Das VI ist ablaufinvariant, daher kann es zur Simulation einer fortlaufenden Erfassung eines Sinuswellen-funktionsgenerators eingesetzt werden. Wenn das Bedienelement **Phase zurücksetzen** FALSE ist, erzeugen nachfolgende Aufrufe an eine bestimmte Instanz des VIs das Ausgabe-Array **Sinus-Schwingung**, das die nächsten **Abtastwerte** einer Sinuswelle enthält.

Phasenausgang ist auf phase[*n*] eingestellt, und, wenn **Phase zurücksetzen** bei der nächsten Ausführung des VIs FALSE ist, verwendet dieses ablaufinvariante VI diesen Wert als seinen neuen **Phaseneingang**.

Rechteck-Schwingung

Erzeugt ein Array, das eine Rechteckwelle enthält.



Wenn die Folge *Y* die **Rechteck-Schwingung** darstellt, erzeugt das VI ein Muster entsprechend der folgenden Gleichung:

$$y_i = a * \text{square}(\text{phase}[i]), \text{ für } i = 0, 1, 2, ..., n–1,$$

wobei *a* die **Amplitude** ist; *n* die Anzahl der **Abtastwerte** ist;

$$\text{square}(\text{phase}[i]) = \begin{cases} 1{,}0 & 0 \le p < \left(\frac{\text{duty}}{100}360\right) \\ -1{,}0 & \left(\frac{\text{duty}}{100}360\right) \le p < 360 \end{cases},$$

wobei *p* = phase[*i*] modulo 360,0, *duty* = **Tastverhältnis**, phase[*i*] = initial_phase + **f***360,0**i* ist; **f** die Frequenz in normierten Einheiten von Zyklen/Abtastwert ist, initial_phase der **Phaseneingang** ist, wenn **Phase zurücksetzen** TRUE ist, oder initial_phase der **Phasenausgang** von der vorangegangenen Ausführung dieser Instanz des VIs ist, wenn **Phase zurücksetzen** FALSE ist.

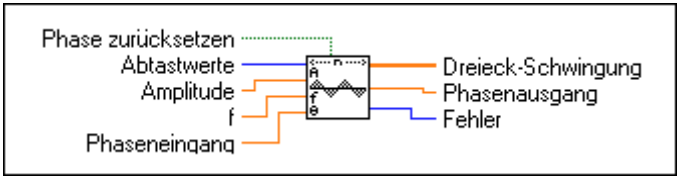
Das VI ist ablaufinvariant, daher kann es zur Simulation einer fortlaufenden Erfassung eines Rechteckwellenfunktionsgenerators eingesetzt werden. Wenn das Bedienelement **Phase**

zurücksetzen FALSE ist, erzeugen nachfolgende Aufrufe an eine bestimmte Instanz des VIs das Ausgabe-Array **Rechteck-Schwingung**, das die nächsten **Abtastwerte** einer Rechteckwelle enthält.

Phasenausgang ist auf phase[*n*] eingestellt, und, wenn **Phase zurücksetzen** bei der nächsten Ausführung des VIs FALSE ist, verwendet dieses ablaufinvariante VI diesen Wert als seinen neuen **Phaseneingang**.

Dreieck-Schwingung

Erzeugt ein Array, das eine Dreieckswelle enthält.



Wenn die Folge *Y* die **Dreieck-Schwingung** darstellt, erzeugt das VI das Muster entsprechend der folgenden Gleichung:

$$y_i = a * \text{tri}(\text{phase}[i]), \text{ für } i = 0, 1, 2, ..., n–1$$

wobei *a* die **Amplitude** ist; *n* die Anzahl der **Abtastwerte** ist;

$$\text{tri}(\text{phase}[i]) = \begin{cases} \frac{p}{90} & 0 \le p < 90 \\ 2 - \frac{p}{90} & 90 \le p < 270 \\ \frac{p}{90} + 4 & 270 \le p < 360 \end{cases}$$

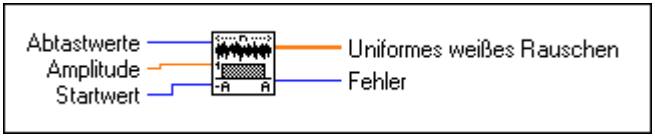
wobei *p* = (phase[*i*] modulo 360,0); phase[*i*] = initial_phase + **f***360,0**i* ist; **f** die Frequenz in normierten Einheiten von Zyklen/Abtastwert ist, initial_phase der **Phaseneingang** ist, wenn **Phase zurücksetzen** TRUE ist, oder initial_phase der **Phasenausgang** von der vorangegangenen Ausführung dieser Instanz des VIs ist, wenn **Phase zurücksetzen** FALSE ist.

Das VI ist ablaufinvariant, daher kann es zur Simulation einer fortlaufenden Erfassung eines Dreieckswellenfunktionsgenerators eingesetzt werden. Wenn das Bedienelement **Phase zurücksetzen** FALSE ist, erzeugen nachfolgende Aufrufe an eine bestimmte Instanz des VIs das Ausgabe-Array **Dreieck-Schwingung**, das die nächsten **Abtastwerte** einer Dreieckswelle enthält.

Phasenausgang ist auf phase[*n*] eingestellt, und, wenn **Phase zurücksetzen** bei der nächsten Ausführung des VIs FALSE ist, verwendet dieses ablaufinvariante VI diesen Wert als seinen neuen **Phaseneingang**.

Uniformes weißes Rauschen

Erzeugt ein gleichförmig verteiltes, pseudozufälliges Muster, dessen Werte sich in dem Bereich [–*a*:*a*] befinden, wobei *a* der absolute Wert von **Amplitude** ist.



Das VI erzeugt eine pseudozufällige Folge, die den Zufallszahl-Generierungsalgorithmus mit sehr langen Zyklen verwendet. Angenommen, daß die Wahrcheinlichkeitsdichtefunktion, *f*(*x*), vom gleichförmig verteilten **Uniformes weißes Rauschen** wie folgt ist:

$$(x) = \begin{cases} \frac{1}{2a} & \text{wenn } -a \le x \le \\ 0 & \text{ansonsten} \end{cases}$$

wobei *a* der absolute Wert der angegebenen **Amplitude** ist, und angenommen, daß die erwarteten Werte, $E\{\bullet\}$, mit der folgenden Gleichung berechnet werden können:

$$E(x) = \int_{-\infty}^{\infty} x(f(x))dx$$

dann sind der erwartete Mittelwert, μ, und die erwartete Standardabweichung, σ, der pseudozufälligen Folge:

$$\mu = E\{x\} = 0,$$

$$\sigma = [E\{(x-\mu)^2\}]^{1/2} = \frac{a}{\sqrt{3}} \approx 0{,}57735a\,.$$

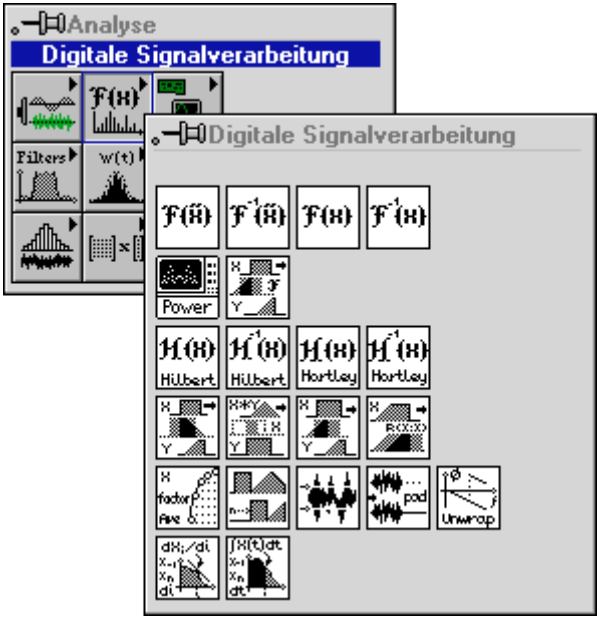
Die pseudozufällige Folge erzeugt ungefähr 2^{90} Abtastwerte, ehe sich das Muster wiederholt.

39

VIs zur digitalen Signalverarbeitung

Dieses Kapitel beschreibt die VIs, die ein erfaßtes oder simuliertes Signal verarbeiten und analysieren. Die VIs Digitale Signalverarbeitung führen Frequenzbereichstransformationen, Frequenzbereichsanalysen, Zeitbereichsanalysen und andere Transformationen wie die Fourier-, Hartley- und Hilbert-Transformationen aus.

Auf die Palette **Digitale Signalverarbeitung** wird über das Menü **Funktionen»Analyse»Digitale Signalverarbeitung** zugegriffen. Die folgende Abbildung zeigt die auf der **Digitale Signalverarbeitung**s-Palette verfügbaren Optionen.



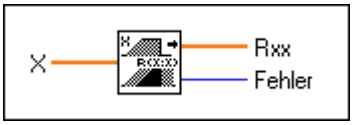
Beispiele für die Verwendung der VIs Digitale Signalverarbeitung finden Sie in `examples\analysis\dspxmpl.llb`.

Beschreibungen der Signalverarbeitungs-VIs

Die folgenden Signalverarbeitungs-VIs sind verfügbar.

Autokorrelation

Berechnet die Autokorrelation der Eingabefolge **X**.



Die Autokorrelation $R_{xx}(t)$ einer Funktion $x(t)$ ist definiert als

$$R_{xx}(t) = x(t) \otimes x(t) = \int_{-\infty}^{\infty} x(\tau)x(t+\tau)dt$$

wobei das Symbol ⊗ die Korrelation bezeichnet.

Bei der diskreten Implementierung dieses VIs sei *Y* eine Folge, dessen Indexierung negativ sein kann, sei *n* die Anzahl der Elemente in der Eingabefolge **X**, und angenommen, daß die indexierten Elemente von **X**, die außerhalb ihres Bereichs liegen, gleich Null sind,

$$x_j = 0, \quad j < 0 \quad \text{oder} \quad j \geq n.$$

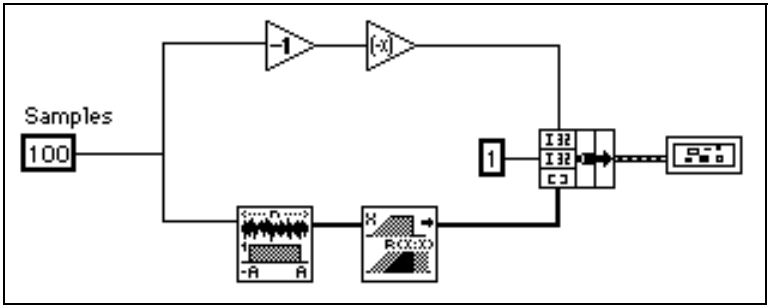
dann erhält das VI die Elemente von *Y* unter Verwendung von

$$y_j = \sum_{k=0}^{n-1} x_k x_{j+k} \quad \text{für } j = -(n-1), -(n-2), \ldots, -2, -1, 0, 1, 2, \ldots, n-1.$$

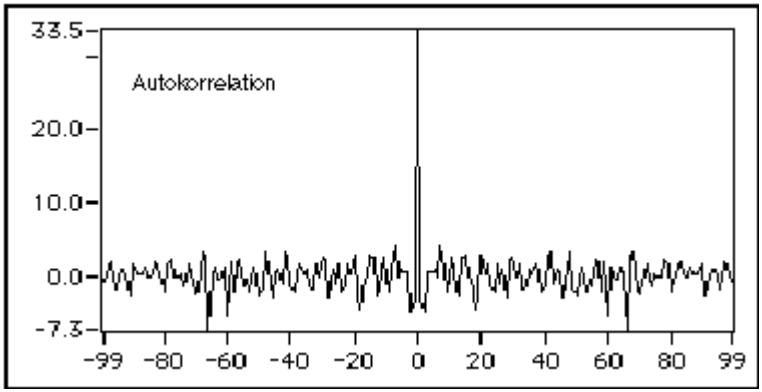
Die Elemente der Ausgangsfolge **Rxx** sind auf die Elemente der Folge *Y* bezogen durch:

$$Rxx_i = y_{i-(n-1)} \text{ für } i = 0, 1, 2, \ldots, 2n-2.$$

Beachten Sie, daß die Anzahl der Elemente in der Ausgangsfolge **Rxx** 2n - 1 ist. Da bei der Indexierung von LabVIEW-Arrays keine negativen Zahlen verwendet werden können, ist der korrespondierende Korrelationswert bei $t = 0$ das *n*. Element der Ausgangsfolge **Rxx**. Somit stellt **Rxx** die Korrelationswerte dar, die das VI *n* Mal in der Indexierung verschoben hat. Das folgende Blockdiagramm zeigt eine Methode zur Anzeige der richtigen Indexierung für die Autokorrelationsfunktion.

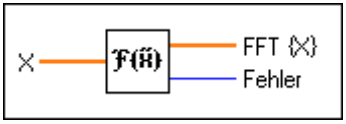


Das folgende Diagramm zeigt das Ergebnis vom vorangegangenen Blockdiagramm.



Komplexe FFT (Fast Fourier-Transformation = schnelle Fouriertransformation)

Berechnet die Fourier-Transformation von der Eingangsfolge **X**.



Dieses VI kann zur Durchführung einer FFT von einem Array aus komplexen, numerischen Darstellungen eingesetzt werden.

Wenn *Y* die komplexe Ausgangsfolge darstellt, dann ist

$$Y = F\{X\}.$$

Daneben kann dieses VI auch zur Ausführung der folgenden Operationen eingesetzt werden, wenn **X** aus einer der komplexen LabVIEW-Datentypen besteht.

- Die FFT einer komplexwertigen Folge *X*
- Die DFT einer komplexwertigen Folge *X*

Dieses VI analysiert zuerst die Eingangsdaten und berechnet anschließend, auf diese Analyse gestützt, die Fourier-Transformation der Daten, indem es eine der vorangegangenen Optionen ausführt. Alle diese Routinen nutzen die Vorteile der verzahnten Verarbeitungsfähigkeiten der CPU und FPU aus.

Wenn die Anzahl der Abtastwerte in der Eingangsfolge **X** eine gültige Potenz von 2 ist, ist

$$n = 2^m \text{ für } m = 1, 2, 3, \ldots, 23,$$

wobei *n* die Anzahl der Abtastwerte ist, berechnet das VI unter Verwendung des Wurzelspaltungsalgorithmus' die schnelle Fourier-Transformation. Die größte komplexe FFT, die das VI berechnen kann, ist 2^{23} = 8.388.608 (8M).

Wenn die Anzahl der Abtastwerte in der Eingangsfolge **X** keine gültige Potenz von 2 ist, ist

$$n \neq 2^m \text{ für } m = 1, 2, 3, \ldots, 23,$$

wobei *n* die Anzahl der Abtastwerte ist, berechnet das VI unter Verwendung des Chirp-z-Algorithmus' die diskrete Fourier-Transformation. Die größte DFT, die berechnet werden kann, ist $2^{22} - 1$ = 4.194.303 (4M – 1).

☞ **Hinweis** ***Da das VI die Transformation vor Ort ausführt, umfassen die Vorteile der FFT Ausführungsgeschwindigkeit und Speicherausnutzung. Die Größe der Eingangsfolge muß allerdings eine Potenz von 2 sein. Die DFT kann eine Folge jeder Größe effektiv verarbeiten, ist jedoch langsamer als die FFT und benötigt mehr Speicherplatz, weil sie während der Verarbeitung mehr Zwischenergebnisse abspeichern muß.***

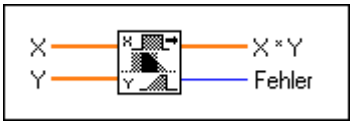
Y sei die komplexe Ausgangsfolge und *n* die Anzahl der enthaltenen Abtastwerte. Es kann bewiesen werden, daß

$$Y_{n-i} = Y_{-i}$$

was bedeutet, daß das (*n* – *i*). Element von *Y* als das – *i*-te. Element der Folge interpretiert werden kann, wenn es physikalisch realisiert werden könnte, was der negativen *i*-ten. Harmonischen entspricht.

Faltung

Berechnet die Faltung der Eingangsfolgen **X** und **Y**.



Die Faltung $h(t)$ der Signale $x(t)$ und $y(t)$ ist definiert als

$$h(t) = x(t)*y(t) = \int_{-\infty}^{\infty} x(\tau)y(t-\tau)d\tau$$

wobei das Symbol $*$ die Faltung bezeichnet.

Bei der diskreten Implementierung der Faltung sei h die Ausgangsfolge **X * Y**, sei n die Anzahl der Elemente in der Eingangsfolge **X** und sei m die Anzahl der Elemente in der Eingangsfolge **Y**. Angenommen, daß die indexierten Elemente **X** und **Y**, die außerhalb ihrer Bereiche liegen, Null sind,

$$x_i = 0, \quad i < 0 \quad \text{oder} \quad i \geq n$$

und

$$y_j = 0, \quad j < 0 \quad \text{oder} \quad j \geq m,$$

dann erhalten Sie die Elemente von h unter Verwendung von

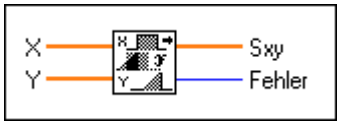
$$h_i = \sum_{k=0}^{n-1} x_k y_{i-k} \quad \text{für } i = 0, 1, 2, \ldots, \text{Größe}-1,$$

Größe = $n + m - 1$,

wobei Größe die Gesamtanzahl der Elemente in der Ausgangsfolge **X * Y** ist.

Kreuzleistungsspektrum

Berechnet das Kreuzleistungsspektrum der Eingangsfolgen **X** und **Y**.



Die Kreuzleistung, $S_{xy}(f)$ der Signale $x(t)$ und $y(t)$ ist definiert als

$$S_{xy}(f) = X^{*}(f)Y(f)$$

wobei $X^{*}(f)$ die konjugiert-komplexe Zahl von $X(f)$, $X(f) = F\{x(t)\}$ und $Y(f) = F\{y(t)\}$ ist.

Dieses VI setzt zur Berechnung des Kreuzleistungsspektrums die FFT- oder DFT-Routine ein, die wie folgt gegeben ist:

$$S_{xy} = \frac{1}{n^2} F^{*}\{X\}F\{Y\},$$

wobei S_{xy} die komplexe Ausgangsfolge **Sxy** darstellt, und n die Anzahl der Abtastwerte ist, die beide Eingangsfolgen **X** und **Y** aufnehmen können.

Die größte Kreuzleistung, die das VI über die FFT berechnen kann, ist 2^{23} (8.388.608 oder 8 Mio.).

Wenn **X** und **Y** aus der gleichen Anzahl von Abtastwerten bestehen und diese eine gültige Potenz von 2 sind,

$$n = m = 2^{k} \text{ für } k = 1, 2, 3, \ldots, 23,$$

wobei n die Anzahl der Abtastwerte in **X** und m die Anzahl der Abtastwerte in **Y** ist, ruft das VI die FFT-Routine zur Berechnung der komplexen Kreuzleistungsfolge direkt auf. Diese Methode ist sowohl für die Ausführungszeit als auch Speicherverwaltung äußerst effektiv, weil das VI die Operationen vor Ort ausführt.

Wenn die Anzahl der Abtastwerte in **X** und **Y** ungleich ist,

$$n \neq m,$$

wobei n die Anzahl der Abtastwerte in **X** und m die Anzahl der Abtastwerte in **Y** ist, gleicht das VI zuerst die Größe der kleineren Folge durch Auffüllen mit Nullen an die der größeren Folge an. Wenn diese Größe eine gültige Potenz von 2 ist,

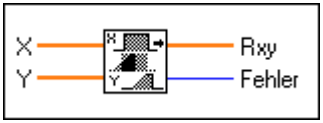
$$\max(n,m) = 2^{k} \text{ für } k = 1, 2, 3, \ldots, 23,$$

berechnet das VI unter Verwendung von FFT das Kreuzleistungsspektrum; ansonsten verwendet das VI zur Berechnung des Kreuzleistungsspektrums die langsamere DFT. Damit ist also die Größe der komplexen Ausgangsfolge

$$\text{Größe} = \max(n,m).$$

Kreuzkorrelation

Berechnet die Kreuzkorrelation der Eingangsfolgen **X** und **Y**.



Die Kreuzkorrelation $R_{xy}(t)$ der Signale $x(t)$ und $y(t)$ ist definiert als

$$R_{xy}(t) = x(t) \otimes y(t) = \int_{-\infty}^{\infty} x(\tau)y(t+\tau)d\tau,$$

wobei das Symbol ⊗ Korrelation bezeichnet.

Bei der diskreten Implementierung dieses VIs sei *h* eine Folge, deren Indexierung negativ sein kann, sei *n* die Anzahl der Elemente in der Eingangsfolge **X**, sei *m* die Anzahl der Elemente in der Folge **Y**, und angenommen, daß die indexierten Elemente von **X** und **Y**, die außerhalb ihrer Bereiche liegen, gleich Null sind,

$$x_j = 0, \quad j < 0 \quad \text{oder} \quad j \geq n$$

und

$$y_j = 0, \quad j < 0 \quad \text{oder} \quad j \geq m,$$

dann erhält das VI die Elemente von *h* unter Verwendung von

$$h_j = \sum_{k=0}^{n-1} x_k y_{j+k} \quad \text{für } j = -(n-1), -(n-2), \ldots, -2, -1, 0, 1, 2, \ldots, m-1.$$

Die Elemente der Ausgangsfolge **Rxy** beziehen sich auf die Elemente in der Folge *h* durch

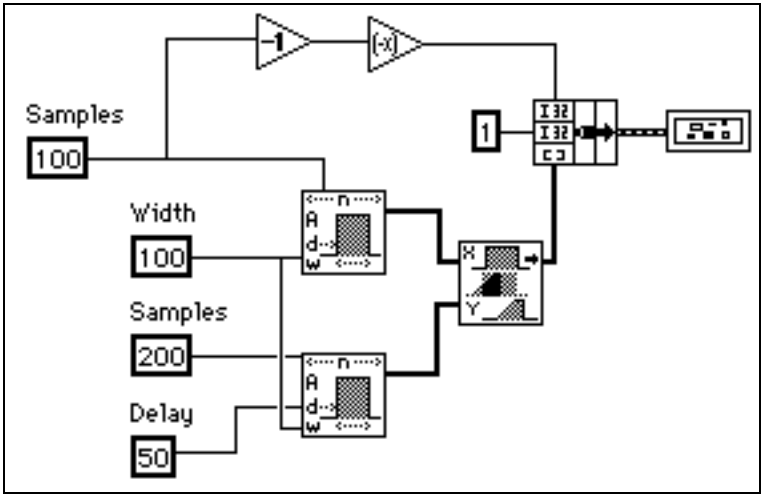
$$Rxy_i = h_{i-(n-1)} \quad \text{für } i = 0, 1, 2, \ldots, \text{Größe}-1,$$

Größe = $n + m - 1$,

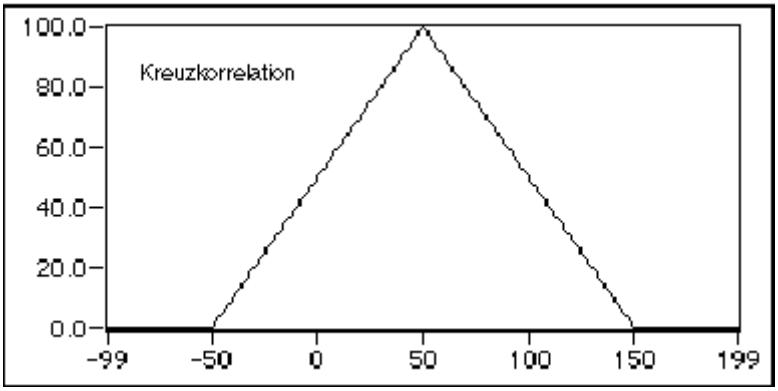
wobei Größe die Anzahl der Elemente in der Ausgangsfolge **Rxy** ist.

Da bei der Indexierung von LabVIEW-Arrays keine negativen Zahlen verwendet werden können, ist der korrespondierende Kreuzkorrelationswert bei $t = 0$ das n. Element der Ausgangsfolge **Rxx**. Somit stellt **Rxx** die Korrelationswerte dar, die das VI n Mal in der Indexierung verschoben hat.

Das folgende Blockdiagramm zeigt eine Methode der Indexierung des Kreuzkorrelations-VI.

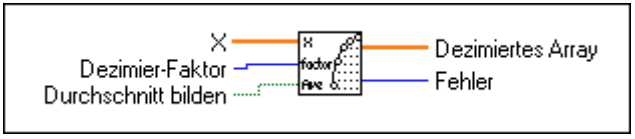


Das folgende Diagramm zeigt das Ergebnis des vorangegangenen Blockdiagramms.



Dezimieren

Dezimiert die Eingangsfolge **X** um den **Dezimier-Faktor** und das binäre Bedienelement **Durchschnitt bilden**.



Wenn *Y* die Ausgangsfolge **Dezimiertes Array** darstellt, erhält das VI die Elemente von der Folge *Y* unter Verwendung von

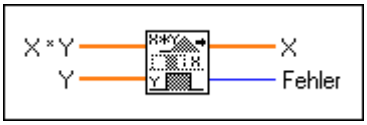
$$y_i = \begin{cases} x_{im} & \text{wenn ave false ist} \\ \frac{1}{m}\sum_{k=0}^{m-1} x_{(im+k)} & \text{wenn ave true ist} \end{cases} \quad \text{für } i = 0, 1, 2, \ldots, \text{Größe}-1$$

$$\text{Größe} = \text{trunc}\left(\frac{n}{m}\right),$$

wobei *n* die Anzahl der Elemente in **X** ist, *m* der **Dezimier-Faktor** ist, *ave* die Option **Durchschnitt bilden** ist und *Größe* die Anzahl der Elemente in der Ausgangsfolge **Dezimiertes Array** ist.

Entfaltung

Berechnet die Entfaltung der Eingangsfolgen **X * Y** und **Y**.



Das VI kann Fourier-Identitäten zur Realisierung der Operation Entfaltung einsetzen, weil

$$x(t) * y(t) \Leftrightarrow X(f)\ Y(f)$$

ein Fourier-Transformationspaar ist, wobei das Symbol $*$ die Entfaltung bezeichnet und die Entfaltung die Umkehrung der Operation Faltung ist. Wenn $h(t)$ das Signal aus der Entfaltung der Signale $x(t)$ und $y(t)$ ist, erhält das VI $h(t)$ unter Verwendung der Gleichung

$$h(t) = F^{-1}\left(\frac{X(f)}{Y(f)}\right),$$

wobei $X(f)$ die Fourier-Transformation von $x(t)$ und $Y(f)$ die Fourier-Transformation von $y(t)$ ist.

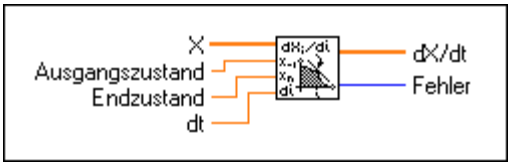
Das VI führt die diskrete Implementierung der Entfaltung unter Verwendung der folgenden Schritte aus:

1. Berechnet die Fourier-Transformation der Eingangsfolge **X * Y**.
2. Berechnet die Fourier-Transformation der Eingangsfolge **Y**.
3. Dividiert die Fourier-Transformation von **X * Y** durch die Fourier-Transformation von **Y**. Benennt die neue Folge mit *H*.
4. Berechnet die umgekehrte Fourier-Transformation von *H*, um die entfaltete Folge von **X** zu erhalten.

☞ **Hinweis** ***Die Entfaltungsoperation ist eine numerische instabile Operation, und es ist nicht immer möglich, das System numerisch zu lösen. Die Berechnung der Entfaltung über die FFTs ist eventuell der stabilste allgemeine Algorithmus, der keine komplizierten DSP-Techniken erfordert. Er ist jedoch nicht fehlerfrei (zum Beispiel, wenn in der Fourier-Transformation der Eingangsfolge Y Nullen enthalten sind).***

Ableitung x(t)

Führt eine diskrete Differenzierung des abgetasteten Signals **X** aus.



Die Differenzierung $f(t)$ einer Funktion $F(t)$ ist definiert als

$$f(t) = \frac{d}{dt}F(t).$$

Y sei die abgetastete Ausgangsfolge **dX/dt**. Die diskrete Implementierung ist gegeben durch

$$y_i = \frac{1}{2dt}(x_{i+1} - x_{i-1}) \text{ für } i = 0, 1, 2, \ldots, n-1,$$

wobei *n* die Anzahl der Abtastwerte in **x(t)** ist,

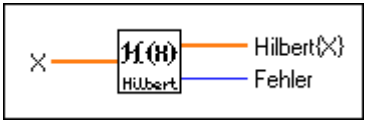
x_{-1} ist angegeben durch den **Ausgangszustand**, wenn $i = 0$ ist, und

x_n ist angegeben durch den **Endzustand**, wenn $i = n-1$ ist.

Der **Ausgangszustand** und **Endzustand** minimiert den Fehler an den Grenzen.

Schnelle-Hilbert-Transformation

Berechnet die schnelle Hilbert-Transformation der Eingangsfolge **X**.



Die Hilbert-Transformation einer Funktion $x(t)$ ist definiert als

$$h(t) = H\{x(t)\} = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x(\tau)}{t-\tau}d\tau.$$

Mit den Fourier-Identitäten kann gezeigt werden, daß die Fourier-Transformation von der Hilbert-Transformation von $x(t)$ wie folgt ist

$$h(t) \Leftrightarrow H(f) = -j \operatorname{sgn}(f)\, X(f),$$

wobei $x(t) \Leftrightarrow X(f)$ ein Fourier-Transformationspaar ist und

$$\operatorname{sgn}(f) = \begin{cases} 1 & f > 0 \\ 0 & f = 0 \\ -1 & f < 0 \end{cases}$$

Das VI vervollständigt die folgenden Schritte bei der Ausführung der diskreten Implementierung der Hilbert-Transformation mit Hilfe der FFT-Routinen, die auf dem Fourier-Transformationspaar $h(t) \Leftrightarrow H(f)$ beruhen (weitere Informationen finden Sie im Ausgabeformat des VIs FFT):

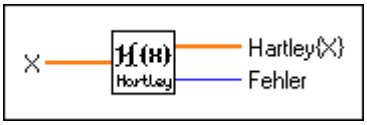
1. Fourier-Transformation der Eingangsfolge **X**: $Y = F\{X\}$.
2. Setzt die DC-Komponente auf Null: $Y_0 = 0$.
3. Wenn die Folge Y eine gerade Größe hat, wird die Nyquist-Komponente auf Null gesetzt: $Y_{\text{Nyq}} = 0$.
4. Multipliziert die positiven Harmonischen mit $-j$.
5. Multipliziert die negativen Harmonischen mit j. Benennt die neue Folge mit H, welche die Form $H_k = -j \operatorname{sgn}(k)\, Y_k$ besitzt.
6. Führt die umgekehrte Fourier-Transformation H aus, um die Hilbert-Transformation von **X** zu erhalten.

Die Hilbert-Transformation wird eingesetzt, um momentane Phaseninformationen zu extrahieren, die Hüllkurve eines Schwingungssignals und, Einzelseitenband-Spektren zu erhalten, Echos aufzufinden und die Abtastrate zu reduzieren.

☞ **Hinweis** ***Weil das VI die DC- und Nyquist-Komponenten auf Null setzt, wenn die Anzahl der Elemente der Eingangsfolge gerade ist, kann das Ursprungssignal nicht immer mit der umgekehrten Hilbert-Transformation wiederhergestellt werden. Die Hilbert-Transformation arbeitet gut mit Signalen mit begrenzter Durchlaßbreite, die die DC- und Nyquist-Komponenten ausschließen.***

FHT

Berechnet die schnelle Hartley-Transformation (FHT) der Eingangsfolge **X**.



Die Hartley-Transformation einer Funktion $x(t)$ ist definiert als

$$X(f) = \int_{-\infty}^{\infty} x(t)\,\mathrm{cas}(2\pi ft)\,dt$$

wobei $\mathrm{cas}(x) = \cos(x) + \sin(x)$.

Wenn *Y* die Ausgangsfolge **Hartley{X}** ist, die über die FHT erhalten wurde, dann wird *Y* durch die diskrete Implementierung des Hartley-Integrals erhalten:

$$Y_k = \sum_{i=0}^{n-1} X_i \mathrm{cas}\left(\frac{2\pi ik}{n}\right), \text{ für } k = 0, 1, 2, \ldots, n-1,$$

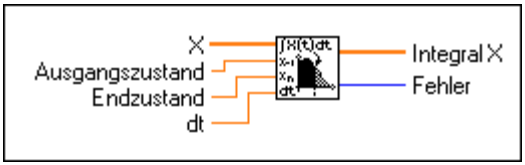
wobei *n* die Anzahl der Elemente in **X** ist.

FHT bildet reellwertige Folgen auf reellwertigen Frequenzbereichsfolgen ab. Sie kann anstelle der Fourier-Transformation zur Faltung, Entfaltung und Korrelationen von Signalen und zum Finden des Leistungspektrums eingesetzt werden. Außerdem kann die Fourier-Transformation von der Hartley-Transformation abgeleitet werden.

Wenn die zu bearbeitenden Folgen reellwertig sind, produziert die Fourier-Transformation komplexwertige Folgen, in denen die Informationen zur Hälfte redundant sind. Der Vorteil in der Verwendung der FHT anstelle der FFT liegt darin, daß die FHT nur halb soviel Speicher zur Erzeugung derselben Informationen benötigt wie die FFT. Außerdem wird die FHT vor Ort berechnet und ist so wirkungsvoll wie die FFT. Der Nachteil der FHT liegt darin, daß die Größe der Eingangsfolge eine gültige Potenz von 2 sein muß.

Integral x(t)

Führt die diskrete Integration des abgetasteten Signals **X** aus.



Das Integral $F(t)$ einer Funktion $f(t)$ ist definiert als

$$F(t) = \int f(t)dt$$

Y sei die abgetastete Ausgangsfolge **Integral X**. Das VI erhält die Elemente von Y durch

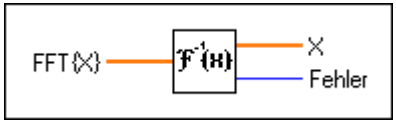
$$y_i = \frac{1}{6}\sum_{j=0}^{i}(x_{j-1} + 4x_j + x_{j+1})dt \quad \text{für } i = 0, 1, 2, \ldots, n-1,$$

wobei n die Anzahl der Elemente in **X** ist, x_{-1} durch **Ausgansgszustand** angegeben ist, wenn $i = 0$ ist, und x_n durch **Endzustand** angegeben ist, wenn $i = n-1$ ist.

Ausgangszustand und **Endzustand** minimiert den Gesamtfehler durch Erhöhen der Genauigkeit an den Grenzen, insbesondere wenn die Anzahl der Abtastwerte gering ist. Das Herausfinden der Grenzbedingungen im voraus verbessert die Genauigkeit.

Inverse komplexe FFT

Berechnet die umgekehrte Fourier-Transformation von der komplexen Eingangsfolge **FFT {X}**.



Dieses VI wird zur Durchführung einer umgekehrten FFT von einem Array aus einem der komplexen numerischen Repräsentierungen von LabVIEW eingesetzt.

Wenn Y die Ausgangsfolge darstellt, dann ist

$$Y = F^{-1}\{X\}.$$

Das VI kann zur Ausführung der folgenden Operationen eingesetzt werden, wenn **FFT {X}** aus einem der komplexen LabVIEW-Datentypen besteht:

- Die inverse FFT einer komplexwertigen Folge *X*
- Die inverse DFT einer komplexwertigen Folge *X*

Dieses FFT-VI analysiert zuerst die Eingangsdaten und berechnet anschließend, auf diese Analyse gestützt, die umgekehrte Fourier-Transformation der Daten, indem es eine der vorangegangenen Optionen ausführt. Alle diese Routinen nutzen die Vorteile der verzahnten Verarbeitungsfähigkeiten der CPU und FPU aus.

Wenn die Anzahl der Abtastwerte in der Eingangsfolge *X* eine gültige Potenz von 2 ist, ist

$n = 2^m$ für $m = 1, 2, 3, \ldots, 23,$

wobei *n* die Anzahl der Abtastwerte ist, berechnet das VI unter Verwendung des Wurzelspaltungs-Algorithmus' die inverse FFT. Die längste Folge mit einer umgekehrten komplexen FFT, die das VI berechnen kann, ist 2^{23}=8.388.608 (8 Mio.).

Wenn die Anzahl der Abtastwerte in der Eingangsfolge *X* keine gültige Potenz von 2 ist, ist

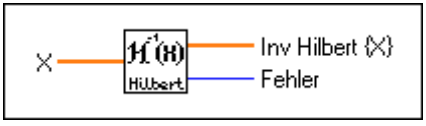
$n \neq 2^m$ für $m = 1, 2, 3, \ldots, 23,$

wobei *n* die Anzahl der Abtastwerte ist, berechnet das VI unter Verwendung des Chirp-Z-Algorithmus' die inverse DFT. Die längste Folge mit der umgekehrten DFT, die das VI berechnen kann, ist $2^{22} - 1$ (4.194.303 oder 4 Mio. – 1).

☞ **Hinweis** ***Da das VI die Transformation vor Ort ausführt, umfassen die Vorteile der FFT Ausführungsgeschwindigkeit und Speicherausnutzung. Die Größe der Eingangsfolge muß allerdings eine Potenz von 2 sein. Die DFT kann eine Folge jeder Größe effektiv verarbeiten, ist jedoch langsamer als die FFT und benötigt mehr Speicherplatz, weil sie während der Verarbeitung Zwischenergebnisse abspeichern muß.***

Inverse Schnelle-Hilbert-Transformation

Berechnet die umgekehrte schnelle Hilbert-Transformation der Eingangsfolge **X**.



Die umgekehrte Hilbert-Transformation einer Funktion $h(t)$ ist definiert als

$$h(t) = H^{-1}\{h(t)\} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{h(\tau)}{t-\tau}d\tau.$$

Mit der Definition der Hilbert-Transformation

$$h(t) = H\{x(t)\} = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x(\tau)}{t-\tau}d\tau$$

kann die umgekehrte Hilbert-Transformation durch Negation der Hilbert-Vorwärts-Transformation erhalten werden

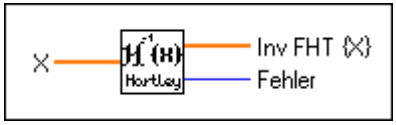
$$x(t) = H^{-1}\{h(t)\} = -H\{h(t)\}.$$

Das VI vervollständigt die folgenden Schritte bei der Ausführung der diskreten Implementierung der umgekehrten Hilbert-Transformation mit Hilfe der Hilbert-Transformation.

1. Führt die Hilbert-Transformation die Eingangsfolge **X** aus: $Y = H\{X\}$.
2. Negiert *Y*, um die umgekehrte Hilbert-Transformation zu erhalten: $H^{-1}\{X\} = -Y$.

Inverse Schnelle-Hartley-Transformation

Berechnet die umgekehrte schnelle Hartley-Transformation (FHT) der Eingangsfolge **X**.



Die umgekehrte Hartley-Transformation einer Funktion *X*(*f*) ist definiert als

$$x(t) = \int_{-\infty}^{\infty} X(f)\,\mathrm{cas}(2\pi ft)\,df,$$

wobei cas(*x*) = cos(*x*) + sin(*x*) ist.

Wenn *Y* die Ausgangsfolge **Inv FHT {X}** darstellt, berechnet das VI die Folge *Y* durch die diskrete Implementierung des umgekehrten Hartley-Integrals:

$$Y_k = \frac{1}{n}\sum_{i=0}^{n-1} X_i \mathrm{cas}\frac{2\pi ik}{n}, \text{ für } k = 0, 1, 2, \ldots, n-1.$$

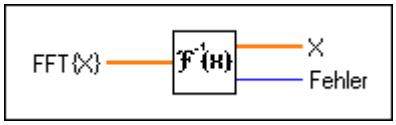
wobei *n* die Anzahl der Elemente in **X** ist.

Die umgekehrte Hartley-Transformation bildet reellwertige Frequenzfolgen auf reellwertigen Folgen ab. Sie kann anstelle der umgekehrten Fourier-Transformation zur Faltung, Entfaltung und Korrelation von Signalen eingesetzt werden. Daneben kann die Fourier-Transformation auch von der Hartley-Transformation abgeleitet werden.

Einen Vergleich der Fourier- und Hartley-Transformationen finden Sie im Abschnitt *FHT* weiter vorn in diesem Kapitel.

Inverse Reale FFT

Berechnet die umgekehrte reale schnelle Fourier-Transformation (FFT) oder die umgekehrte reale diskrete Fourier-Transformation (DFT) der Eingangsfolge **FFT{X}**.



Die Eingangsfolge ist komplexwertig. Dieses VI legt die folgenden Optionen automatisch fest:

- Die Inverse Reale FFT einer komplexwertigen Folge, wenn die Größe eine Potenz von 2 ist.
- Die Inverse Reale DFT einer komplexwertigen Folge, wenn die Größe eine Potenz von 2 ist.

Dieses VI führt die umgekehrten FFT-Routinen aus, wenn die Größe der Eingangsfolge eine gültige Potenz von 2 ist:

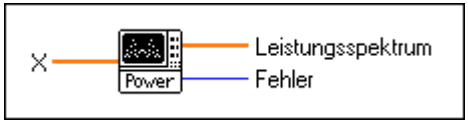
Größe = 2^m, m = 1, 2, ..., 23.

Wenn die Größe der Eingangsfolge keine Potenz von 2 ist, ruft dieses VI eine effektive Inverse-DFT-Routine auf.

Die Ausgangsfolge **X** = Inverse Reale FFT [**FFT{X}**] ist real und wird in einem realen Array ausgegeben.

Leistungsspektrum

Berechnet das Leistungsspektrum der Eingangsfolge **X**.



Das **Leistungsspektrum** $S_{xx}(f)$ einer Funktion $x(t)$ ist definiert als

$$S_{xx}(f) = X^{*}(f)X(f) = |X(f)|^{2},$$

wobei $X(f) = F\{x(t)\}$, und $X^{*}(f)$ die konjugiert komplexe Zahl von $X(f)$ ist.

Dieses VI verwendet zur Berechnung des Leistungsspektrums FFT- und DFT-Routinen, die gegeben sind durch

$$S_{xx} = \frac{1}{n^2}|F\{\mathbf{X}\}|^2,$$

wobei S_{xx} die Ausgangsfolge **Leistungsspektrum** darstellt, und n die Anzahl der Abtastwerte in der Eingangsfolge **X** ist.

Wenn die Anzahl der Abtastwerte, n, in der Eingangsfolge **X** eine gültige Potenz von 2 ist, ist

$n = 2^m$ für m = 1, 2, 3, …, 23,

berechnet dieses VI die FFT einer realwertigen Folge unter Verwendung des Wurzelspaltungs-Algorithmus' und skaliert effektiv das Wertequadrat. Das größte Leistungsspektrum, das das VI unter Verwendung der FFT berechnen kann, ist 2^{23} (8.388.608 oder 8 Mio.).

Wenn die Anzahl der Abtastwerte in der Eingangsfolge X keine gültige Potenz von 2 ist, ist

$n \neq 2^m$ für m = 1, 2, 3,…, 23,

wobei n die Anzahl der Abtastwerte ist, berechnet dieses VI unter Verwendung des Chirp-z-Algorithmus' die diskrete Fourier-Transformation einer realwertigen Folge und skaliert das Wertequadrat. Das größte Leistungsspektrum, das das VI unter Verwendung der schnellen DFT berechnen kann, ist $2^{22} - 1$ (4.194.303 oder 4 Mio. - 1).

Die FFT-Berechnung des Leistungsspektrums ist zeit- und speicher-effektiv, weil die Transformation real ist und vor Ort ausgeführt wird. Die Größe der Eingangsfolge muß jedoch genau eine Potenz von 2 sein. Die DFT-Version berechnet effektiv das Leistungsspektrum von Folgen beliebiger Größe. Die DFT-Version ist langsamer als die FFT-Version, verwendet mehr Speicherplatz und skaliert nicht so effektiv.

Y sei die Fourier-Transformation von der Eingangsfolge **x** und n sei die Anzahl der Abtastwerte darin. Es kann gezeigt werden, daß

$$|Y_{n-i}|^2 = |Y_{-i}|^2.$$

Die Leistung des $(n - i)$. Elements von Y kann als Leistung des $-i$. Elements der Folge interpretiert werden, welche die Leistung in der *i. negativen* Harmonischen darstellt. Die

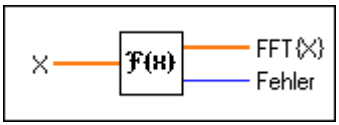
Gesamtleistung der *i.* Harmonischen (DC- und Nyquist-Komponente nicht eingeschlossen) kann gefunden werden unter Verwendung von

$$\text{Leistung der i}^{\text{ten}}\text{ Oberwelle} = 2|Y_i|^2 = |Y_i|^2 + |Y_{n-i}|^2, \quad 0 < i <.$$

Die Gesamtleistung in den DC- und Nyquist-Komponenten ist $|Y_0|^2$ bzw. $|Y_{n/2}|^2$.

Reale FFT

Berechnet die reale schnelle Fourier-Transformation (FFT) oder die reale diskrete Fourier-Transformation (DFT) der Eingangsfolge **X**.



Die Eingangsfolge ist realwertig. Das Reale FFT-VI legt die folgenden Optionen automatisch fest:

- Die FFT einer realwertigen Folge
- Die DFT einer realwertigen Folge

Das VI Reale FFT führt FFT-Routinen aus, wenn die Größe der Eingangsfolge eine gültige Potenz von 2 ist:

Größe = 2^m, m = 1, 2, ..., 23.

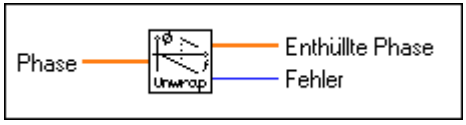
Wenn die Größe der Eingangsfolge keine Potenz von 2 ist, ruft das VI Reale FFT eine effektive Reale DFT-Routine aus.

Die Ausgangsfolge Y = Real FFT[**X**] ist komplex und wird in einem komplexen Array ausgegeben:

$$Y = Y\text{Re} + jY\text{Im}$$

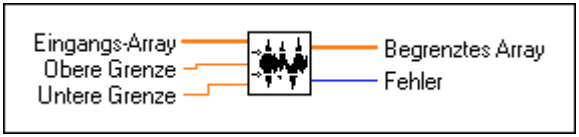
Unwrap Phase

Enthüllt das **Phasen**-Array durch Entfernen von Sprüngen, deren absolute Werte π übersteigen.



Y[i] = Clip {X[i]}

Schneidet die Elemente vom **Eingangs-Array** auf die durch **Obere Grenze** und **Untere Grenze** angegebenen Begrenzungen zurecht.



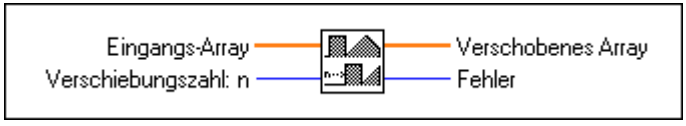
Wenn die Folge *Y* die Ausgangsfolge **Begrenztes Array** darstellt, dann sind die Elemente von *Y* auf die Elemente von **Eingangs-Array** bezogen durch

$$y_i = \begin{cases} a & x_i > a \\ x_i & b \le x_i \le a \quad \text{für } i = 0, 1, 2, \ldots, n-1, \\ b & x_i < b \end{cases}$$

wobei *n* die Anzahl der Elemente in **Eingangs-Array**, *a* die **Obere Grenze** und *b* die **Untere Grenze** ist.

Y[i] = X[i-n]

Verschiebt die Elemente im **Eingangs-Array** um die angegebene Verschiebungszahl.



Wenn die Folge *Y* die Ausgangsfolge **Verschobenes Array** darstellt, dann sind die Elemente von *Y* auf die Elemente von **X** bezogen durch:

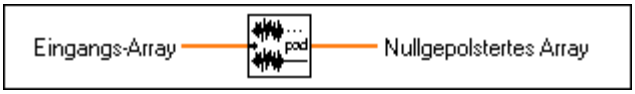
$$y_i = \begin{cases} x_{i-shifts} & \text{wenn } 0 \le i - shifts < n \quad \text{für } i = 0, 1, 2, \ldots, n\text{-}1, \\ 0 & \text{anderweitig} \end{cases}$$

wobei *n* die Anzahl der Elemente in **Eingangs-Array** ist.

☞ **Hinweis** ***Dieses VI rotiert die Elemente im Array nicht. Das VI beseitigt die Elemente der Eingangsfolge, die aus dem Bereich verschoben werden, und sie können nicht durch Verschieben des Arrays in die entgegengesetzte Richtung wiederhergestellt werden.***

Nullpolsterung

Vergrößert die Eingangsfolge **Eingangs-Array** auf die nächsthöhere Potenz von 2, setzt die neuen Schlußelemente der Folge auf Null und beläßt das erste Element, *n*, unverändert, wobei *n* die Anzahl der Abtastwerte in der Eingangsfolge ist.



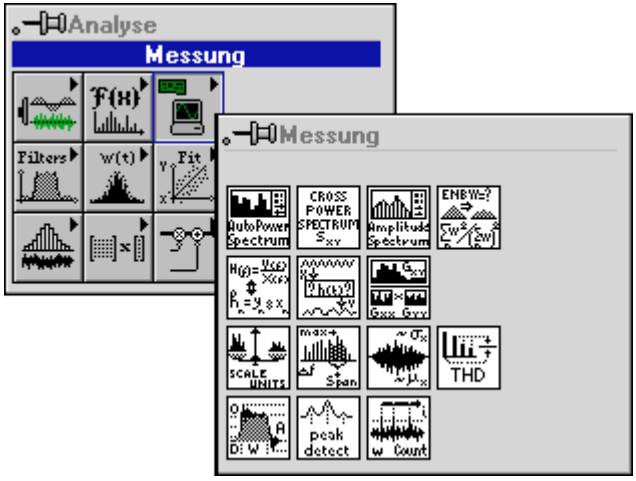
Dieses VI erweist sich als nützlich, wenn die Größe der erfaßten Datenpuffer keine Potenz von 2 ist und der Vorteil der schnellen Verarbeitungsalgorithmen in den Analyse-VIs genutzt werden soll. Diese Algorithmen schließen die Fourier-Transformationen, das Leistungsspektrum und die FHTs ein, die für Puffergrößen von einer Potenz von 2 äußerst effektiv arbeiten.

40

Messungs-VIs

Dieses Kapitel beschreibt die Messungs-VIs, die auf die Ausführung von Analysen mittels DFT und FFT sowie der Signalerfassung von Frequenzmeßanwendungen zugeschnitten sind, wie sie in typischen Frequenzmeßgeräten wie dynamischen Signalanalysatoren gefunden werden.

Auf die **Messung**s-Palette kann über das Menü **Funktionen»Analyse» Messung** zugegriffen werden. Die folgende Abbildung zeigt die auf der Palette **Messung** verfügbaren Optionen.



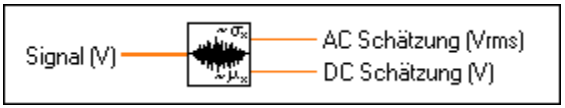
Beispiele der Verwendung der Messungs-VIs finden Sie unter den Beispielen, die die Datenerfassung verwenden, in `examples\analysis\measure\daqmeas.llb`, und die simulierte Signale verwenden, in `examples\analysis\ measure\measxmpl.llb`.

Beschreibungen der Messungs-VIs

Die folgenden Messungs-VIs sind verfügbar.

AC & DC Schätzfunktion

Berechnet eine Schätzung der AC- und DC-Niveaus des Eingangssignals.



Amplituden- und Phasenspektrum

Berechnet die Größe und die Phase des einseitigen, skalierten Amplitudenspektrums eines Signals im Echtzeitbereich.

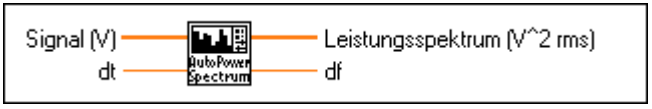
Das VI berechnet das Amplitudenspektrum als

$$\frac{\text{FFT(Signal)}}{N},$$

wobei N die Anzahl der Punkte im **Signal**-Array ist. Danach wandelt das VI das Amplitudenspektrum in einseitige rms-Größen- und Phasenspektren um.

Einseitiges Leistungsspektrum

Berechnet das einseitige, skalierte Leistungsspektrum eines Zeitbereichssignals.



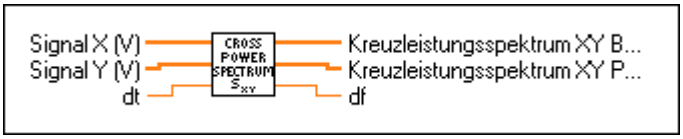
Dieses VI berechnet das Leistungsspektrum als

$$\frac{\text{FFT*(Signal) x FFT(Signal)}}{N^2},$$

wobei *N* die Anzahl der Punkte im **Signal**-Array ist und * für die konjugiert komplexe Zahl steht. Danach konvertiert das VI das Leistungsspektrum in einen Wert des einseitigen Leistungsspektrums.

Kreuzleistungsspektrum

Berechnet das einseitige, skalierte Leistungsspektrum von zwei Echtzeitsignalen. Das Kreuzleistungsspektrum liefert das Produkt der Amplituden der Signale X und Y sowie die Differenz ihrer Phasen (Phase von Y minus Phase von X).



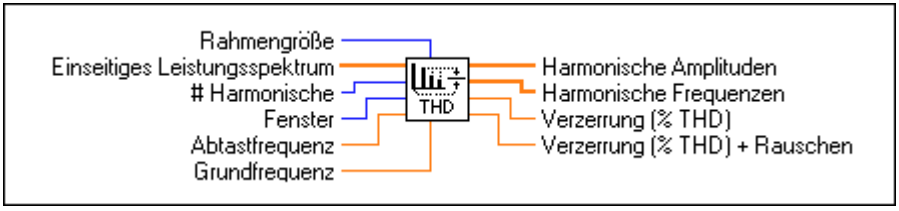
Dieses VI berechnet das Kreuzleistungsspektrum als

$$\frac{\text{FFT*(Signal X) x FFT(Signal Y)}}{N^2},$$

wobei *N* die Anzahl der Punkte in den **Signal X**- oder **Signal Y**-Arrays ist. Danach konvertiert das VI das Kreuzleistungsspektrum in einseitige Größen- und Phasenspektren.

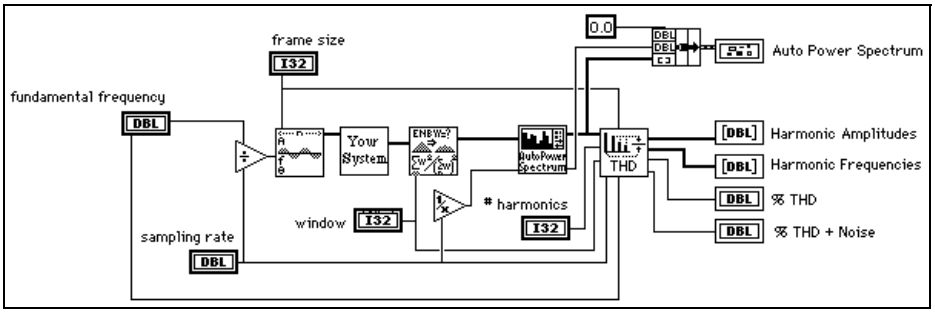
Spektrumanalysator

Findet die Grund- und Oberwellenkomponenten (Amplitude und Frequenz), die im Eingang **Einseitiges Leistungsspektrum** vorhanden sind, und berechnet den Anteil der gesamten Verzerrung der Oberwellen (**% THD**) und der gesamten Verzerrung der Oberwellen zzgl. Rauschen (**%THD + Noise)**.



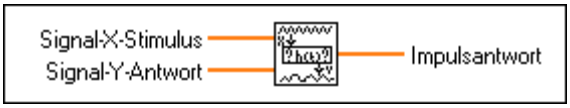
Damit dieses VI ordnungsgemäß funktioniert, muß das gefensterte einseitige Leistungsspektrum des Signals zum VI geleitet werden. Das Zeitbereich signal sollte durch das skalierte Zeitbereichsfenster und anschließend durch das einseitige Leistungsspektrum gegeben werden, wobei der Ausgang des einseitigen Leistungsspektrums mit diesem VI verbunden ist.

Die folgende Abbildung zeigt ein Beispiel für den Einsatz dieses VIs.



Impulsantwort-Funktion

Berechnet die Impulsantwort eines Netzwerks, das auf den realen Signalen X (**Signal-X-Stimulus**) und Y (**Signal-Y-Antwort**) beruht.



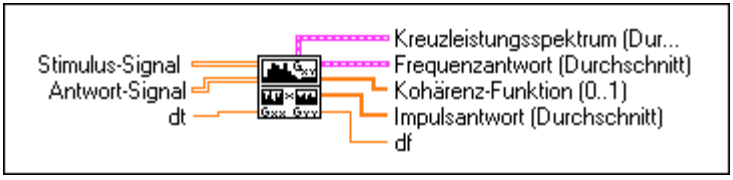
Die **Impulsantwort** liegt im Zeitbereich, so daß Zeiteinheiten nicht in Frequenzeinheiten umgewandelt werden müssen. Die **Impulsantwort** stellt die Umkehrtransformation der Übertragungsfunktion dar.

Dieses VI berechnet die **Impulsantwort** als

$$\text{Inverse FFT}\left[\frac{\text{Kreuzleistung(Stimulus, Antwort)}}{\text{Leistungsspektrum(Stimulus)}}\right].$$

Netzwerkfunktionen (avg)

Berechnet mehrere Netzwerkantwortfunktionen von zwei Echtzeitbereichssignalen X (**Stimulus-Signal**) und Y (**Antwort-Signal**).



Die Signale X (**Stimulus-Signal**) und Y (**Antwort-Signal**) umfassen die Funktionen Kohärenz, durchschnittliche Größe und Phase des Kreuzleistunspektrums, durchschnittliche Übertragungsfunktion (**Frequenzantwort**) und durchschnittliche **Impulsantwort**.

Diese Funktionen werden gewöhnlich von den Stimulus- und Antwortsignalen eines Netzwerkes, das getestet wird, berechnet. Die Kohärenz-Funktion zeigt den Frequenzinhalt von **Antwort-Signal** Y entsprechend **Stimulus-Signal** X an und mißt die Zuverlässigkeit der Frequenzantwort-Messung des Netzwerks.

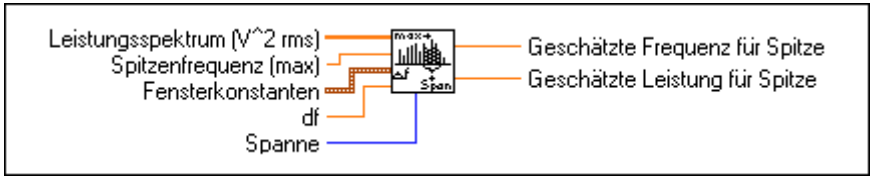
Dieses VI kann zur Messung der Kohärenz von zwei beliebigen Signalen eingesetzt werden. Das VI bildet den Durchschnitt mehrfacher Stimulus- und Antwortsignale, um gültige Kohärenzmessungen zu erhalten. **Kreuzleistungsspektrum** und **Impulsantwort** sind die rms-Durchschnittsversionen von VIs mit ähnlichen Namen. **Frequenzantwort** ist die rms-Durchschnittsversion der Frequenzantwortausgänge des VIs Übertragungsfunktion.

Peak-Detektor

Information über dieses VI finden Sie in diesem Handbuch in Kapitel 47, *Zusätzliche Numerische Methoden-VIs*.

Leistungs- & Frequenz-Schätzung

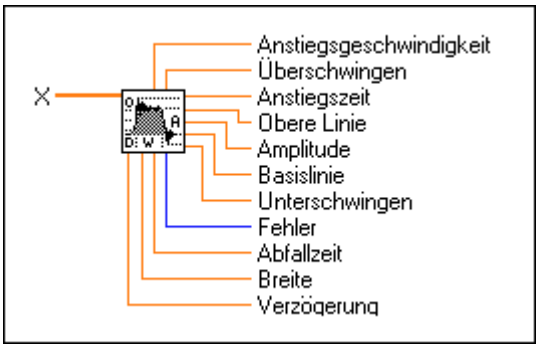
Berechnet die geschätzte Leistung und Frequenz um eine Spitze im Leistungsspektrum eines Zeitbereichssignals herum.



Mit diesem VI können gute Frequenzschätzungen für gemessene Frequenzen, die zwischen den Frequenzlinien in dem Spektrum liegen, erzielt werden. Das VI nimmt Korrekturen für die verwendete Fensterfunktion vor.

Impuls-Parameter

Analysiert die Eingangsfolge **X** nach einem Impulsmuster und bestimmt den besten Satz an Impulsparametern zur Beschreibung des Impulses.



Die mit der Wellenform verbundenen Parameter sind **Anstiegsgeschwindigkeit**, **Überschwingen**, oberste Linie (**Oben**), **Amplitude**, Basislinie (**Basis**) und **Unterschwingen**. Die mit Zeit verbundenen Parameter sind **Anstiegszeit**, **Abfallzeit**, **Breite** (Dauer) und **Verzögerung**.

Dieses VI führt die folgenden Schritte zur Berechnung der Ausgabeparameter aus:

1. Findet die Höchst- und Kleinstwerte in der Eingangsfolge **X**.
2. Erzeugt das Histogramm des Impulses mit einer 1%igen Bereichsauflösung.
3. Legt den oberen und unteren Modus fest, um die Werte von **Oben** und **Basis** einzurichten.
4. Findet **Überschwingen**, **Amplitude** und **Unterschwingen** von den Werten **Oben**, **Basis**, Maximum und Minimum.
5. Sucht **X** und bestimmt die **Anstiegsgeschwindigkeit**, **Anstiegszeit**, **Abfallzeit**, **Breite** und **Verzögerung.**

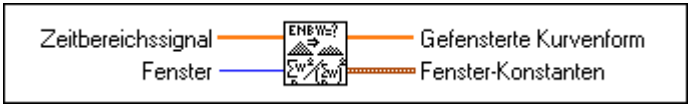
Das VI interpoliert **Breite** und **Verzögerung**, um genauere Ergebnisse nicht nur von **Breite** und **Verzögerung**, sondern auch von **Anstiegsgeschwindigkeit**, **Anstiegszeit** und **Abfallzeit** zu erhalten.

Wenn **X** eine Impulsfolge enthält, verwendet das VI die Folge, um **Überschwingen**, **Oben**, **Amplitude**, **Basis** und **Unterschwingen** zu bestimmen, verwendet allerdings nur den ersten Impuls in der Folge, um die **Anstiegsgeschwindigkeit**, **Anstiegszeit**, **Abfallzeit**, **Breite** und **Verzögerung** festzulegen.

Hinweis ***Weil Impulse gemeinhin in der negativen Richtung auftreten, kann dieses VI zwischen positiven und negativen Impulsen unterscheiden und die Folge X korrekt analysieren. Die Folge braucht vor der Analyse nicht bearbeitet zu werden.***

Skaliertes Zeitbereichsfenster

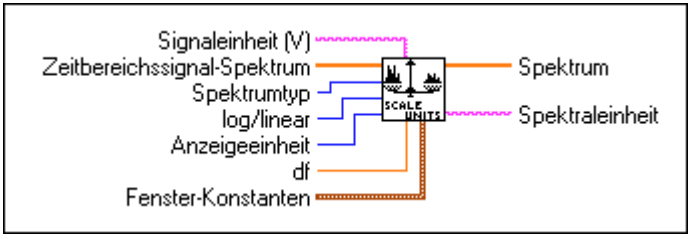
Wendet das ausgewählte Fenster auf das Zeitbereichssignal an.



Das VI skaliert die Ergebnisse, so daß alle Fenster bei der Berechnung des Leistungs- oder Amplitudenspektrums von **Gefensterte Kurvenform** denselben Maßstab innerhalb der Genauigkeitsbeschränkungen des Fensters bereitstellen. Dieses VI gibt auch die für das ausgewählte Fenster wichtigen **Fenster-Konstanten** aus. Diese Konstanten sind nützlich, wenn VIs eingesetzt werden, die Berechnungen vom Leistungsspektrum ausführen, wie die VIs Leistungs- & Frequenzschätzung und Spektrumeinheitenumwandlung.

Spektrumeinheitenumwandlung

Konvertiert entweder das Leistungs-, Amplituden- oder Verstärkungs- (Amplitudenverhältnis-) -spektrum in alternative Formate, einschließlich Log (Dezibel und dbm) und Spektraldichte.

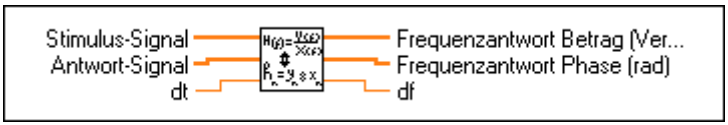


Schwellwert-Peak-Detektor

Informationen über dieses VI finden Sie im Kapitel 47, *Zusätzliche Numerische Methoden-VIs*.

Übertragungsfunktion

Berechnet die Übertragungsfunktion (auch als Frequenzantwort bekannt) aus dem Zeitbereich- **Stimulus-Signal** und -**Antwort-Signal** von einem Netzwerk, das getestet wird.



Dieses VI berechnet die Übertragungsfunktion eines Systems auf der Grundlage der realen Signale X (**Stimulus-Signal**) und Y (**Antwort-Signal**). Die Ausgabe ist die Amplitudenverstärkung des Netzwerks, die ohne Einheit ist.

Die VI-Computer-Frequenzantwort ist

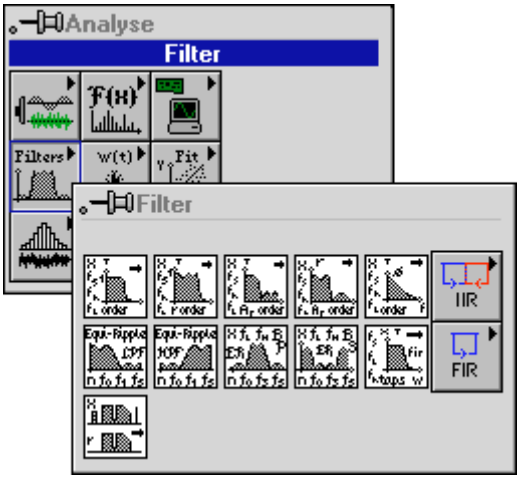
$$\frac{\text{Kreuzleistung(Stimulus, Antwort)}}{\text{Leistungsspektrum(Stimulus)}}.$$

41

Filter-VIs

Dieses Kapitel beschreibt die VIs, die IIR, FIR und nichtlineare Filter implementieren.

Auf die **Filter**-Palette kann über das Menü **Funktionen»Analyse»Filter** zugegriffen werden. Die folgende Abbildung zeigt die auf der **Filter**-Palette verfügbaren Optionen.



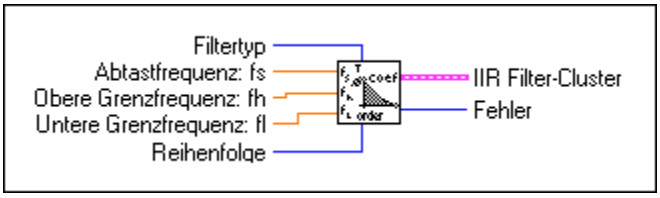
Beispiele für die Verwendung der Filter-VIs finden Sie in `examples\analysis\fltrxmpl.llb`.

Beschreibungen der Filter-VIs

Die folgenden Filter-VIs sind verfügbar.

Bessel-Koeffizienten

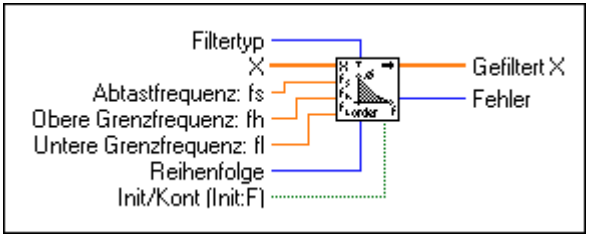
Erzeugt zur Realisierung eines IIR-Filters den Satz Filterkoeffizienten, wie durch das Bessel-Filter-Modell vorgegeben. Diese Koeffizienten können anschließend an das VI IIR-Kaskadenfilter weitergeleitet werden.



Das VI Bessel-Koeffizienten ist ein SubVI vom VI Bessel-Filter.

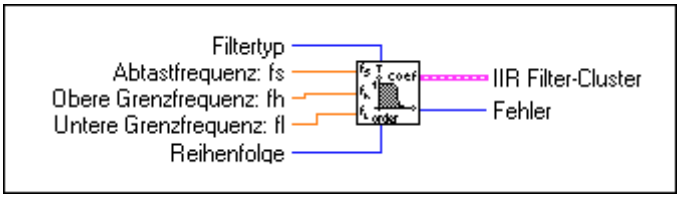
Bessel-Filter

Erzeugt ein digitales Bessel-Filter, wobei es die Koeffizienten **Filtertyp**, **Abtastfrequenz: fs**, **Obere Frequenzgrenze: fh**, **Untere Frequenzgrenze: fl** und **Reihenfolge** verwendet, indem es das VI Bessel-Koeffizienten aufruft. Anschließend ruft das VI das VI IIR-Kaskadenfilter auf, um die Sequenz **X** unter Verwendung dieses Modells in eine Sequenz Bessel-**Gefiltert X** zu filtern.



Butterworth-Koeffizienten

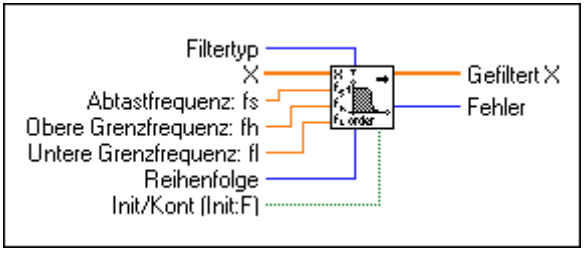
Erzeugt zur Realisierung eines IIR-Filters den Satz Filterkoeffizienten, wie durch das Butterworth-Modell vorgegeben. Diese Filterkoeffizienten (**IIR-Filter-Cluster**) können anschließend an das **VI IIR-Kaskadenfilter** zur Filterung einer Datensequenz weitergeleitet werden.



Dieses VI ist ein SubVI des VIs Butterworth-Filter.

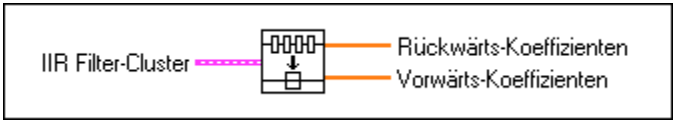
Butterworth-Filter

Erzeugt ein digitales Butterworth-Filter, wobei es die Koeffizienten **Abtastfrequenz: fs**, **Untere Frequenzgrenze: fl**, **Obere Frequenzgrenze: fh**, **Reihenfolge** und **Filtertyp** verwendet, indem es das VI Butterworth-Koeffizienten aufruft. Das VI Butterworth-Filter ruft anschließend das VI IIR-Kaskadenfilter auf, um die Sequenz **X** unter Verwendung dieses Modells in eine Sequenz Butterworth-**Gefiltert X** zu filtern.



Kaskadierung⟶Direkte Koeffizienten

Wandelt die IIR-Filterkoeffizienten aus der Kaskadenform in die direkte Form um.



Zum Beispiel kann ein Kaskadenfilter, der aus zwei Stufen zweiter Ordnung besteht, wie folgt in ein Filter der direkten Form konvertiert werden:

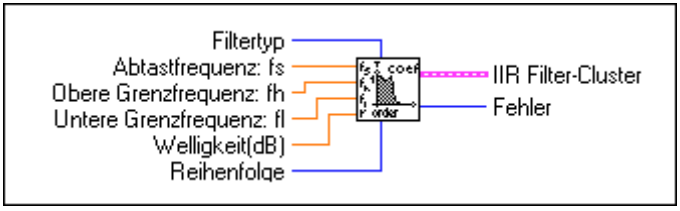
Rückwärts-Koeffizienten: $\{a_{11}, a_{21}, a_{12}, a_{22}\} \rightarrow \{1{,}0, a_1, a_2, a_3, a_4\}$

Vorwärts-Koeffizienten: $\{b_{01}, b_{11}, b_{21}, b_{02}, b_{12}, b_{22}\} \rightarrow \{b_0, b_1, b_2, b_3, b_4\}$

Für Informationen über die Kaskadenform-Filterung sehen Sie bitte unter dem IIR-Kaskadenfilter-VI und für Informationen über die Direktform-filterung unter dem IIR-Filter-VI nach.

Chebyshev-Koeffizienten

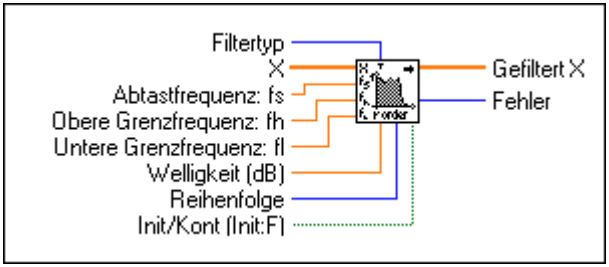
Erzeugt zur Realisierung eines wie im Chebyshev-Filtermodell vorgegebenen IIR-Filters den Satz Filterkoeffizienten. Diese Koeffizienten können zur Filterung einer Datensequenz an das VI IIR-Cluster-Filter weitergegeben werden.



Das VI Chebyshev-Koeffizienten ist ein SubVI der VIs Chebyshev-Filter.

Chebyshev-Filter

Erzeugt ein digitales Chebyshev-Filter, wobei es die Koeffizienten **Abtastfrequenz: fs**, **Untere Frequenzgrenze: fl**, **Obere Frequenzgrenze: fh**, **Welligkeit**, **Reihenfolge** und **Filtertyp** verwendet, indem es das VI Chebyshev-Koeffizienten aufruft. Das VI Chebyshev-Filter filtert anschließend unter Verwendung dieses Modells die Sequenz **X** in eine Chebyshev-**Gefiltert X**, indem es das VI IIR-Kaskadenfilter aufruft.

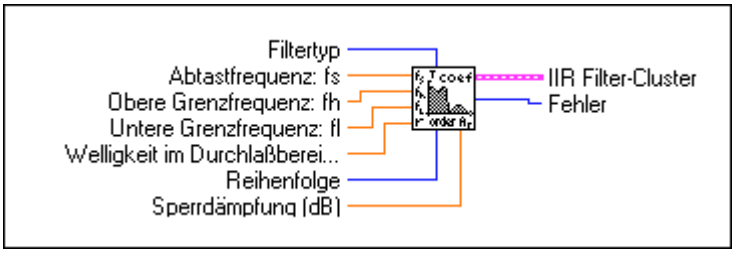


Faltung

Informationen über die Faltung finden Sie in Kapitel 39, *VIs zur digitalen Signalverarbeitung*.

Cauer-Koeffizienten

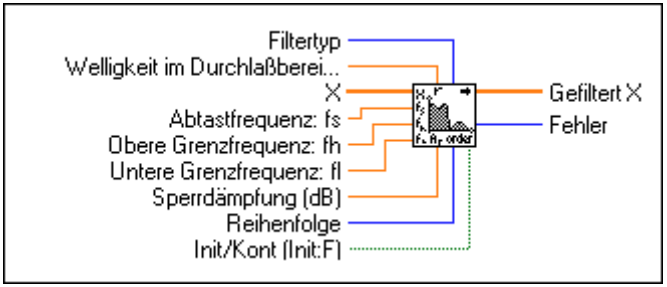
Erzeugt den Satz Filterkoeffizienten zur Realisierung eines digitalen IIR-Cauer-Filter. Diese Koeffizienten können an das VI IIR Kaskadenfilter weitergleitet werden.



Das VI Cauer-Koeffizienten ist ein SubVI des VIs Cauer-Filter.

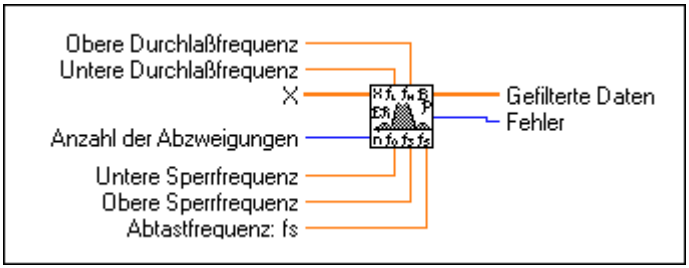
Cauer-Filter

Erzeugt ein digitales Cauer-Filter, wobei es die Koeffizienten **Abtastfrequenz: fs**, **Untere Frequenzgrenze: fl**, **Obere Frequenzgrenze: fh**, **Filtertyp**, **Welligkeit im Durchlaßbereich**, **Sperrdämpfung** und **Reihenfolge** verwendet, indem es das VI Cauer-Koeffizienten aufruft. Das VI Cauer-Filter ruft anschließend das VI IIR-Filter auf, um die Sequenz **X** unter Verwendung dieses Modells in eine Sequenz Cauer-**Gefiltert X** zu filtern.



Equi-Ripple Bandpaß

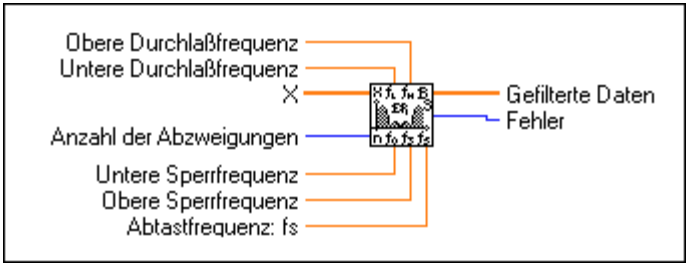
Erzeugt ein FIR-Durchlaßfilter mit Eigenschaften gleicher Welligkeit, indem es den Parks-McClellan-Algorithmus und **Obere Durchlaßfrequenz**, **Untere Durchlaßfrequenz**, **Anzahl der Abzweigungen**, **Untere Sperrfrequenz, Obere Sperrfrequenz** und **Abtastfrequenz: fs** verwendet. Das VI filtert dann die Eingangssequenz **X**, um die gefilterte, lineare Phasensequenz **Gefilterte Daten** des Durchlaßbereichs zu gewinnen.



Der erste Bandsperrbereich des Filters erstreckt sich von Null (DC = Gleichstrom) bis zu **Untere Sperrfrequenz**. Der Bandpaßbereich erstreckt sich von **Untere Durchlaßfrequenz** bis **Obere Durchlaßfrequenz**, und der zweite Bandsperrbereich erstreckt sich von **Obere Sperrfrequenz** bis zur Nyquist-Frequenz.

Equi-Ripple Bandsperre

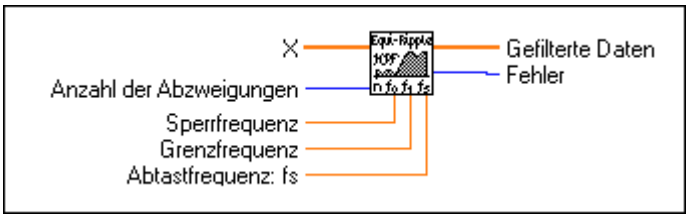
Erzeugt ein digitales FIR-Bandsperrfilter mit Eigenschaften gleicher Welligkeit, unter Verwendung des Parks-McClellan-Algorithmus' und **Obere Durchlaßfrequenz, Untere Durchlaßfrequenz**, **Anzahl der Abzweigungen**, **Untere Sperrfrequenz**, **Obere Sperrfrequenz** und **Abtastfrequenz: fs.** Danach filtert das VI die Eingangssequenz **X**, um die gefilterte, lineare Phasensequenz **Gefilterte Daten** des Sperrbereichs zu gewinnen.



Der erste Bandpaßbereich des Filters erstreckt sich von Null (DC = Gleichstrom) bis zu **Untere Durchlaßfrequenz**. Der Sperrbereich erstreckt sich von **Untere Sperrfrequenz** bis **Obere Sperrfrequenz**, und der zweite Bandpaßbereich erstreckt sich von **Obere Durchlaßfrequenz** bis zur Nyquist-Frequenz.

Equi-Ripple Hochpaß

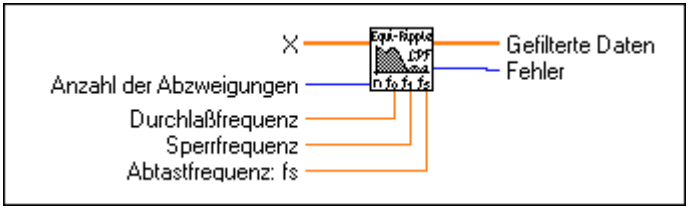
Erzeugt ein FIR-Hochpaßfilter mit Eigenschaften gleicher Welligkeit, unter Verwendung des Parks-McClellan-Algorithmus' und **Anzahl der Abzweigungen**, **Sperrfrequenz**, **Grenzfrequenz** und **Abtastfrequenz**. Danach filtert das VI die Eingangssequenz **X**, um die gefilterte, lineare Phasensequenz **Gefilterte Daten** des Grenzbereichs zu gewinnen.



Der Bandsperrbereich des Filters erstreckt sich von Null (DC = Gleichstrom) bis **Sperrfrequenz**. Das Übergangsband erstreckt sich von **Sperrfrequenz** bis **Grenzfrequenz**, und der Bandpaßbereich erstreckt sich von **Grenzfrequenz** bis zur Nyquist-Frequenz.

Equi-Ripple Tiefpaß

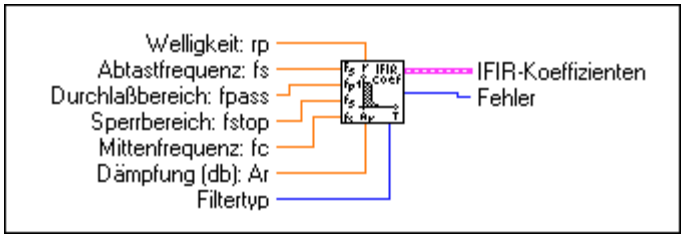
Erzeugt ein FIR-Tiefpaßfilter mit Eigenschaften gleicher Welligkeit, unter Verwendung des Parks-McClellan-Algorithmus' und **Anzahl der Abzweigungen**, **Durchlaßfrequenz**, **Sperrfrequenz** und **Grenzfrequenz**. Danach filtert das VI die Eingangssequenz **X**, um die gefilterte, lineare Phasensequenz **Gefilterte Daten** des Tiefpaßbereichs zu gewinnen.



Der Bandpaßbereich des Filters erstreckt sich von Null (DC) bis **Durchlaßfrequenz**. Das Übergangsband erstreckt sich von **Durchlaßfrequenz** bis **Sperrfrequenz**, und das Sperrband erstreckt sich von der **Sperrfrequenz** bis zur Nyquist-Frequenz.

FIR Schmalband-Koeffizienten

Erzeugt zur Realisierung eines digitalen, interpolierten FIR-Filters einen Satz Filterkoeffizienten. Diese Koeffizienten können an das VI FIR Schmalband-Filter zur Filterung der Daten weitergegeben werden.



Die folgenden Abbildungen zeigen, wie die Schmalband-Filter-Parameter die Tiefpaß-, Hochpaß-, Durchlaß- und Bandsperr-Filter definieren. Die Durchlaßwelligkeit wird mit S_P angegeben. Die Filterantwort ist auf der Y-Achse mit einem linearen Maßstab dargestellt. Aus diesem Grund wurde der Sperrdämpfung A_r mit den folgenden Gleichungen eine lineare Dämpfung zugeordnet:

$$A_r = -20\log\delta_A$$

$$\delta_A = 10^{\frac{-Ar}{20}}.$$

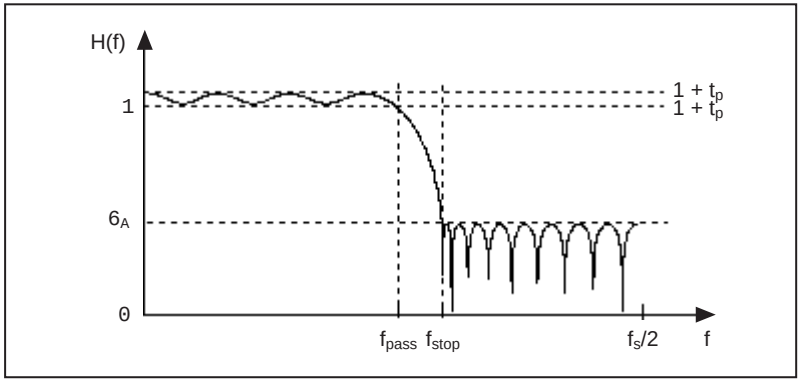


Abbildung 41-1. Tiefpaß-Filter

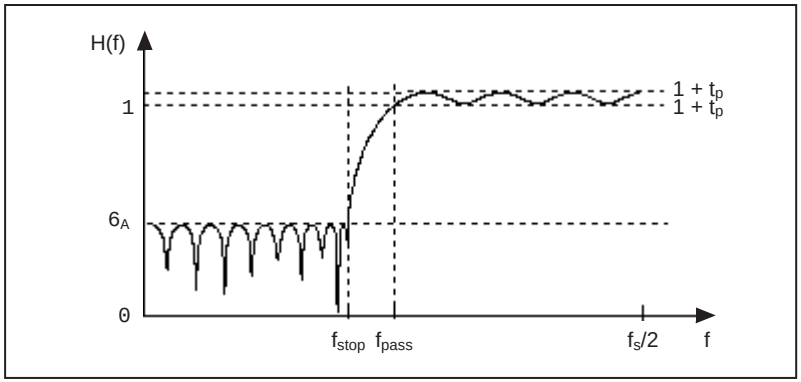


Abbildung 41-2. Hochpaß-Filter

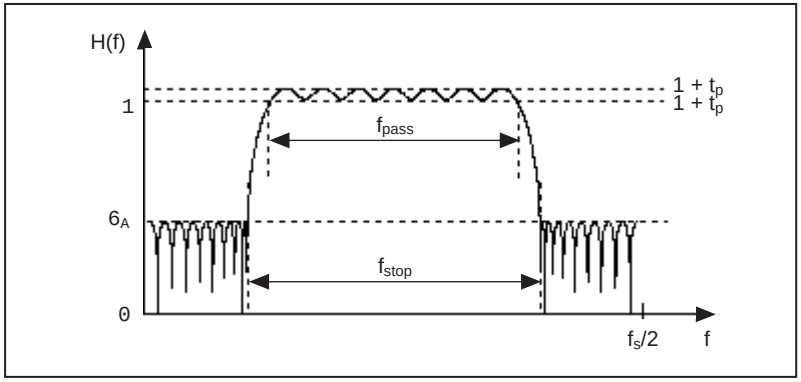


Abbildung 41-3. Durchlaß-Filter

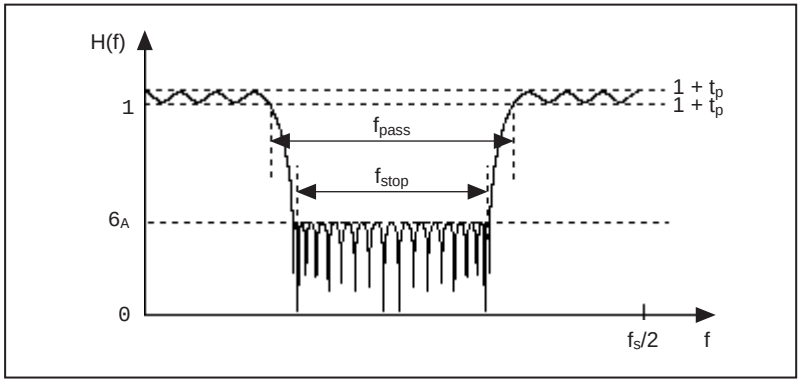
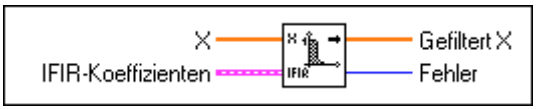


Abbildung 41-4. Sperr-Filter

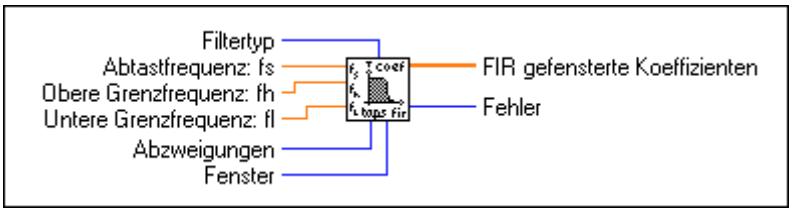
FIR Schmalband-Filter

Filtert die Eingangssequenz **X** mit dem IFIR-Filter entsprechend der **IFIR-Koeffizienten**, die von dem VI FIR Schmalband-Filter-Koeffizienten erstellt wurden.



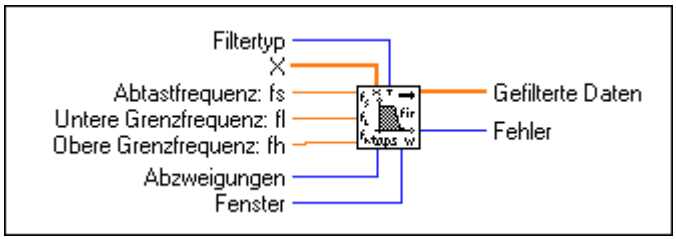
FIR Gefensterte Koeffizienten

Erzeugt den zur Realisierung eines FIR gefensterten Filters benötigten Satz Filterkoeffizienten.



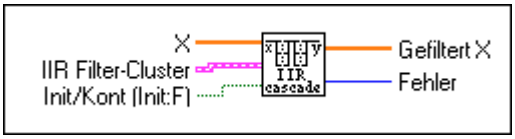
FIR Fenster-Filter

Filtert die Eingangsdatenssequenz **X** unter Verwendung des Satzes FIR-Filterkoeffizienten aus **Abtastfrequenz: fs**, **Untere Grenzfrequenz: fl**, **Obere Grenzfrequenz: fh** und Anzahl der **Abzweigungen**.

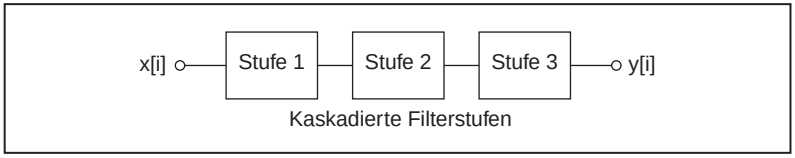


IIR Kaskaden-Filter

Filtert die Eingangssequenz **X** unter Verwendung der Kaskadenform des IIR-Filters, der durch den **IIR Filter-Cluster** bestimmt ist.



Diese IIR-Realisierung wird als Kaskade bezeichnet, weil es eine Kaskade der Filterstufen zweiter oder vierter Ordnung bildet. Die Ausgabe einer Filterstufe ist die Eingabe der nächsten Filterstufe für alle N_s Filterstufen.



Filterung 2. Ordnung

Jede Stufe zweiter Ordnung (Stufenanzahl k = 1, 2, ..., N_s) verfügt über zwei Rückwärts-Koeffizienten (a_{1k}, a_{2k}) und drei Vorwärts-Koeffizienten (b_{0k}, b_{1k}, b_{2k}). Die Gesamtanzahl der Rückwärts-koeffizienten beträgt $2N_s$ und die der Vorwärts-Koeffizienten $3N_s$. Das Array aus den Rückwärts- und Vorwärts-Koeffizienten enthält die Koeffizienten für eine Stufe, gefolgt von den Koeffizienten der nächsten Stufe, usw. Zum Beispiel muß ein IIR-Filter, das aus Stufen zweiter Ordnung zusammengesetzt ist, insgesamt 4 Rückwärts- und 6 Vorwärts-Koeffizienten wie folgt aufweisen:

Rückwärts-Koeffizienten = $\{a_{11}, a_{21}, a_{12}, a_{22}\}$

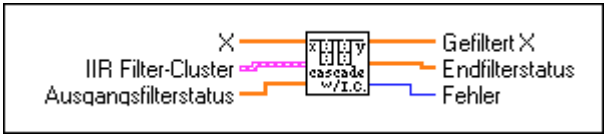
Vorwärts-Koeffizienten = $\{b_{01}, b_{11}, b_{21}, b_{02}, b_{12}, b_{22}\}$

Filterung 4. Ordnung

Bei Kaskadenstufen vierter Ordnung ist die Filterung in derselben Weise wie in der Filterung zweiter Ordnung realisiert, wobei jedoch jede Stufe über vier Rückwärts-Koeffizienten (a_{1k}, ..., a_{4k}) und fünf Vorwärts-Koeffizienten ($b_{0k,}$..., b_{4k}) verfügen muß.

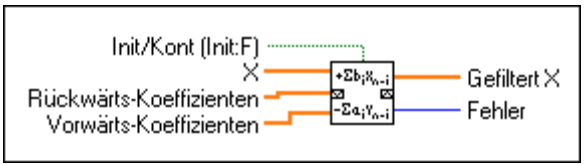
IIR Kaskaden-Filter mit IC

Filtert die Eingangssequenz **X** unter Verwendung der Kaskadenform des IIR-Filters, wie es durch den **IIR Filter Cluster** festgelegt ist.



IIR Filter

Filtert die Eingangssequenz **X** unter Verwendung des IIR-Filters mit direkter Form, wie es durch die **Rückwärts-Koeffizienten** und **Vorwärts-Koeffizienten** festgelegt ist.



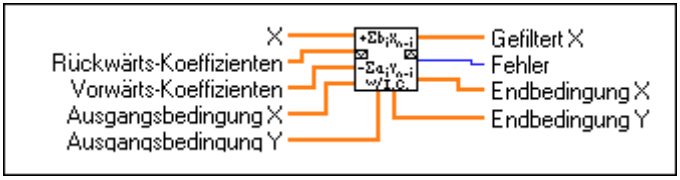
Wenn *y* die Ausgangssequenz **Gefiltert X** darstellt, erhält das VI die Elemente von *y* durch

$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{n-1} b_j x_{i-j} - \sum_{k=1}^{m-1} a_k y_{i-k}\right),$$

wobei *n* die Anzahl der **Vorwärts-Koeffizienten** (dargestellt durch b_j) und *m* die Anzahl der **Rückwärts-Koeffizienten** (dargestellt durch a_k) ist.

IIR Filter mit IC

Filtert die Eingangssequenz **X** unter Verwendung des IIR-Filters mit direkter Form, wie es durch die **Rückwärts-Koeffizienten** und **Vorwärts-Koeffizienten** festgelegt ist.



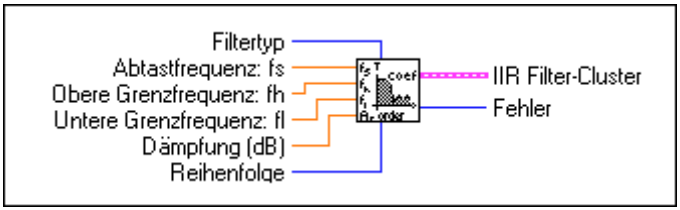
Wenn *y* die Ausgangssequenz **Gefiltert X** darstellt, erhält das VI die Elemente von *y* durch

$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{n-1} b_j x_{i-j} - \sum_{k=1}^{m-1} a_k y_{i-k}\right),$$

wobei *n* die Anzahl der **Vorwärts-Koeffizienten** (dargestellt durch b_j) und *m* die Anzahl der **Rückwärts-Koeffizienten** (dargestellt durch a_k) ist.

Inverse Chebyshev-Koeffizienten

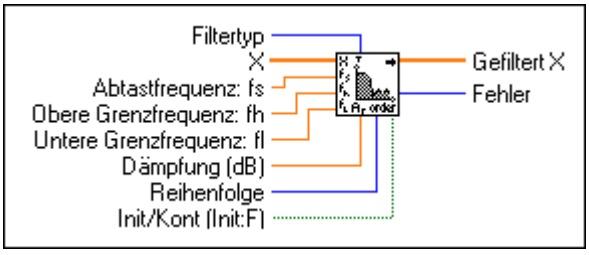
Erzeugt den Satz Filterkoeffizienten zur Realisierung eines IIR-Filters entsprechend dem Chebyshev-II-Filtermodell. Diese Koeffizienten können an das VI IIR Kaskaden-Filter zur Filterung einer Datensequenz weitergegeben werden.



Das VI Inverse Chebyshev-Koeffizienten ist ein SubVI des VIs Inverses Chebyshev-Filter.

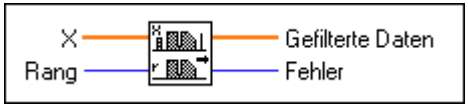
Inverses Chebyshev-Filter

Erzeugt ein digitales Chebyshev-II-Filter unter Verwendung der vorgegebenen **Abtastfrequenz: fs**, **Obere Grenzfrequenz fh**, **Untere Grenzfrequenz fl**, **Dämpfung** in Dezibel, **Filtertyp** und Filter-**Reihenfolge**, indem es das VI Inverse Chebyshev-Koeffizienten aufruft. Das VI Inverser Chebyshev-Filter filtert die Eingangssequenz **X** entsprechend diesem Modell, um durch Aufruf der VIs IIR Filter eine Chebyshev-II-Sequenz **Gefiltert X** zu gewinnen.



Median-Filter

Wendet ein Median-Filter von **Rang** auf die Eingangssequenz **X** an.



Wenn *Y* die Ausgangssequenz **Gefilterte Daten** darstellt und J_i eine Teilmenge der Eingangssequenz **X** mit dem *i.* Element von **X** als ungefähren Mittelpunkt darstellt, ist

$$J_i = \{x_{i-r}, x_{i-r+1}, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_{i+r-1}, x_{i+r}\},$$

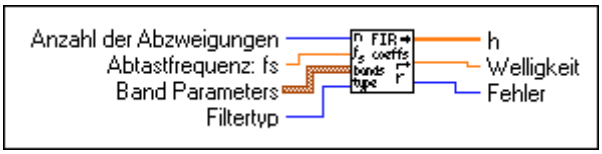
und wenn die indexierten Elemente außerhalb des Bereichs von **X** gleich Null sind, erhält das VI die Elemente von *y* durch

$$y_i = \text{Median}(J_i) \quad \text{für } i = 0, 1, 2, \ldots, n-1,$$

wobei *n* die Anzahl der Elemente in der Eingangssequenz **X** und *r* der Filter-**Rang** ist.

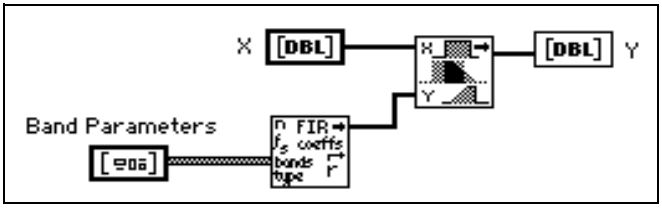
Parks-McClellan

Erzeugt einen Satz Koeffizienten für ein digitales Mehrband-FIR-Filter mit linearer Phase, indem es **Anzahl der Abzweigungen**, **Abtastfrequenz: fs**, **Bandparameter** und **Filtertyp** verwendet.



☞ **Hinweis** ***Dieses VI findet die Koeffizienten durch iterative Verfahren, die nach einem Fehlerwert aufgestellt sind. Es kann durchaus passieren, daß der Algorithmus keine Konvergenz herstellen kann, obwohl gültige Filterparameter vorgegeben wurden.***

Das VI Parks-McClellan erzeugt nur die Filter-Koeffizienten. Es führt selbst keine Filterungsfunktion aus. Zur Filterung einer Sequenz **X** unter Verwendung des Satzes FIR-Filterkoeffizienten **h** ist das VI Faltung mit **X** und **h** als Eingangssequenzen einzusetzen.



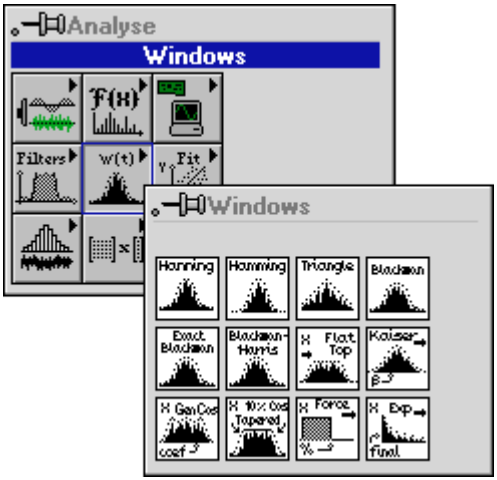
Die Equi-Ripple Filter (gleicher Welligkeit) verwenden ein ähnliches Verfahren zur Filterung der Daten.

42

Fenster-VIs

Dieses Kapitel beschreibt die VIs, die die Fensterglättung realisieren.

Auf die **Windows**-Palette (**Fenster**-Palette) wird über das Menü **Funktionen»Analyse»Windows** zugegriffen. Die folgende Abbildung zeigt die auf der **Fenster**-Palette verfügbaren Optionen.



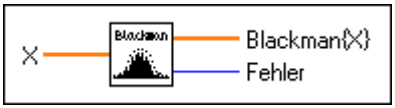
Beispiele für die Verwendung der Fenster-VIs finden Sie unter den Beispielen in `examples\analysis\windxmpl.llb`.

Beschreibungen der Fenster-VIs

Die folgenden Fenster-VIs stehen zur Verfügung.

Blackman-Fenster

Wendet ein Blackman-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Blackman{X}** darstellt, gewinnt das VI die Elemente von *y* durch

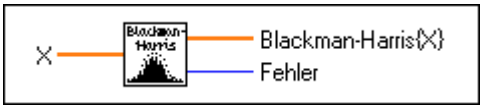
$$y_i = x_i\,[0{,}42 - 0{,}50\cos(w) + 0{,}08\cos(2w)] \quad \text{für } i = 0, 1, 2, \ldots, n-1,$$

$$w = \frac{2\pi i}{n},$$

wobei *n* die Anzahl der Elemente in **X** ist.

Blackman-Harris-Fenster

Wendet ein dreigliedriges Blackman-Harris-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Blackman-Harris{X}** darstellt, gewinnt das VI die Elemente von *y* durch

$$y_i = x_i\,[0{,}42323 - 0{,}49755\cos(w) + 0{,}07922\cos(2w)]$$

für $i = 0, 1, 2, \ldots, n-1$

$$w = \frac{2\pi i}{n},$$

wobei *n* die Anzahl der Elemente in **X** ist.

Cosine-Tapered-Fenster

Wendet ein Cosine-Tapered-Fenster auf die Eingangssequenz **X** an.

Wenn *y* die Ausgangssequenz **Cosine Tapered{X}** darstellt, gewinnt das VI die Elemente von *y* durch

$$y_i = \begin{cases} 0,5x_i(1-\cos w) & \text{für i = 0, 1, 2, ..., m-1, und für i = n-m, n-m+1, ..., n-1} \\ x_i & \text{anderweitig} \end{cases}$$

$$\text{wobei } w = \frac{2\pi i}{n},$$

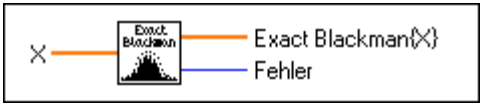
$$m = \text{round}\left(\frac{n}{10}\right), \text{ und}$$

wobei *n* die Anzahl der Elemente der Eingangssequenz **X** ist.

Die Verwendung dieses Fensters kommt der Anwendung des Hanning-Fensters auf die ersten und letzten 10% der Eingangssequenz **X** gleich.

Exaktes Blackman-Fenster

Wendet ein exaktes Blackman-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Exact Blackman{X}** darstellt, gewinnt das VI die Elemente von *y* durch

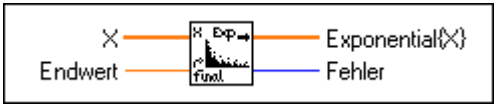
$$y_i = x_i\,[a_0 - a_1\cos(w) + a_2\cos(2w)] \text{ für } i = 0, 1, 2, \ldots, n-1$$

$$w = \frac{2\pi i}{n},$$

wobei *n* die Anzahl der Elemente in **X** ist, a_0 = 7938/18608, a_1 = 9240/18608 und a_2 = 1430/18608 ist.

Exponential-Fenster

Wendet ein Exponential-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Exponential{X}** darstellt, gewinnt das VI die Elemente von *y* durch

$$y_i = x_i \exp(ai) \qquad \text{für } i = 0, 1, 2, \ldots, n-1,$$

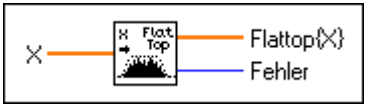
$$a = \frac{\ln(f)}{n-1},$$

wobei *f* der **Endwert** ist und *n* die Anzahl der Abtastwerte in **X** ist.

Dieses VI kann zur Analyse von Einschwingvorgängen eingesetzt werden.

Flat-Top Fenster

Wendet ein Flat-Top-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Flattop{X}** darstellt, gewinnt das VI die Elemente von *y* durch

$$y_i = x_i \,[0{,}2810639 - 0{,}5208972 \cos(w) + 0{,}1980399 \cos(2w)]$$

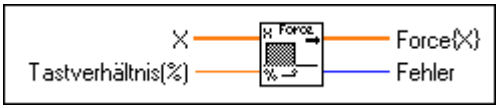
für $i = 0, 1, 2, \ldots, n-1$

$$w = \frac{2\pi i}{n},$$

wobei *n* die Anzahl der Elemente in **X** ist.

Force-Fenster

Wendet ein Force-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Force{X}** ist, gewinnt das VI die Elemente von *y* durch

$$y_i = \begin{cases} x_i & \text{falls } 0 \le i \le d \\ 0 & \text{anderweitig} \end{cases} \quad \text{für } i = 0, 1, 2, ..., n-1$$

d = (0,01)(*n*)(**Tastverhältnis(%)**), wobei *n* die Anzahl der Elemente in **X** ist.

Dieses VI kann auch zur Analyse von Einschwingvorgängen eingesetzt werden.

Allgemeines Cosinus-Fenster

Wendet ein allgemeines Kosinus-Fenster auf die Eingangssequenz **X** an.

Wenn *a* die **Cosinus-Koeffizienten**-Eingangssequenz und *y* die Ausgangssequenz **GenCos{X}** darstellt, gewinnt das VI die Elemente von *y* durch

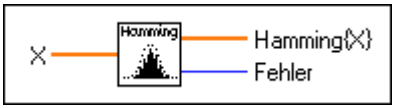
$$y_i = x_i \sum_{k=0}^{m-1} (-1)^k a_k \cos(kw) \quad \text{für } i = 0, 1, 2, \ldots, n-1$$

$$w = \frac{2\pi i}{n},$$

wobei *n* die Anzahl der Elemente in **X** und *m* die Anzahl der **Cosinus-Koeffizienten** ist.

Hamming-Fenster

Wendet ein Hamming-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Hamming{X}** darstellt, gewinnt das VI die Elemente von *y* durch

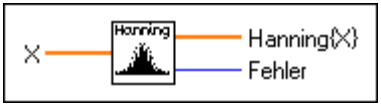
$$y_i = x_i\,[0{,}54 - 0{,}46\cos(w)] \quad \text{für } i = 0, 1, 2, \ldots, n-1,$$

$$w = \frac{2\pi i}{n},$$

wobei *n* die Anzahl der Elemente der Eingangssequenz **X** ist.

Hanning-Fenster

Wendet ein Hanning-Fenster auf die Eingangssequenz **X** an.



Wenn *y* die Ausgangssequenz **Hanning {X}** darstellt, gewinnt das VI die Elemente von *y* durch

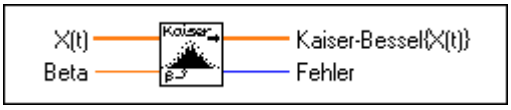
$$y_i = 0{,}5\,x_i\,[1 - \cos(w)] \text{ für } i = 0, 1, 2, \ldots, n-1,$$

$$w = \frac{2\pi i}{n},$$

wobei *n* die Anzahl der Elemente in **X** ist**.**

Kaiser-Bessel-Fenster

Wendet ein Kaiser-Bessel-Fenster auf die Eingangssequenz **X(t)** an.



Wenn *y* die Ausgangssequenz **Kaiser-Bessel{X(t)}** darstellt, gewinnt das VI die Elemente von *y* durch

$$y_i = x_i \frac{I_o(\beta\sqrt{1,0 - a^2})}{I_o(\beta)} \text{ für } i = 0, 1, 2, \ldots, n-1$$

$$a = \frac{i-k}{k},$$

$$k = \frac{n-1}{2},$$

wobei *n* die Anzahl der Elemente in **X(t)** und $I_o(\bullet)$ die modifizierte Bessel-Funktion nullter Ordnung ist.

Dreieck-Fenster

Wendet ein Dreieck-Fenster auf die Eingangssequenz **X** an.

☞ **Hinweis** ***Die Dreieck-Fensterglättung ist auch als Bartlett-Fensterglättung bekannt.***

Wenn *y* die Ausgangssequenz **Dreieck{X}** darstellt, gewinnt das VI die Elemente von *y* durch

$$y_i = x_i \,\text{tri}(w) \text{ für } i = 0, 1, 2, \ldots, n-1,$$

$$w = \frac{2i-n}{n},$$

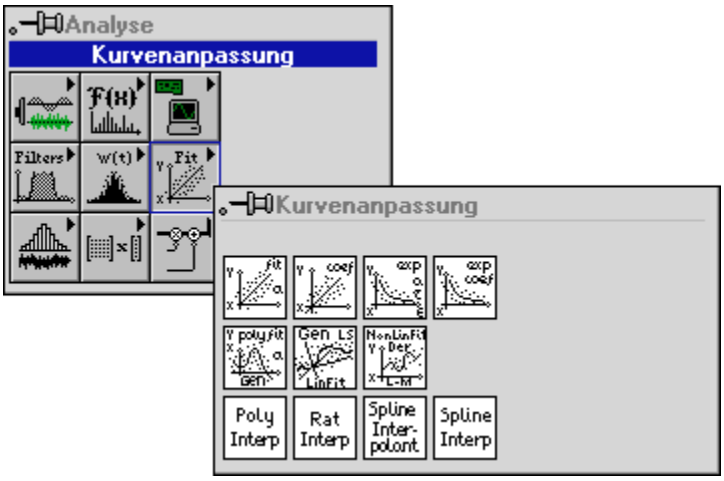
wobei tri(*w*) = 1 – |*w*| und *n* die Anzahl der Elemente in **X** ist.

43

Kurvenanpassungs-VIs

Dieses Kapitel beschreibt die VIs, die Kurvenanpassungen oder Regressionsanalysen ausführen.

Auf die Palette **Kurvenanpassung** wird über das Menü **Funktionen**»**Analyse**»**Kurvenanpassung**, wie in der folgenden Abbildung dargestellt, zugegriffen.



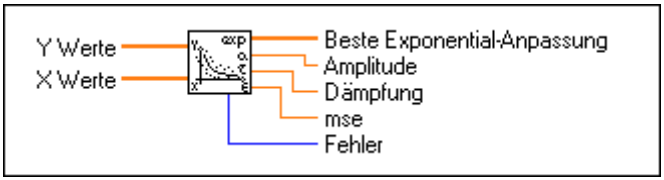
Beispiele für die Verwendung der Regressions-VIs finden Sie unter den Beispielen in `examples\analysis\regressn.llb`.

Beschreibungen der Kurvenanpassungs-VIs

Die folgenden Kurvenanpassungs-VIs stehen zur Verfügung.

Exponentialanpassung

Findet zur Beschreibung der Exponentialkurve, die den Eingangsdatensatz am besten repräsentiert, die Werte der Exponentialkurve und den Satz Exponentialkoeffizienten aus **Amplitude** und **Dämpfung**.



Die allgemeine Form der Exponentialanpassung ist gegeben durch

$$F = ae^{\tau X},$$

wobei *F* die Ausgangssequenz **Beste Exponential-Anpassung**, *X* die Eingangssequenz **X Werte**, *a* die **Amplitude** und τ die **Dämpfungs**-Konstante ist.

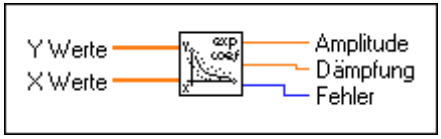
Das VI gewinnt **mse** über die Gleichung

$$\text{mse} = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - y_i)^2,$$

wobei *f* die Ausgangssequenz **Beste Exponential-Anpassung**, *y* die Eingangssequenz **Y Werte** und *n* die Anzahl der Datenpunkte ist.

Koeffizienten der Exponentialanpassung

Findet zur Beschreibung der Exponentialkurve, die den Eingangsdatensatz am besten darstellt, den Satz Exponentialkoeffizienten aus **Amplitude** und **Dämpfung**.



Dieses VI ist ein SubVI des VIs Exponentialanpassung.

Die allgemeine Form der Exponentialanpassung ist gegeben durch

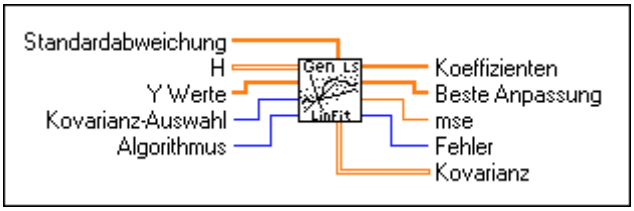
$$F = ae^{\tau X},$$

wobei *F* die Sequenz ist, die die am besten angepaßten Werte darstellt, *X* die Eingangssequenz **X Werte** darstellt, *a* die **Amplitude** und τ die **Dämpfungs**-Konstante ist.

Allgemeine LS Linearanpassung

Findet die k-dimensionale Ebene und den Satz linearer Koeffizienten für die beste Anpassung, indem es die Methode des Chi-Quadrats für Beobachtungsdatensätze anwendet.

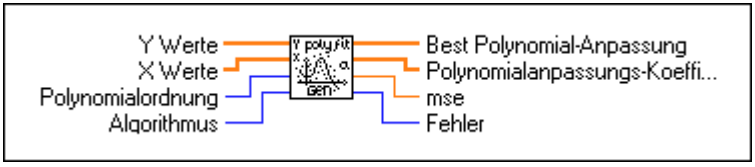
$\{x_{i0}, x_{i1}, \ldots x_{ik-1}, y_i\}$ wobei $i = 0, 1, \ldots, n-1$. *n* ist die Anzahl der Beobachtungsdatensätze.



Dieses VI kann zur Lösung mehrfacher linearer Regressionsaufgaben genutzt werden. Außerdem kann es zum Auffinden der linearen Koeffizienten in einer mehrfachen Funktionsgleichung eingesetzt werden.

Allgemeine Polynomanpassung

Findet zur Beschreibung der polynomischen Kurve, die den Eingangsdatensatz am besten darstellt, die Werte der Polynomkurve und den Satz **Polynomialanpassungs-Koeffizienten**.



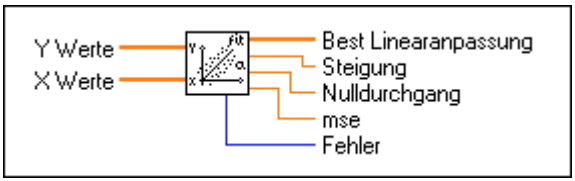
Die allgemeine Form der polynomischen Anpassung ist gegeben durch

$$f_i = \sum_{j=0}^{m} a_j x_i^j,$$

wobei *f* die Ausgangssequenz **Best Polynomialanpassung** darstellt, *x* die Eingangssequenz **X Werte** ist, *a* die **Polynomialanpassungs-Koeffizienten** und *m* die **Polynomialordnung** darstellt.

Lineare Anpassung

Findet zur Beschreibung der Geraden, die den Eingangsdatensatz am besten darstellt, die Werte der Geraden und den Satz linearer Koeffizienten **Steigung** und **Nulldurchgang**.



Die allgemeine Form der linearen Anpassung ist gegeben durch

$$F = mX + b,$$

wobei *F* die Ausgangssequenz **Best Linearanpassung**, *X* die Eingangssequenz **X Werte**, *m* die **Steigung** und *b* **Nulldurchgang** ist.

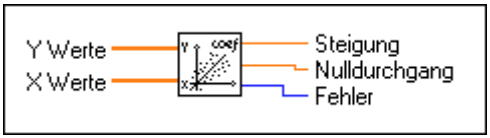
Das VI gewinnt **mse** über die Gleichung

$$\text{mse} = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - y_i)^2,$$

wobei *F* die Ausgangssequenz **Best Linearanpassung**, *y* die Eingangssequenz **Y Werte** und *n* die Anzahl der Datenpunkte ist.

Lineare Anpassungskoeffizienten

Findet zur Beschreibung der Geraden, die den Eingangsdatensatz am besten darstellt, den Satz linearer Koeffizienten **Steigung** und **Nulldurchgang**.



Dieses VI ist ein SubVI des VIs Lineare Anpassung.

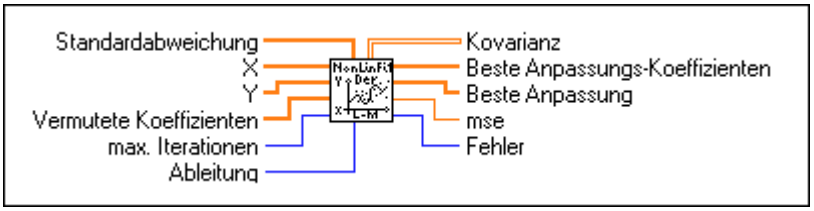
Die allgemeine Form der linearen Anpassung ist gegeben durch

$$F = mX + b,$$

wobei *F* die Sequenz ist, die die am besten angepaßten Werte darstellt. *X* ist die Eingangssequenz **X Werte**; *m* ist die **Steigung,** und *b* ist **Nulldurchgang**.

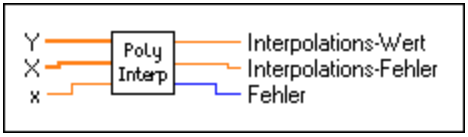
Nicht-lineare Levenberg-Marquardt-Anpassung

Verwendet die Levenberg-Marquardt-Methode zur Bestimmung eines Satzes aus nichtlinearen Koeffizienten, die die Chi-Quadrat-Größe minimieren.



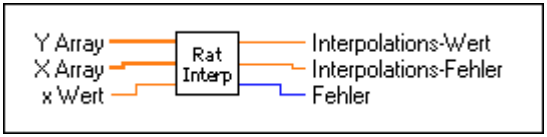
Polynominterpolation

Inter- oder extrapoliert die Funktion *f* bei *x*, bei einem gegebenen Satz von *n* Punkten (x_i, y_i), wobei $f(x_i) = y_i$, *f* eine beliebige Funktion und *x* eine gegebene Zahl ist. Das VI berechnet den Ausgangs-**Interpolations-Wert** $P_{n-1}(x)$, wobei P_{n-1} das eindeutige Polynom von $n-1$ Grad ist, das durch die *n* Punkte (x_i, y_i) verläuft.



Rationale Interpolation

Inter- oder extrapoliert *f* bei **x** unter Anwendung einer rationalen Funktion.



Die rationale Funktion

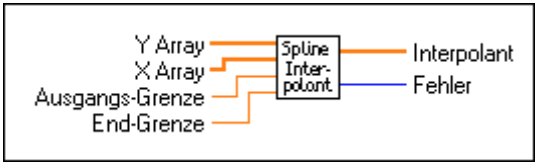
$$\frac{P(x_i)}{Q(x_i)} = \frac{p_0 + p_1 x_i + \ldots + p_m x_i^m}{q_0 + q_1 x_i + \ldots + q_v x^v}$$

verläuft durch alle Punkte, die **Y Array** und **X Array** bilden. *P* und *Q* sind Polynome, und die rationale Funktion ist eindeutig, wenn ein Satz von *n* Punkten (x_i, y_i) gegeben ist, wobei

$f(x_i) = y_i$, f eine beliebige Funktion und x eine gegebene Zahl im Wertebereich der X-Werte ist. Dieses VI berechnet den Ausgangs-**Interpolations-Wert** y durch $y = \frac{P(x)}{Q(x)}$. Wenn die Punkteanzahl ungerade ist, betragen die Freiheitsgrade von P und Q $\frac{n-1}{2}$. Wenn die Punkteanzahl gerade ist, ist der Freiheitsgrad von P gleich $\frac{n}{2} - 1$ und der von Q gleich $\frac{n}{2}$, wobei n die Gesamtanzahl der Punkte ist, die von **Y Array** und **X Array** gebildet werden.

Spline-Interpolant

Gibt ein Array **Interpolant** von der Länge n aus, welches die zweiten Ableitungen von der Spline-Interpolationsfunktion $g(x)$ an den tabulierten Punkten x_i enthält, wobei $i = 0, 1, ..., n-1$ ist. Die Eingabearrays **X Array** und **Y Array** besitzen die Länge n und enthalten eine tabulierte Funktion, $y_i = f(x_i)$, mit $x_0 < x_1 < ... x_{n-1}$. **Ausgangs-Grenze** und **End-Grenze** sind die erste Ableitung der interpolierenden Funktion g(x) an den Punkten 0 bzw. $n-1$.



Falls **Ausgangs-Grenze** und **End-Grenze** gleich oder größer als 10^{30} sind, setzt das VI die entsprechenden Grenzbedingungen für ein natürliches Spline, ohne eine zweite Ableitung an der Grenze.

Die interpolierende Funktion $g(x)$ verläuft durch alle Punkte

$$\{x_i, y_i\},\ g(x_i) = y_i$$

wobei $i = 0, 1, ..., n-1$ ist.

Das VI erhält die interpolierende Funktion $g(x)$, indem es jedes Intervall $[x_i, x_{i+1}]$ mit einer kubischen Polynomfunktion $p_i(x)$ interpoliert, die die folgenden Bedingungen erfüllt:

- $p_i(x_i) = y_i$
- $p_i(x_{i+1}) = y_{i+1}$
- $g(x)$ besitzt überall im Bereich $[x_0, x_{n-1}]$ kontinuierliche erste und zweite Ableitungen:
  - $p_i'(x_i) = p_{i+1}'(x_i)$
  - $p_i''(x_i) = p_{i+1}''(x_i)$

In den vorangegangenen Bedingungen ist $i = 0, 1, ..., n-2$.

Aus der letzten Bedingung, $p_i''(x_i) = p_{i+1}''(x_i)$, ergeben sich die folgenden Gleichungen:

$$\frac{x_i - x_{i-1}}{6} g''(x_{i-1}) + \frac{x_{i+1} - x_{i-1}}{3} g''(x_i) + \frac{x_{i+1} - x_i}{6} g''(x_{i+1})$$

$$= \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \quad i = 1, 2, ...n - 2$$

Bei ihnen handelt es sich um $n-2$ lineare Gleichungen mit n Unbekannten $g''(x_i)$ $i = 0, 1,\ldots, n - 1$. Dieses VI berechnet $g''(x_0)$, $g''(x_{n-1})$ von der **Ausgangs-Grenze** und der **End-Grenze** über die Gleichung

$$g'(x) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{3A^2 - 1}{6}(x_{i+1} - x_i) g''(x_i)$$

$$+ \frac{3B^2 - 1}{6}(x_{i+1} - x_i) g''(x_{i+1}).$$

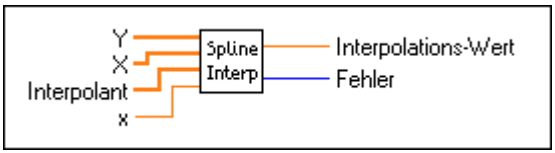
Hierbei gilt

$$A = \frac{x_{i+1} - x}{x_{i+1} - x_i} \qquad B = 1 - A = \frac{x - x_i}{x_{i+1} - x_i}$$

Diese Gleichung kann aus den vorangegangenen Bedingungen abgeleitet werden. Dieses VI verwendet $g''(x_0)$, $g''(x_{n-1})$ zur Lösung aller $g''(x_i)$, für $i = 1, \ldots n - 2$. $g''(x_i)$ ist der Ausgangs-**Interpolant**. Der **Interpolant** kann als Eingabe in das VI Spline Interpolation zur Interpolierung von *y* an jedem beliebigen Wert von $x_0 \le x \le x_{n-1}$ verwendet werden.

Spline Interpolation

Führt eine kubische Spline-Interpolation von *f* an *x* aus, wenn eine tabularische Funktion gegeben ist.



Dieses VI führt kubische Spline-Interpolationen mit einer tabulierten Funktion der Form $y_i = f(x_i)$, für $i = 0, 1, ..., n - 1$, und den gegebenen zweiten Ableitungen **Interpolant**, die das VI von dem VI Spline Interpolant erhält, aus. Der Wert von **x** muß sich im Wertebereich **X Werte** befinden. Die Punkte werden durch die Eingabearrays **X** und **Y** gebildet, und *n* ist die Gesamtanzahl der Punkte.

Für das Intervall $[x_i, x_{i+1}]$ ist der Ausgangs-**Interpolations-Wert** *y* definiert durch

$$y = Ay_i + By_i + 1 + Cy''_i + Dy''_i +,$$

und

$$A = \frac{x_{i+1} - x}{x_{i+1} - x_i},$$

$$B = 1 - A,$$

$$C = \frac{1}{6}(A^3 - A)(x_{i+1} - x_i)^2,$$

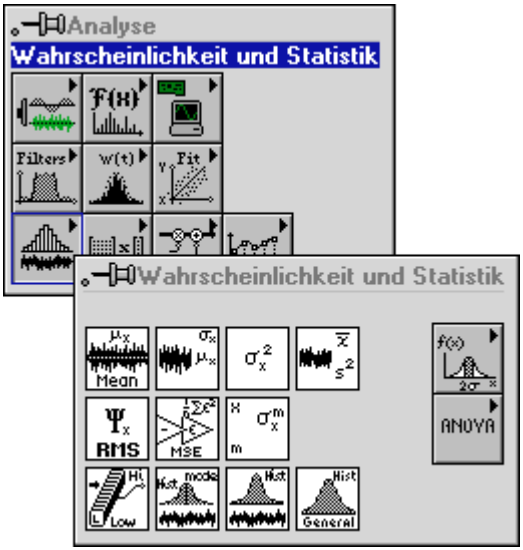
$$D = \frac{1}{6}(B^3 - B)(x_{i+1} - x_i)^2.$$

44

Wahrscheinlichkeits- und Statistik-VIs

Dieses Kapitel beschreibt die VIs, die Wahrscheinlichkeits-, beschreibende Statistik-, Varianzanalysen- und Interpolationsfunktionen ausführen.

Auf die **Wahrscheinlichkeit und Statistik**-Palette wird, wie in der folgenden Abbildung dargestellt, über das Menü **Funktionen»Analyse» Wahrscheinlichkeit und Statistik** zugegriffen.



Beispiele für die Verwendung der Statistik-VIs finden Sie unter den Beispielen in `examples\analysis\statxmpl.llb`.

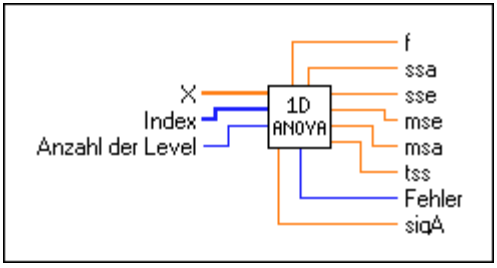
☞ **Hinweis** ***Diese VIs sind nicht im Analyse-Grundpaket enthalten.***

Beschreibungen der Wahrscheinlichkeit und Statistik-VIs

Die folgenden Wahrscheinlichkeit und Statistik-VIs sind verfügbar.

1D ANOVA (Analysis of Variance)

Nimmt ein Array **X** mit experimentellen Beobachtungen auf mehreren **Stufen** eines Faktors, wobei jede Stufe über mindestens eine Beobachtung verfügt, und führt eine einfache Varianzanalyse nach dem Modell mit endlichen Effekten, auch feste Effekte genannt, aus. In der einfachen Varianzanalyse prüft das VI, ob die Stufe des Faktors einen Einfluß auf das experimentelle Ergebnis ausübt.



Faktoren und Stufen

Ein Faktor dient als Grundlage zur Kategorisierung von Daten. Sie können z.B. die Anzahl der Rumpfbeugen, die eine Person machen kann, zählen und als Grundlage zur Kategorisierung das Alter nehmen. Sie könnten für das Alter die folgenden Stufen haben:

Stufe 0 6 bis 10 Jahre alt

Stufe 1 11 bis 15 Jahre alt

Stufe 2 16 bis 20 Jahre alt

Stellen Sie sich jetzt vor, daß Sie eine Reihe von Beobachtungen durchführen, um herauszufinden, wie viele Rumpfbeugen Personen machen können. Wenn Sie eine Zufallsstichprobe von fünf Personen ziehen, könnten Sie u.U. die folgenden Ergebnisse erhalten:

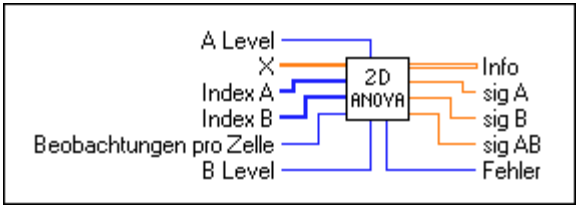
Person 1	8 Jahre (Stufe 0)	10 Rumpfbeugen
Person 2	12 Jahre (Stufe 1)	15 Rumpfbeugen
Person 3	16 Jahre (Stufe 2)	20 Rumpfbeugen
Person 4	20 Jahre (Stufe 2)	25 Rumpfbeugen
Person 5	13 Jahre (Stufe 1)	17 Rumpfbeugen

Bitte beachten Sie, daß Sie mindestens eine Beobachtung pro Stufe gemacht haben. Um eine Varianzanalyse auszuführen, müssen Sie wenigstens eine Beobachtung pro Stufe machen.

Zur Ausführung der Varianzanalyse geben Sie ein Array **X** mit Beobachtungen mit den Werten 10, 15, 20, 25 und 17 ein. Der Array-**Index** gibt die Stufen (oder Kategorien) an, zu der jede Beobachtung gehört. In diesem Fall hat der **Index** die Werte 0, 1, 2, 2 und 1. Zum Schluß geben Sie für den Parameter **Anzahl der Level** einen Wert von 3 ein, weil drei Stufen (oder Level) möglich sind.

2D ANOVA

Nimmt ein Array aus experimentellen Beobachtungen, die auf mehreren Stufen von zwei Faktoren gemacht wurden, und führt eine zweifache Varianzanalyse durch.



Faktoren, Stufen und Zellen

Ein Faktor dient als Grundlage zur Kategorisierung von Daten. Wenn Sie z.B. die Anzahl der Rumpfbeugen, die eine Person machen kann, zählen, kann das Alter als eine Grundlage zur Kategorisierung dienen. Sie könnten für das Alter die folgenden Stufen haben:

Stufe 0 6 bis 10 Jahre alt

Stufe 1 11 bis 15 Jahre alt

Ein anderer möglicher Faktor kann das Körpergewicht mit den folgenden Stufen sein:

Stufe 0 weniger als 50 kg

Stufe 1 zwischen 50 und 75 kg

Stufe 2 mehr als 75 kg

Stellen Sie sich jetzt vor, daß Sie eine Reihe von Beobachtungen gemacht haben, um zu sehen, wie viele Rumpfbeugen Personen machen können. Wenn Sie eine Zufallsstichprobe von *n* Personen gezogen haben, könnten Sie u.U. die folgenden Ergebnisse erhalten haben:

Person 1	8 Jahre (Stufe 0)	30 kg (Stufe 0)	10 Rumpfbeugen
Person 2	12 Jahre (Stufe 1)	40 kg (Stufe 0)	15 Rumpfbeugen
Person 3	15 Jahre (Stufe 1)	76 kg (Stufe 2)	20 Rumpfbeugen
Person 4	14 Jahre (Stufe 1)	60 kg (Stufe 1)	25 Rumpfbeugen
Person 5	9 Jahre (Stufe 0)	51 kg (Stufe 1)	17 Rumpfbeugen
Person 6	10 Jahre (Stufe 0)	80 kg (Stufe 2)	4 Rumpfbeugen

und so weiter.

Wenn die Beobachtungen als eine Funktion von Faktor A und Faktor B aufgezeichnet werden, fallen sie in die Zellen einer Matrix aus Faktor A als Zeilen und Faktor B als Spalten. Jede Zelle muß mindestens eine Beobachtung enthalten, und jede Zelle muß dieselbe Anzahl an Beobachtungen enthalten.

Zur Ausführung der Varianzanalyse geben Sie ein Array **X** mit Beobachtungen mit den Werten 10, 15, 20, 25, 17 und 4 ein. Das Array **Index A** gibt die Stufen (oder Kategorien) von Faktor A an, zu der jede Beobachtung gehört. In diesem Fall würde das Array die Werte 0, 1, 1, 1, 0 und 0 aufführen.

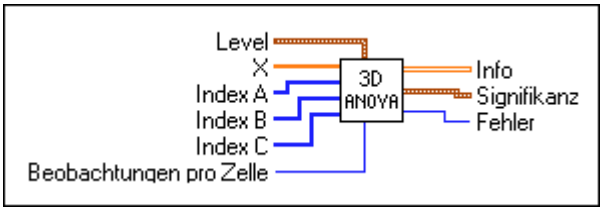
Das Array **Index B** gibt die Stufen (oder Kategorien) von Faktor B an, dem jede Beobachtung angehört. In diesem Fall würde das Array die Werte 0, 0, 2, 1, 1 und 2 haben. Schließlich - da zwei mögliche Stufen von Faktor A und drei von Faktor B vorhanden sind - geben Sie für den Parameter **A Level** einen Wert von 2 und für den Parameter **B Level** einen Wert von 3 ein.

Sie können eins der folgenden Modelle anwenden, wobei *L* die angegebenen **Beobachtungen pro Zelle** ist:

- Modell 1: Endliche (feste) Effekte ohne Wechselwirkungen und mit einer Beobachtung pro Zelle (pro angegebener Stufen *x* und *y* von den Faktoren A bzw. B)
- Modell 2: Endliche Effekte mit Wechselwirkungen und $L > 1$ Beobachtungen pro Zelle.
- Modell 3: Eins der Modelle von gemischten Effekten mit Wechselwirkungen und $L > 1$ Beobachtungen pro Zelle.
- Modell 4: Zufällige Effekte mit Wechselwirkungen und $L > 1$ Beobachtungen pro Zelle.

3D ANOVA

Nimmt ein Array aus experimentellen Beobachtungen, die auf mehreren Stufen von drei Faktoren gemacht wurden, und führt eine dreifache Varianzanalyse aus. In jeder Varianzanalyse suchen Sie nach einem Nachweis, daß die Faktoren oder deren Wechselwirkungen eine signifikante Wirkung auf das experimentelle Ergebnis ausüben. Die verwendete Methode der Varianzanalyse unterscheidet jedes der Modelle.



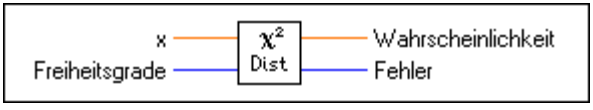
Die Modelle der dreifachen, auch 3-Faktor- oder Drei-Weg-, Varianzanalyse sind die folgenden, wobei *L* die Anzahl der **Beobachtungen pro Zelle** ist:

- Endliche Effekte mit Wechselwirkungen und $L > 1$ Beobachtungen pro Zelle.
- Jedes der sechs Modelle mit gemischten Effekten mit Wechselwirkungen und $L > 1$ Beobachtungen pro Zelle.
- Zufällige Effekte mit Wechselwirkungen und $L > 1$ Beobachtungen pro Zelle.

Ein Faktor dient als Grundlage zur Kategorisierung von Daten. Eine Zelle mit Daten besteht aus all den Beobachtungen, die auf die jeweiligen Stufen aller drei Faktoren fallen. Die Anzahl der Beobachtungen in einer Zelle muß eine gleichbleibende Zahl *L* sein, die sich nicht zwischen den Zellen unterscheidet. Lesen Sie die Beschreibung der Faktoren, Stufen und Zellen in der Beschreibung des VIs 2D ANOVA. Behalten Sie bitte im Gedächtnis, daß eine Zelle in diesem 3D ANOVA-VI die Überschneidung von drei anstatt von zwei Faktoren wie in der Beschreibung des VIs 2D ANOVA bildet.

Chi-Quadratverteilung

Berechnet die einseitige **Wahrscheinlichkeit** *p* der zufälligen Variable **x** mit der χ^2 -Verteilung nach den angegebenen **Freiheitsgraden**.

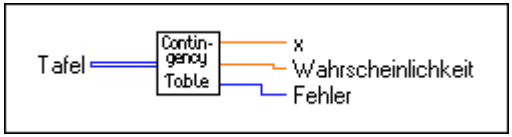


$$p = \text{Prob}\ \{X \leq \mathbf{x}\},$$

wobei X nach einer χ^2 -Verteilung mit *n* **Freiheitsgraden** verteilt ist, *p* die **Wahrscheinlichkeit** ist, *n* die Anzahl von **Freiheitsgraden** ist *und* **x** der Wert ist.

Kontingenztafel

Klassifiziert Objekte im Experiment und führt eine Strichliste nach zwei Kategorisierungsplänen.



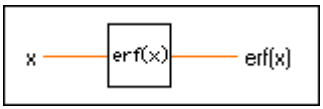
Bei dem χ^2 -Prüftest auf Homogenität nimmt das VI eine zufällige Probe von festgelegter Größe aus jeder Kategorie eines Kategorisierungsplans. Für jede Stichprobe kategorisiert das VI die Experimentalobjekte nach einem zweiten Plan und führt eine Zählung durch. Das VI prüft die Hypothese, um herauszufinden, ob die Grundgesamtheiten, von denen jede Stichprobe stammt, mit Hinblick auf den zweiten Kategorisierungsplan identisch verteilt sind.

Bei dem χ^2 -Prüftest auf Unabhängigkeit nimmt das VI nur eine Stichprobe von der vereinten Grundgesamtheit. Danach kategorisiert das VI jedes Objekt und führt eine Zählung in zwei Kategorisierungsplänen durch. Das VI prüft die Hypothese, daß die Kategorisierungspläne voneinander unabhängig sind.

Für jeden Prüftest muß ein Signifikanzniveau ausgewählt werden, das heißt, wie hoch die Wahrscheinlichkeit sein soll, daß das VI die Hypothese zurückweist, wenn sie wahr ist. Normalerweise soll sie nicht sehr hoch sein. Sie sollten eine kleine Zahl (0,05 oder 5 Prozent ist eine gängige Wahl) verwenden, um das Signifikanzniveau festzulegen. Der Ausgabeparameter **Wahrscheinlichkeit** ist das Signifikanzniveau, auf dem die Hypothese zurückgewiesen wird. Wenn die **Wahrscheinlichkeit** also kleiner als das Signifikanzniveau ist, muß die Hypothese verworfen werden.

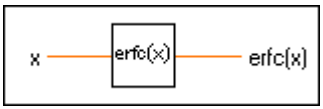
erf(x)

Bewertet die Fehlerfunktion am Eingangswert.



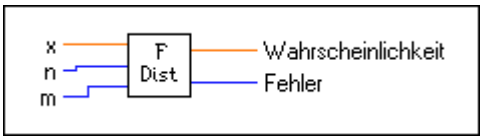
erfc(x)

Bewertet die komplementäre Fehlerfunktion am Eingangswert.



F-Verteilung

Berechnet die einseitige **Wahrscheinlichkeit** *p* der Zufallsvariable *F* mit einer F-Verteilung nach den angegebenen **n** und **m** Freiheitsgraden

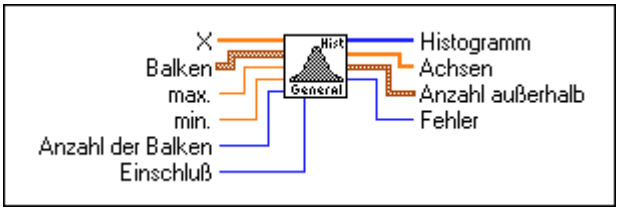


$$p = \text{Prob}\{F_{n,m} \leq \mathbf{x}\},$$

wobei *F* nach einer *F*-Verteilung verteilt ist, *p* die **Wahrscheinlichkeit** ist, **n** den ersten Freiheitsgrad angibt, **m** den zweiten Freiheitsgrad angibt und **x** der Wert ist.

Allgemeines Histogramm

Findet das diskrete Histogramm der Eingangssequenz **X** entsprechend gegebener Balken-Angaben.



Das VI erhält das **Histogramm** wie folgt. Zuerst richtet das VI alle Intervalle (auch Balken genannt) nach den Informationen im Eingangs-Array **Balken** ein. Die Intervalle (Balken = Bins) sind:

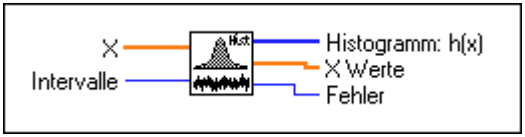
$$\Delta_i = (\textbf{Balken}[i].\text{unterhalb}: \textbf{Balken}[i].\text{oberhalb}) i = 0, 1, 2, ..., k–1$$

wobei **Balken**[*i*].unterhalb der Wert **Unterhalb** im *i*-ten. Cluster vom Array **Balken** ist, **Balken**[*i*].oberhalb der Wert **Oberhalb** im *i*-ten. Cluster vom Array **Balken** ist, *k* die Anzahl von Elementen in **Balken** ist, die sich aus der Zahl aller Intervalle (Balken = Bins) zusammensetzen.

Ob die zwei Endpunkte **Balken**[*i*].unterhalb und **Balken**[*i*].oberhalb von jedem Intervall (Kasten) in dem Intervall (Kasten) Δ_i eingeschlossen sind, hängt von dem Wert von **Balken einschließen** in dem korrespondierenden Cluster *i* der **Balken** ab.

Histogramm

Findet das diskrete Histogramm der Eingangssequenz **X**. Das Histogramm ist eine Häufigkeitszählung, wie oft ein bestimmtes Intervall in der Eingangssequenz auftritt.



Wenn die Eingangssequenz

$$\mathbf{X} = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 8\},$$

ist, dann ist das **Histogramm: h(x)** von **X** für acht **Intervalle**

$$h(X) = \{h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7\} = \{1, 1, 0, 2, 3, 2, 0, 1\},$$

Beachten Sie bitte, daß das Histogramm der Eingangssequenz **X** eine Funktion von **X** ist.

Das VI erhält das **Histogramm: h(x)** wie folgt. Das VI sucht die Eingangssequenz **X** ab, um den enthaltenen Wertebereich zu bestimmen. Danach richtet das VI die Intervallbreite, Δx, entsprechend der vorgegebenen Anzahl von **Intervalle** ein,

$$\Delta x = \frac{\max - \min}{m},$$

wobei max der in der Eingangssequenz **X** gefundene Höchstwert ist, min der in der Eingangssequenz **X** gefundene Kleinstwert ist und *m* die festgelegte Anzahl der **Intervalle** ist.

χ sei die Ausgangssequenz **X Werte**, weil das Histogramm eine Funktion von **X** ist. Das VI bewertet Elemente von χ durch

$$\chi_i = \min + 0,5\Delta x + i\Delta x \quad \text{für } i = 0, 1, 2, \ldots, m-1.$$

Das VI erhebt das *i*. Intervall Δ_i zu dem Wertebereich von $\chi_i - 0{,}5\ \Delta x$ bis zu, doch nicht eingeschlossen, $\chi_i + 0{,}5\ \Delta x$,

$$\Delta_i = (\chi_i - 0{,}5\ \Delta x : \chi_i + 0{,}5\ \Delta x), \text{ für } i = 0, 1, 2, \ldots, m-1,$$

und legt die Funktion $y_i(x)$ fest als

$$y_i(x) = \begin{cases} 1 & \text{wenn } x \in \supseteq \Delta_i \\ 0 & \text{anderweitig} \end{cases}.$$

Die Funktion hat einen Wert von Eins, wenn der Wert von *x* innerhalb des festgelegten Intervalls fällt. Ansonsten beträgt er Null. Beachten Sie, daß das Intervall Δ_i um χ_i zentriert und seine Breite Δ_x ist.

Das letzte Intervall, Δ_{m-1}, ist mit $[\chi_{m-1} - 0{,}5\Delta x : \chi_{m-1} + 0{,}5\Delta x]$ definiert. Mit anderen Worten, wenn ein Wert dem Höchstwert max entspricht, wird er zum letzten Intervall gezählt.

Letztendlich bewertet das VI die Histogramm-Sequenz *H* durch

$$h_i = \sum_{j=0}^{n-1} y_i(x_j) \text{ für } i = 0, 1, 2, \ldots, m-1,$$

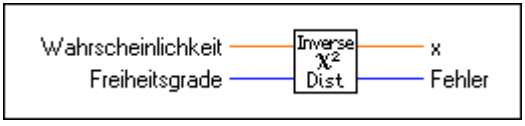
wobei h_i die Elemente der Ausgangssequenz **Histogramm: h(x)** und *n* die Anzahl von Elementen in der Eingangssequenz **X** ist.

Inverse Chi-Quadratverteilung

Berechnet den Wert von **x** derart, daß die Bedingung

$p = \text{Prob}\ \{X \leq \mathbf{x}\}$

erfüllt ist, bei einem Wahrscheinlichkeits-Wert *p* einer χ^2 -verteilten Zufallsvariable *X* mit *n* **Freiheitsgraden**.

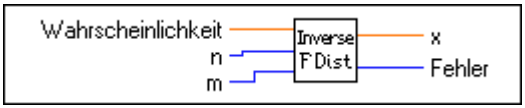


Inverse F-Verteilung

Berechnet den Wert von **x** derart, daß die Bedingung

$$p = \{Prob_{n,m} \leq \mathbf{X}\}$$

erfüllt ist, bei einem Wahrscheinlichkeits-Wert *p* einer F-verteilten Zufallsvariable *F* mit **n** und **m** Freiheitsgraden.



Inverse Normalverteilung

Berechnet den Wert von **x** derart, daß die Bedingung

$$p = \text{Prob}\ \{X \leq \mathbf{x}\}$$

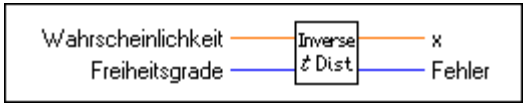
erfüllt ist, bei einem Wahrscheinlichkeits-Wert *p* einer normalverteilten Zufallsvariable *X*.

Inverse T-Verteilung

Berechnet den Wert von **x** derart, daß die Bedingung

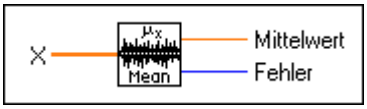
$$p = \text{Prob}\ \{T_n \leq \mathbf{x}\}$$

erfüllt ist, bei einem **Wahrscheinlichkeit**s-Wert *p* von einer t-verteilten Zufallsvariable *T* mit *n* **Freiheitsgraden**.



Mittelwert

Berechnet den Mittelwert (Durchschnitt) der Werte in der Eingangssequenz **X**.



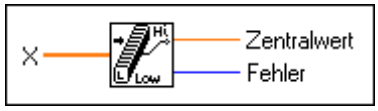
Dieses VI berechnet den **Mittelwert** (μ) nach der folgenden Gleichung:

$$\mu = \frac{1}{n}\sum_{i=0}^{n-1} x_i ,$$

wobei *n* die Anzahl von Elementen in **X** ist.

Zentralwert

Findet den Zentralwert (Median) der Eingangssequenz **X**, indem es die Werte von **X** sortiert und das mittlere Element/die mittleren Elemente im Array auswählt



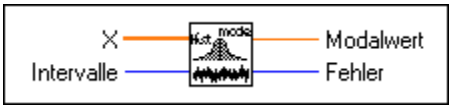
Es sei *n* die Anzahl der Elemente in der Eingangssequenz **X** und *S* die sortierte Sequenz von **X**. Das VI findet den **Zentralwert** durch die folgende Identität:

$$\text{Zentralwert} = \begin{cases} s_i & \text{falls n ungerade ist} \\ 0.5(s_{k-1} + s_k) & \text{wenn n gerade ist} \end{cases},$$

$$\text{wobei } i = \frac{n-1}{2} \text{ und } k = \frac{n}{2} \text{ ist.}$$

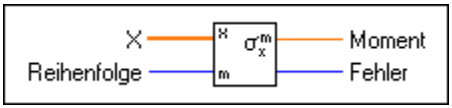
Modalwert

Findet den **Modalwert** der Eingangssequenz **X**.



Moment in Bezug auf den Mittelwert

Berechnet das Moment in Bezug auf den Mittelwert der Eingangssequenz **X** durch die **Reihenfolge**.



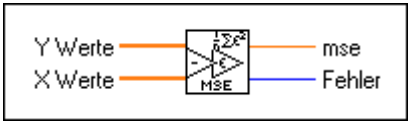
Es sei *m* die gewünschte **Reihenfolge**. Das VI berechnet **Moment** der *m*. Ordnung durch die folgende Gleichung:

$$\sigma_x^m = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \mu)^m,$$

wobei σ_x^m das **Moment** der *m*. Ordnung und *n* die Anzahl von Elementen in der Eingangssequenz **X** ist.

Mittlerer quadratischer Fehler

Berechnet den mittleren quadratischen Fehler (**mse**) der Eingangssequenzen **X Werte** und **Y Werte**.



Das VI gewinnt den **mse** durch folgende Gleichung:

$$\text{mse} = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - y_i)^2,$$

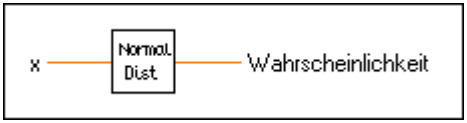
wobei *n* die Anzahl der Datenpunkte ist.

Normalverteilung

Berechnet die einseitige **Wahrscheinlichkeit** *p* der normalverteilten Zufallsvariable **x**,

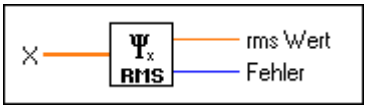
$$p = \text{Prob}\ \{X \leq \mathbf{x}\},$$

wobei *X* standardmäßig normalverteilt, *p* die **Wahrscheinlichkeit** und **x** der Wert ist.



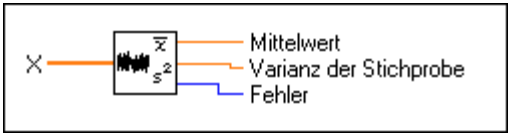
Quadratisches Mittel

Berechnet das quadratische Mittel (rms) der Eingangssequenz **X**.



Varianz der Stichprobe

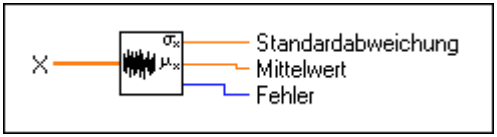
Berechnet den **Mittelwert** und die **Varianz der Stichprobe** von den Werten in der Eingangssequenz **X**.



☞ **Hinweis** ***Wenn die Stichproben-Standardabweichung von* X *berechnet werden soll, ist die Quadratwurzel der* Varianz der Stichprobe *zu ziehen.***

Standardabweichung

Berechnet den Mittelwert und die Standardabweichung der Werte in der Eingangssequenz **X**.



Dieses VI berechnet die **Standardabweichung** (σ_x) und den **Mittelwert** (μ) nach der folgenden Gleichung:

$$\sigma_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \mu)^2},$$

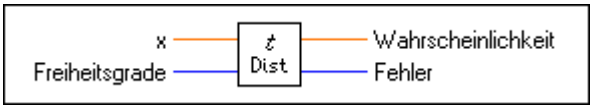
wobei $\mu = \frac{1}{n}\sum_{i=0}^{n-1} x_i$ und *n* die Anzahl von Elementen in **X** ist.

T-Verteilung

Berechnet die einseitige **Wahrscheinlichkeit** *p* der Zufallsvariable T_n mit der t-Verteilung nach den angegebenen **Freiheitsgraden**

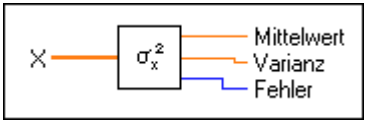
$$p = \text{Prob}\,\{T_n \le \mathbf{x}\},$$

wobei *T* in einer t-Verteilung verteilt, *p* die **Wahrscheinlichkeit**, *n* die **Freiheitsgrade** und **x** der Wert ist.



Varianz

Berechnet die Varianz und den Mittelwert der Eingangssequenz **X**.



Dieses VI berechnet die **Varianz** (σ_x^2) und den **Mittelwert** (μ) nach der folgenden Gleichung:

$$\sigma_x^2 = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \mu)^2,$$

wobei $\mu = \frac{1}{n}\sum_{i=0}^{n-1} x_i$ und *n* die Anzahl von Elementen in **X** ist.

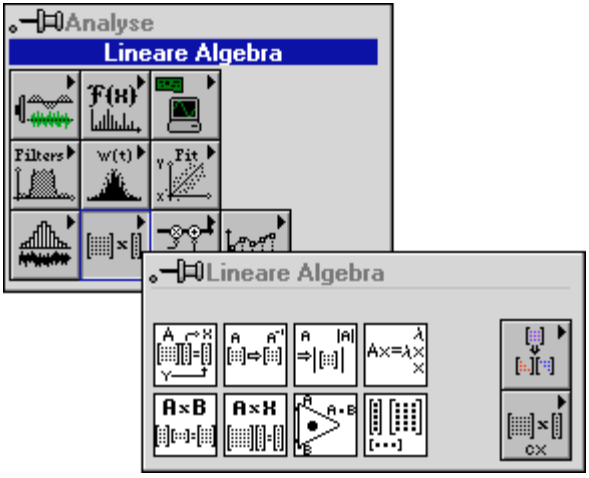
45

Lineare-Algebra-VIs

Dieses Kapitel beschreibt die VIs, die reelle und komplexe auf Matrizen bezogene Berechnungen und Analysen, einschließlich der folgenden, ausführen:

- Grundlegende Matrix-Manipulationen
- Lösen von linearen Gleichungen und Matrixumkehrungen
- Eigenwerte und Eigenvektoren
- Matrix-Analyse

Auf die Palette **Lineare Algebra** wird, wie in der folgenden Abbildung dargestellt, über das Menü **Funktionen**»**Analyse**»**Lineare Algebra** zugegriffen.



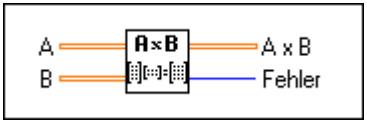
Beispiele für die Verwendung der Lineare-Algebra-VIs finden Sie unter den Beispielen in `examples\analysis\linaxmpl.llb`.

Beschreibungen von Lineare-Algebra-VIs

Die folgenden Lineare-Algebra-VIs sind verfügbar.

A x B

Führt die Matrixmultiplikation zweier Eingabematrizen aus.



Wenn *A* eine *n* x *k*-Matrix und *B* eine *k* x *m*-Matrix ist, ergibt die Matrixmultiplikation von *A* und *B*, *C* = *AB*, eine Matrix *C* mit den Dimensionen *n* x *m*. *A* sei das 2D-Eingangs-Array **A** Matrix; *B* sei das 2D-Eingangs-Array **B** Matrix, und *C* sei das 2D-Ausgangs-Array **A * B**. Das VI gewinnt die Elemente von *C* durch die Gleichung

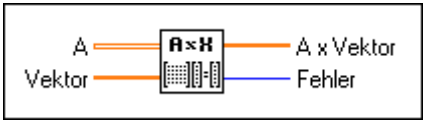
$$c_{ij} = \sum_{l=0}^{k-1} a_{il} b_{lj} \qquad \text{für} \begin{cases} i = 0, 1, 2, ..., n-1 \\ j = 0, 1, 2, ..., m-1 \end{cases},$$

wobei *n* die Anzahl der Reihen in der **A**-Matrix ist, *k* die Anzahl der Spalten in der **A**-Matrix und die Anzahl der Reihen in der **B**-Matrix ist und *m* die Anzahl der Spalten in der **B**-Matrix ist.

☞ **Hinweis** ***Das VI A x B führt eine strikte Matrixmultiplikation und keine 2D-Einzelelement-Multiplikation aus. Für eine Einzelelement-Multiplikation muß die LabVIEW Funktion Multiplizieren eingesetzt werden. Allgemein gilt AB≠BA.***

A x Vektor

Führt die Multiplikation einer Eingangs-Matrix und eines Eingangs-Vektors aus.



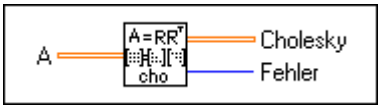
Wenn **A** eine *n* x *k*-Matrix und *X* ein Vektor mit *k* Elementen ist, ergibt die Multiplikation von **A** und *X*, *Y* = *AX*, einen Vektor *Y* mit *n* Elementen. *Y* sei die Ausgabe **A x Vektor**. Das VI gewinnt die Elemente von *Y* über die Gleichung

$$y_i = \sum_{j=0}^{k-1} a_{ij}x_j, \text{ für } i = 0, 1, 2, \ldots, n-1,$$

wobei *n* die Anzahl der Reihen in **A** ist und *k* die Anzahl der Spalten in **A** und die Anzahl der Elemente in *X* ist.

Cholesky-Faktorisierung

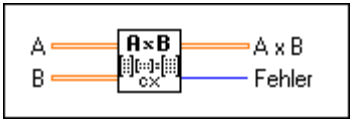
Führt die Cholesky-Faktorisierung an einer realen, positiven endlichen Matrix **A** aus.



Wenn die reale Quadrat-Matrix **A** positiv endlich ist, kann sie in $A = R^T R$ zerlegt werden, wobei R eine obere Dreiecks-Matrix und R^T die transponierte Matrix von R ist.

A x B; komplex

Führt die Matrix-Multiplikation zweier komplexer Eingangs-Matrizen aus.



Wenn *A* eine *n* x *k*-Matrix und *B* eine *k* x *m*-Matrix ist, ergibt die Matrix-Multiplikation von *A* und *B*, *C* = *AB*, eine Matrix *C* mit den Dimensionen *n* x *m*. Es sei *A* die 2D-Eingangs-Array-**A**-Matrix; *B* sei die 2D-Eingangs-Array-**B-Matrix,** und *C* sei das 2D-Ausgangs-Array **A x B**. Das VI gewinnt die Elemente von *C* über die Gleichung

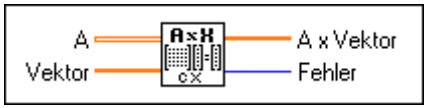
$$c_{ij} = \sum_{l=0}^{k-1} a_{il}b_{lj} \quad \text{für} \begin{cases} i = 0, 1, 2, \ldots, n-1 \\ j = 0, 1, 2, \ldots, m-1 \end{cases},$$

wobei *n* die Anzahl der Reihen in der **A-Matrix** ist, *k* die Anzahl der Spalten in der **A-Matrix** und die Anzahl der Reihen in der **B-Matrix** ist und *m* die Anzahl der Spalten in der **B-Matrix** ist.

Hinweis ***Das VI A x B; komplex führt eine strikte Matrixmultiplikation und keine 2D-Einzelelement-Multiplikation aus. Für eine Einzelelement-Multiplikation muß die LabVIEW Funktion Multiplizieren eingesetzt werden. Allgemein gilt AB≠BA.***

A x Vektor; komplex

Führt die Multiplikation einer komplexen Eingangs-Matrix und eines komplexen Eingangs-Vektors aus.



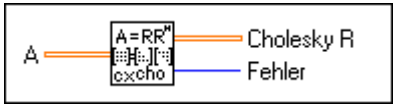
Wenn **A** eine *n* x *k*-Matrix und *X* ein Vektor mit *k*-Elementen ist, ergibt die Multiplikation von **A** und *X*, *Y* = *AX*, einen Vektor *Y* mit *n*-Elementen. Es sei *Y* der Ausgang **A x Vektor;** *X* sei der Eingangs-Vektor. Das VI gewinnt die Elemente von *Y* über die Gleichung

$$y_i = \sum_{j=0}^{k-1} a_{ij} x_j, \text{ für } i = 0, 1, 2, \ldots, n-1,$$

wobei *n* die Anzahl der Reihen in **A** ist und *k* die Anzahl der Spalten in **A** und die Anzahl der Elemente in *X* ist.

Komplexe Cholesky-Faktorisierung

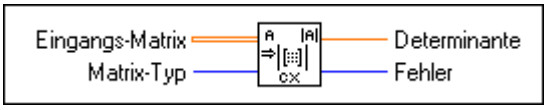
Führt Cholesky-Faktorisierung an einer komplexen, positiven endlichen Matrix **A** aus.



Wenn die komplexe Quadrat-Matrix **A** positiv endlich ist, kann sie in $A = R^H R$ zerlegt werden, wobei *R* eine obere Dreiecks-Matrix und R^H die konjugiert komplexe Transponente von *R* ist.

Komplexe Determinante

Findet die **Determinante** einer komplexen Quadrat-Matrix **Eingangs-Matrix**.



A sei eine Quadrat-Matrix, die die **Eingangs-Matrix** darstellt und *L* und *U* seien die untere bzw. obere Dreiecksmatrix von *A*, so daß

$$A = LU,$$

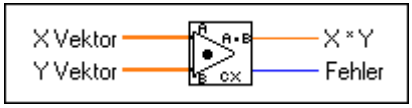
wobei die Elemente der Hauptdiagonalen der unteren Dreiecks-Matrix *L* willkürlich auf eins gesetzt sind. Das VI findet die **Determinante** von *A* über das Produkt der Elemente der Hauptdiagonalen der oberen Dreiecks-Matrix *U*:

$$|A| = \prod_{i=0}^{n-1} u_{ii},$$

wobei $|A|$ die Determinante von **A** und *n* die Dimension von **A** ist.

Komplexes skalares Produkt

Berechnet das skalare Produkt vom komplexen **X Vektor** und **Y Vektor**.



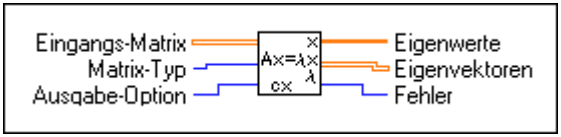
Es sei *X* die Eingangs-Sequenz **X Vektor** und *Y* die Eingangs-Sequenz **Y Vektor**. Das VI gewinnt das skalare Produkt **X*Y** über die Gleichung:

$$X*Y = \sum_{i=0}^{n-1} x_i y_i,$$

wobei *n* die Anzahl der Datenpunkte ist. Beachten Sie bitte, daß der Ausgangswert **X*Y** ein komplexer Skalarwert ist.

Komplexe Eigenwerte und -vektoren

Findet die **Eigenwerte** und rechten **Eigenvektoren** einer komplexen quadratischen Eingangs-Matrix A.



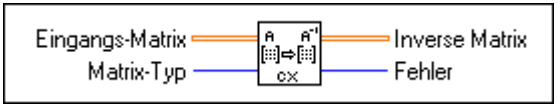
Die Eigenwertaufgabe besteht im Auffinden von nicht trivialen Lösungen zur Gleichung:

$$AX = \lambda X$$

wobei *A* eine *n* x *n*-**Eingangs-Matrix** darstellt, X einen Vektor mit *n* Elementen darstellt und λ ein Skalarwert ist. Die *n* Werte von λ, die der Gleichung genügen, sind die **Eigenwerte** von *A*, und die korrespondierenden Werte von *X* sind die rechten **Eigenvektoren** von *A*. Eine Hermitesche Matrix besitzt immer reale Eigenwerte.

Komplexe Inverse Matrix

Findet die **Inverse Matrix** einer komplexen Matrix **Eingangs-Matrix**.



Es sei *A* die Eingangs-Matrix und *I* die Identitätsmatrix. Sie erhalten die **Inverse Matrix** durch Auflösen des Systems *AB* = *I* nach *B*.

Wenn *A* eine nichtsinguläre Matrix ist, kann gezeigt werden, daß die Lösung zum vorstehenden System einmalig ist und mit der inversen Matrix von *A* korrespondiert

$$B = A^{-1},$$

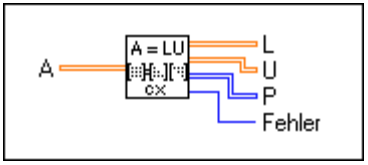
und daß *B* damit die **Inverse Matrix** ist. Eine nichtsinguläre Matrix ist eine Matrix, deren Reihen oder Spalten keine linearen Kombinationen einer anderen Zeile bzw. Spalte enthalten.

Hinweis ***Es kann nicht immer im voraus herausgefunden werden, ob eine Matrix singulär ist, insbesondere nicht bei großen Systemen. Das VI Komplexe Inverse Matrix findet singuläre Matrizen heraus und gibt einen Fehler aus, so daß nicht vor dem Einsatz dieses VIs überprüft werden muß, ob ein gültiges System vorliegt.***

Die numerische Realisierung der Matrixumkehrung ist nicht nur numerisch anspruchsvoll, sondern wegen ihrer Rekursivität auch gegen Aufrundungsfehler vom mathematischen Fließkomma-Koprozessor sehr anfällig. Obwohl das VI für die Berechnungen die größtmögliche Genauigkeit walten läßt, kann es das System nicht unbedingt jedes Mal lösen.

Komplexe LU-Faktorisierung

Führt die komplexe LU-Faktorisierung einer komplexen Quadrat-Matrix **A** aus.



LU-Faktorisierung zerlegt die Quadrat-Matrix **A** in zwei Dreiecks-Matrizen; eine ist eine untere-Matrix **L** mit Einsen auf der Diagonalen, und die andere ist eine obere Dreiecks-Matrix **U**, so daß

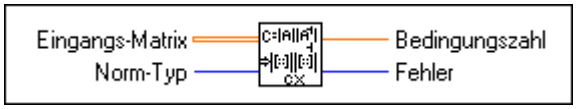
$$PA = LU,$$

wobei **P** eine Permutations-Matrix ist, welche aus der Identitätsmatrix besteht, in der einige Reihen ausgetauscht sind.

Faktorisierung, die Zerlegung in Faktoren, ist der entscheidende Schritt zur Umkehrung einer Matrix, der Berechnung der Determinante einer Matrix und dem Lösen einer linearen Gleichung.

Zustand der komplexen Matrix

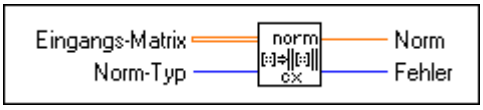
Berechnet die **Bedingungszahl** einer komplexen Matrix **Eingangs-Matrix**.



Die **Bedingungszahl** einer Matrix mißt die Anfälligeit der Lösung des linearen Gleichungssystems gegen Datenfehler. Es stellt eine Einschätzung der Genauigkeit der Ergebnisse der Matrixumkehrung und Lösungen der linearen Gleichungen dar.

Norm der komplexen Matrix

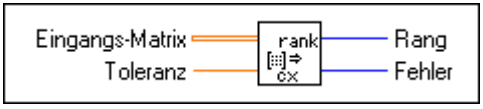
Berechnet die **Norm** einer komplexen Matrix **Eingangs-Matrix**.



Die Norm einer Matrix ist ein Skalarwert, der einen Anhaltspunkt über die Größe der Elemente der Matrix gibt. Es sei **A** die **Eingangs-Matrix**; $\|A\|_p$ sei die **Norm** von A, wobei *p* die Werte $1,2,f,\infty$ haben kann. Unterschiedliche Werte von *p* zeigen die unterschiedlichen Typen von Normen an, die berechnet werden.

Rang der komplexen Matrix

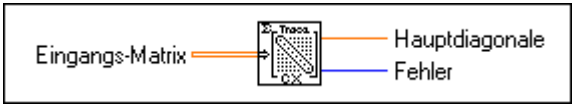
Berechnet den **Rang** einer komplexen Rechtecks-Matrix **Eingangs-Matrix**.



Rang ist die Anzahl der singulären Werte der **Eingangs-Matrix**, die über der **Toleranz liegen**. **Rang** ist die höchste Anzahl von unabhängigen Reihen oder Spalten der **Eingangs-Matrix**.

Komplexe Hauptdiagonale

Findet die **Hauptdiagonale** der **Eingangs-Matrix**.



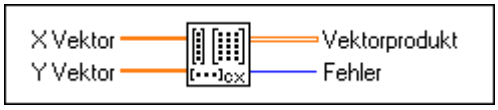
Es sei *A* eine Quadrat-Matrix, die die **Eingangs-Matrix** darstellt, und tr(*A*) sei die **Hauptdiagonale**. Die **Hauptdiagonale** von A ist die Summe der Elemente der Hauptdiagonalen von *A*

$$\mathrm{tr}(A) = \sum_{i=0}^{n-1} a_{ii},$$

wobei *n* die Dimension der **Eingangs-Matrix** ist.

Komplexes Vektorprodukt

Berechnet das Vektorprodukt vom komplexen **X Vektor** und **Y Vektor.**



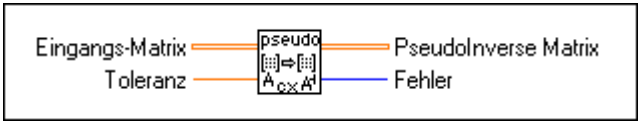
Es sei *X* die Eingangs-Sequenz **X Vektor** und *Y* die Eingangs-Sequenz **Y Vektor**. Das VI ermittelt das **Vektorprodukt** über die Gleichung:

$$a_{ij} = x_i y_j, \text{ für } \begin{cases} i = 0, 1, 2, ..., n-1 \\ j = 0, 1, 2, ..., m-1 \end{cases},$$

wobei *A* die 2D-Ausgangs-Sequenz **Vektorprodukt** darstellt, *n* die Anzahl der Elemente in der Eingangssequenz **X Vektor** und *m* die Anzahl der Elemente in der Eingangssequenz **Y Vektor** ist.

Komplexe Pseudo-Inverse Matrix

Findet die pseudo-inverse Matrix einer komplexen Rechtecks-Matrix **Eingangs-Matrix**.

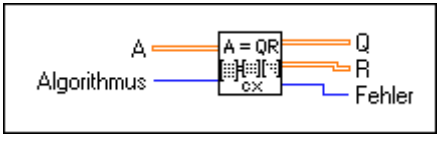


Ein SVD-Algorithmus berechnet die **PseudoInverse Matrix** A^+ und behandelt jeden singulären Wert unterhalb der **Toleranz** als Null.

Wenn die Eingangs-Matrix A quadratisch und nicht singulär ist, ist A^+ dieselbe wie A^{-1}; es ist jedoch effektiver, zur Berechnung von A^{-1} das VI Komplexe inverse Matrix anstatt dieses VI zu verwenden.

Komplexe QR-Faktorisierung

Führt QR-Faktorisierung an einer komplexen Matrix A aus.

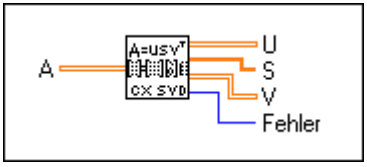


Die QR-Faktorisierung wird auch als orthogonal-trianguläre Faktorisierung bezeichnet. Sie zerlegt eine komplexe Matrix **A** in zwei Matrizen; eine ist eine orthogonale Matrix **Q** und die andere ist eine obere Dreiecks-Matrix **R**, so daß $A = QR$. Dieses VI unterstützt den Householder-Algorithmus.

Die QR-Faktorisierung kann zur Lösung von linearen Sytemen, die weniger oder mehr Gleichungen als Unbekannte enthalten, eingesetzt werden.

Komplexe SVD-Faktorisierung

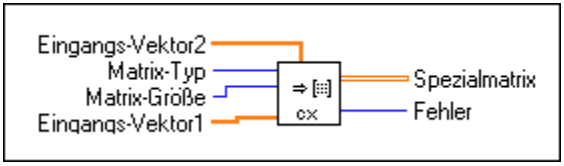
Führt die Zerlegung in singuläre Werte (SVD) an einer komplexen *m* x *n*-Matrix A mit $m > n$ aus.



SVD erzeugt drei Matrizen, **U**, **S** und **V**, so daß $A = US_0V^H$, wobei *U* und *V* orthogonale Matrizen sind, und S_0 eine diagonale *n* x *n*-Matrix ist, mit den Elementen vom Array **S** in absteigender Reihenfolge auf der Diagonalen. Bei den Elementen auf der Diagonalen handelt es sich um die singulären Werte von **A**.

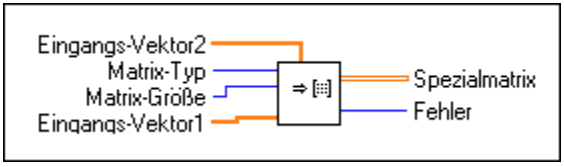
Komplexe Matrix definieren

Erzeugt eine komplexe, auf **Matrix-Typ** basierende Spezialmatrix. Die verfügbaren Matrix-Typen sind Identitäts-, Diagonal-, Toeplitz- und Vandermonde.



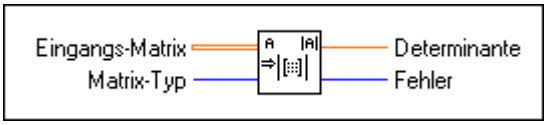
Matrix definieren

Erzeugt eine reale, auf dem **Matrix-Typ** basierende Spezialmatrix. Die verfügbaren Matrix-Typen sind Identitäts-, Diagonal-, Toeplitz- und Vandermonde-Matrix.



Determinante

Berechnet die Determinante einer realen Quadrat-Matrix **Eingangs-Matrix**.



Es sei *A* eine Quadrat-Matrix, die die **Eingangs-Matrix** darstellt, und *L* und *U* seien die untere bzw. die obere Dreiecks-Matrix von *A*, so daß

$$A = LU,$$

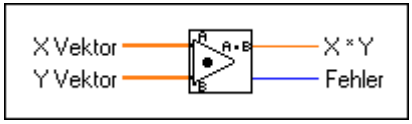
wobei die Elemente der Hauptdiagonalen der unteren Dreiecks-Matrix *L* willkürlich auf Eins gesetzt wurden. Das VI findet die **Determinante von** *A* über das Produkt der Elemente der Hauptdiagonalen der oberen Dreiecks-Matrix *U*

$$|A| = \prod_{i=0}^{n-1} u_{ii},$$

wobei $|A|$ die Determinante von *X* und *n* die Dimension von *X* ist.

Skalares Produkt

Berechnet das skalare Produkt vom **X Vektor** und **Y Vektor**.



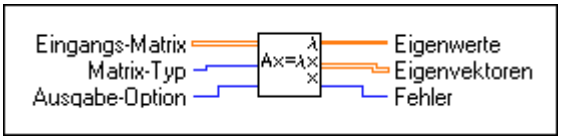
Es sei *X* die Eingangs-Sequenz **X Vektor** und *Y* die Eingangs-Sequenz **Y Vektor**. Das VI gewinnt das skalare Produkt **X*Y** über die Gleichung:

$$X*Y = \sum_{i=0}^{n-1} x_i y_i,$$

wobei *n* die Anzahl der Datenpunkte ist. Beachten Sie bitte, daß der Ausgangswert **X*Y** ein Skalarwert ist.

Eigenwerte und -vektoren

Findet die Eigenwerte und Eigenvektoren direkt von einer realen quadratischen **Eingangs-Matrix**.



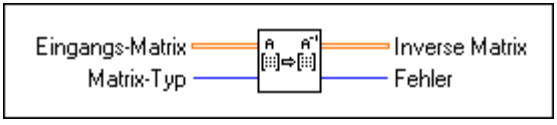
Die Eigenwertaufgabe besteht im Auffinden von nicht trivialen Lösungen zur Gleichung:

$$AX = \lambda X$$

wobei *A* eine *n* x *n*-**Eingangs-Matrix** ist, *X* ein Vektor mit *n* Elementen ist und λ ein Skalar ist. Die *n* Werte von λ, die der Gleichung genügen, sind die **Eigenwerte** von *A* und die korrespondierenden Werte von *X* sind die rechten **Eigenvektoren** von *A*. Eine reale, symmetrische Matrix besitzt immer reale Eigenwerte und Eigenvektoren.

Inverse Matrix

Findet die **Inverse Matrix** der **Eingangs-Matrix**.



Es sei **A** die **Eingangs-Matrix** und **I** die Identitätsmatrix. Sie gewinnen den Wert von **Inverse Matrix** durch Auflösen des Systems *AB* = *I* nach *B*.

Wenn **A** eine nichtsinguläre Matrix ist, kann gezeigt werden, daß die Lösung zum vorstehenden System einmalig ist und mit der **Inversen Matrix** von **A** korrespondiert:

$$B = A^{-1},$$

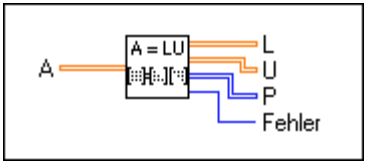
und daß **B** damit eine **Inverse Matrix** ist. Eine nichtsinguläre Matrix ist eine Matrix, deren Reihen oder Spalten keine linearen Kombinationen einer anderen Zeile bzw. Spalte enthalten.

☞ **Hinweis** ***Die numerische Realisierung der Matrixumkehrung ist nicht nur numerisch anspruchsvoll, sondern wegen ihrer Rekursivität auch gegen Aufrundungsfehler vom mathematischen Fließkomma-Koprozessor sehr anfällig. Obwohl das VI für die Berechnungen die größtmögliche Genauigkeit walten läßt, kann es das System nicht unbedingt jedes Mal lösen.***

Es kann nicht immer im voraus herausgefunden werden, ob eine Matrix singulär ist, insbesondere nicht bei großen Systemen. Das VI Inverse Matrix findet singuläre Matrizen heraus und gibt einen Fehler aus, so daß nicht vor dem Einsatz dieses VIs überprüft werden muß, ob ein gültiges System vorliegt.

LU-Faktorisierung

Führt die LU-Faktorisierung an einer realen Quadrat-Matrix **A** aus.



LU-Faktorisierung zerlegt die Quadrat-Matrix **A** in zwei Dreiecks-Matrizen. Eine ist eine untere Dreiecksmatrix **L** mit Einsen auf der Diagonalen. Die andere ist eine obere Dreiecks-Matrix **U**, so daß

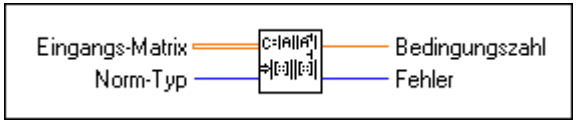
$$PA = LU,$$

wobei **P** eine Permutations-Matrix ist, die als die Identitäts-Matrix dient, in der einige Reihen ausgetauscht sind.

Faktorisierung, die Zerlegung in Faktoren, ist der entscheidende Schritt zur Umkehrung einer Matrix, der Berechnung der Determinante einer Matrix und dem Lösen einer linearen Gleichung.

Zustand der Matrix

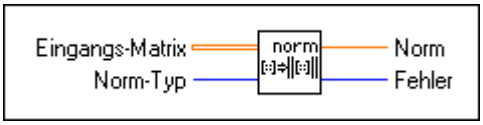
Berechnet die **Bedingungszahl** einer realen Matrix **Eingangs-Matrix**.



Die **Bedingungszahl** einer Matrix mißt die Anfälligkeit einer Systemlösung von linearen Gleichungen gegen Datenfehler. Sie stellt einen Anhaltspunkt zur Einschätzung der Genauigkeit der Ergebnisse einer Matrixumkehrung und der Lösung einer linearen Gleichung dar.

Norm der Matrix

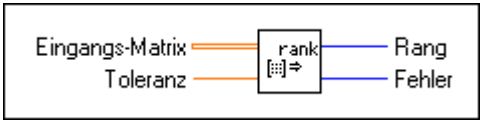
Berechnet die **Norm** einer realen Matrix **Eingangs-Matrix**.



Die Norm einer Matrix ist ein Skalarwert, der eine gewisse Einschätzung der Größenordnung der Elemente in der Matrix erlaubt. Es sei **A** die **Eingangs-Matrix**, die Norm von **A** sei durch $\|A\|_p$ dargestellt, wobei *p* die Werte 1,2,F,∞ haben kann. Unterschiedliche Werte von *p* zeigen die unterschiedlichen Typen von Normen an, die berechnet werden.

Rang der Matrix

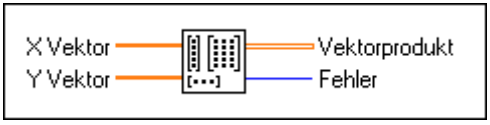
Berechnet den **Rang** einer realen Rechteck-Matrix **Eingangs-Matrix**.



Der Rang der Matrix ist die Anzahl der singulären Werte in der **Eingangs-Matrix**, die über der **Toleranz** liegen. Der **Rang** ist die Höchstanzahl von unabhängigen Reihen oder Spalten in der **Eingangs-Matrix**.

Vektorprodukt

Berechnet das Vektorprodukt von **X Vektor** und **Y Vektor**.



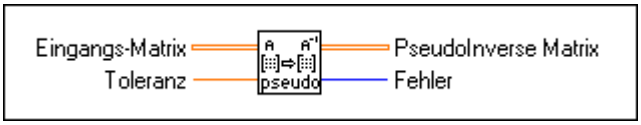
Es sei *X* die Eingangs-Sequenz **X Vektor** und *Y* die Eingangs-Sequenz **Y Vektor**. Das VI gewinnt das **Vektorprodukt** über die Gleichung

$$a_{ij} = x_i y_j, \text{ für } \begin{cases} \text{i} = 0, 1, 2, ..., \text{n} - 1 \\ \text{j} = 0, 1, 2, ..., \text{m} - 1 \end{cases},$$

wobei *A* die 2D-Ausgangs-Sequenz **Vektorprodukt**, *n* die Anzahl der Elemente in der Eingangssequenz **X Vektor** und *m* die Anzahl der Elemente in der Eingangssequenz **Y Vektor** ist.

PseudoInverse Matrix

Findet die **PseudoInverse Matrix** einer realen Rechteck-Matrix **Eingangs-Matrix**.

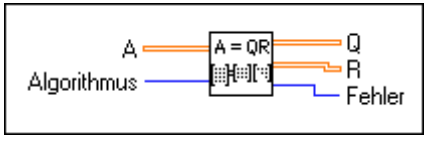


Sie können die **PseudoInverse-Matrix** A^+ mit dem SVD-Algorithmus und einem beliebigen der singulären Werte unterhalb der **Toleranz,** welche auf Null gesetzt wurden, berechnen.

Wenn die Eingangs-Matrix A quadratisch und nicht singulär ist, ist A^+ dieselbe wie A^{-1}, es ist jedoch effektiver, das VI Inverse Matrix als dieses VI zur Berechnung von A^{-1} einzusetzen.

QR-Faktorisierung

Führt die QR-Faktorisierung an einer realen Matrix **A** aus.

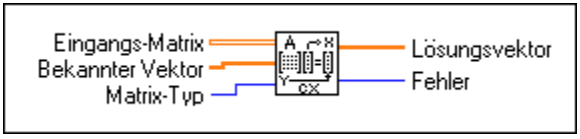


Die QR-Faktorisierung wird auch als orthogonal-trianguläre Faktorisierung bezeichnet. Sie zerlegt eine reale Matrix **A** in zwei Matrizen. Eine ist eine orthogonale Matrix **Q**, und die andere ist eine obere Dreiecks-Matrix **R**, so daß A = QR ist. Dieses VI stellt drei Methoden zur Faktorisierung bereit: Householder, Givens und Fast Givens.

Sie können die QR-Faktorisierung zur Lösung von linearen Systemen mit mehr Gleichungen als Unbekannten einsetzen.

Komplexe Lineargleichungen lösen

Löst ein komplexes, lineares System $AX = Y$.



Es sei A die m x n-**Eingangs-Matrix**, Y sei der Satz von m Elementen in **Bekannter Vektor** und X sei der Satz von n Elementen im **Lösungsvektor**, der folgendes System löst

$$AX = Y.$$

Wenn m > n, besitzt das System mehr Gleichungen als Unbekannte und ist demnach ein überdeterminiertes System. Da die Lösung, die die Gleichung $AX = Y$ befriedigt, u.U. nicht existiert, findet das VI die Lösung X des kleinsten Quadrates, das $||AX–Y||$ minimiert.

Wenn $m < n$, besitzt das System mehr Unbekannte als Gleichungen und ist demnach unterdeterminiert. Es könnte unendlich viele Lösungen zur Gleichung $AX = Y$ besitzen. Das VI wählt dann eine dieser Lösungen aus.

Wenn $m=n$ und A eine nichtsinguläre Matrix ist - keine Zeile oder Spalte ist eine lineare Kombination einer anderen Zeile bzw. Spalte-kann das System durch Zerlegen der **Eingangs-Matrix** A in ihre untere und obere Dreiecksmatrizen L und U nach X gelöst werden, so daß

$$AX = LZ = Y,$$

und

$$Z = UX$$

alternative Darstellungen des Ursprungssystems sind. Beachten Sie, daß Z auch ein Vektor mit n Elementen ist.

Trianguläre Systeme lassen sich mit rekursiven Verfahren leicht lösen. Wenn die Matrizen L und U von A gewonnen werden, können daher Z durch das $LZ = Y$ -System und X durch das $UX = Z$ -System gefunden werden.

Wenn $m \neq n$, kann A in eine orthogonale Matrix Q und eine obere Dreiecks-Matrix R zerlegt werden, so daß $A = QR$ und das lineare System durch $QRX = Y$ dargestellt werden kann. Danach kann $RX = Q^H Y$ gelöst werden.

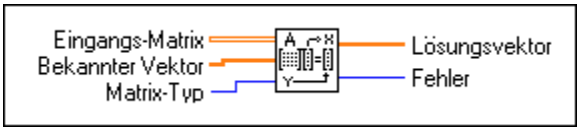
Sie können dieses trianguläre System leicht nach X lösen, indem Sie rekursive Verfahren verwenden.

☞ **Hinweis** ***Es läßt sich nicht immer im voraus herausfinden, ob die Matrix singulär ist, insbesondere nicht bei großen Systemen. Das VI Inverse Matrix entdeckt singuläre Matrizen und gibt einen Fehler aus, so daß vor dem Einsatz dieses VIs nicht überprüft werden muß, ob ein gültiges System vorliegt.***

Die numerische Realisierung der Matrixumkehrung ist numerisch anspruchsvoll und ist wegen ihrer Rekursivität auch gegen Aufrundungsfehler vom mathematischen Fließkomma-Koprozessor sehr anfällig. Obwohl das VI für die Berechnungen die größtmögliche Genauigkeit walten läßt, kann es das System nicht unbedingt jedes Mal lösen.

Lineargleichungen lösen

Löst ein reales lineares System *AX = Y*.



Es sei *A* eine *m* x *n*-Matrix, die die **Eingangs-Matrix** darstellt; *Y* sei der Satz von *m* Koeffizienten in **Bekannter Vektor**, und *X* sei der Satz von *n* Elementen in **Lösungsvektor**, der folgendes System löst

$$AX = Y.$$

Wenn m > n, besitzt das System mehr Gleichungen als Unbekannte und ist demnach ein überdeterminiertes System. Die Lösung, die der Gleichung *AX = Y* genügt, existiert u.U. nicht, deshalb findet das VI die Lösung *X* des kleinsten Quadrates, das $||AX–Y||$ minimiert.

Wenn $m < n$, besitzt das System mehr Unbekannte als Gleichungen und ist demnach unterdeterminiert. Es könnte unendlich viele Lösungen zur Gleichung *AX = Y* besitzen. Das VI findet eine dieser Lösungen.

Im Falle, daß $m=n$ und *A* eine nichtsinguläre Matrix ist - keine Zeile oder Spalte ist eine lineare Kombination einer anderen Zeile bzw. Spalte - kann das System durch Zerlegen der Eingangs-Matrix in ihre untere und obere Dreiecksmatrizen *L* und *U* nach *X* gelöst werden, so daß

$$AX = LZ = Y,$$

und

$$Z = UX$$

alternative Darstellungen des Ursprungssystems sind. Beachten Sie, daß *Z* auch ein Vektor mit *n* Elementen ist.

Trianguläre Systeme lassen sich mit rekursiven Verfahren leicht lösen. Wenn also die Matrizen *L* und *U* von *A* gewonnen werden, können daher *Z* durch das *LZ = Y* -System und *X* durch das *UX = Z* -System gefunden werden.

Im Falle von $m \neq n$ kann A in eine orthogonale Matrix *Q* und eine obere Dreiecks-Matrix *R* zerlegt werden, so daß *A = QR* und das lineare System durch *QRX = Y* dargestellt werden kann. Danach kann $RX = Q^H Y$ gelöst werden.

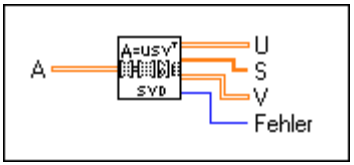
Sie können dieses trianguläre System leicht nach *X* lösen, indem Sie rekursive Verfahren verwenden.

☞ **Hinweis** ***Es läßt sich nicht immer im voraus herausfinden, ob die Matrix singulär ist, insbesondere nicht bei großen Systemen. Das VI Inverse Matrix entdeckt singuläre Matrizen und gibt einen Fehler aus, so daß vor dem Einsatz dieses VIs nicht überprüft werden muß, ob ein gültiges System vorliegt.***

Die numerische Realisierung der Matrixumkehrung ist numerisch anspruchsvoll und ist wegen ihrer Rekursivität auch gegen Aufrundungsfehler vom mathematischen Fließkomma-Koprozessor sehr anfällig. Obwohl das VI für die Berechnungen die größtmögliche Genauigkeit walten läßt, kann es das System nicht unbedingt jedes Mal lösen.

SVD-Faktorisierung

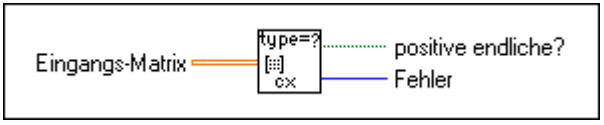
Führt die Zerlegung nach Einzelwerten (SVD) an einer gegebenen realen *m* x *n*-Matrix **A** mit *m* > *n* aus.



SVD erzeugt drei Matrizen *U*, S_0 und *V*, so daß $A = US_0V^T$, wobei *U* und V^T orthogonale Matrizen sind und S_0 eine *n* x *n*-Diagonalmatrix mit den Elementen von Array **S** in absteigender Folge auf der Diagonalen ist.

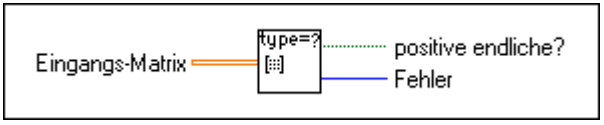
Testen auf komplexe endliche positive Matrix

Testet, ob es sich bei der **Eingangs-Matrix** um eine endliche positive Matrix handelt.



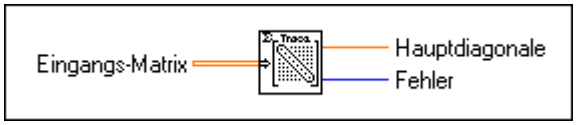
Testen auf endliche positive Matrix

Testet, ob es sich bei der **Eingangs-Matrix** um eine endliche positive Matrix handelt.



Hauptdiagonale

Findet die **Hauptdiagonale** der **Eingangs-Matrix**.



Es sei *A* eine Quadrat-Matrix, die die **Eingangs-Matrix** darstellt, und tr(*A*) sei die **Hauptdiagonale**. Die **Hauptdiagonale** von *A* ist die Summe der Elemente der Hauptdiagonalen von *A*

$$\mathrm{tr}(A) = \sum_{i=0}^{n-1} a_{ii},$$

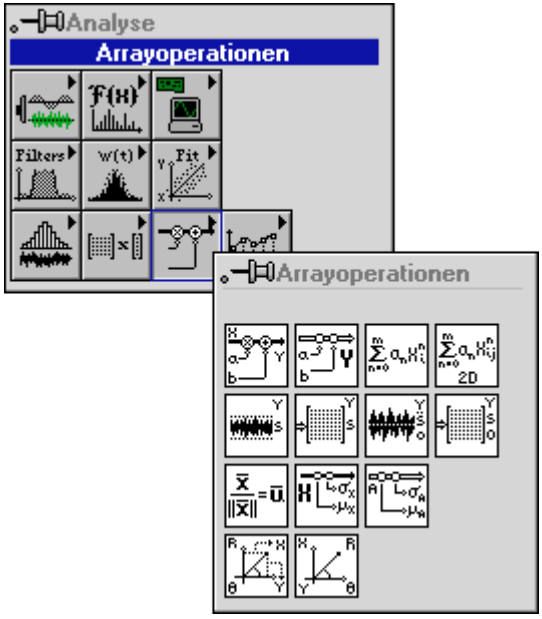
wobei *n* die Dimension der **Eingangs-Matrix** ist.

46

Arrayoperationen-VIs

Dieses Kapitel beschreibt die VIs, die gängige ein- und zweidimensionale numerische Arrayoperationen ausführen.

Die folgende Abbildung zeigt die **Arrayoperationen**-Palette, auf die über das Menü **Funktionen»Analyse»Arrayoperationen** zugegriffen werden kann.

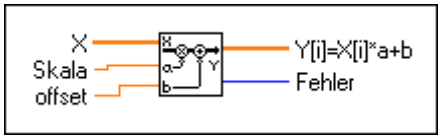


Beschreibungen der Arrayoperationen-VIs

Die folgenden Arrayoperations-VIs stehen zur Verfügung.

1D Linearentwicklung

Führt eine lineare Auswertung vom Eingangs-Array **X** aus.



Das Ausgangs-Array **Y[i] = X[i]*a + b** ist gegeben durch

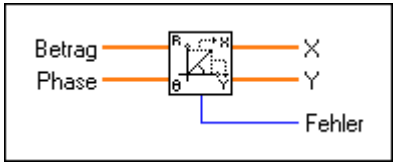
$$Y = aX + b,$$

wobei *a* die multiplikative **Skala**-Konstante und *b* die Additionskonstante **Offset** ist.

1D polar in karthesisch

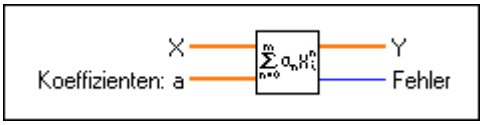
Konvertiert nach der folgenden Gleichung zwei Arrays aus polaren Koordinaten in zwei Arrays aus karthesischen Koordinaten:

x = **Magnitude** cos(**Phase**)
y = **Magnitude** sin(**Phase**).



1D Polynomialentwicklung

Führt eine polynomische Auswertung von **X** mit Hilfe von **Koeffizienten: a** durch.



Das Ausgangs-Array **Y** ist gegeben durch

$$Y = \sum_{n=0}^{m} a_n X^n,$$

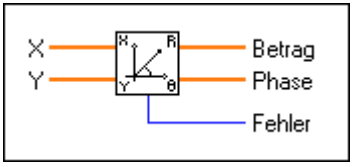
wobei *m* die polynomische Ordnung angibt.

1D karthesisch in polar

Konvertiert nach der folgenden Gleichung zwei Arrays aus karthesischen Koordinaten in zwei Arrays aus polaren Koordinaten:

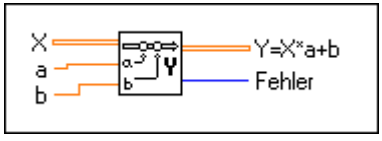
$$\text{Betrag} = \sqrt{x^2 + y^2}$$

$$\text{Phase} = \tan^{-1}\left(\frac{y}{x}\right).$$



2D Linearentwicklung

Führt eine lineare Auswertung des zweidimensionalen Eingangs-Arrays **X** aus.



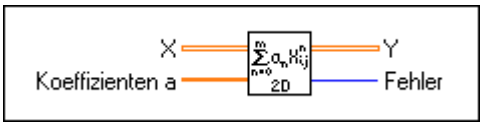
Das zweidimensionale Ausgangs-Array **Y = X*a + b** ist gegeben durch

$$Y = Xa + b,$$

wobei *a* die multiplikative Konstante und *b* die additive Konstante bezeichnet.

2D Polynomialentwicklung

Führt eine polynomische Auswertung vom zweidimensionalen Eingangs-Array **X** mit Hilfe von **Koeffizienten a durch**.



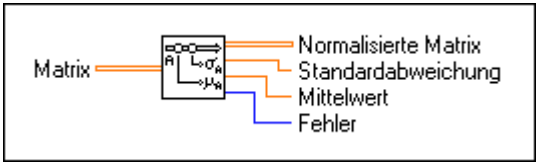
Das 2D-Ausgangs-Array **Y** ist gegeben durch

$$Y = \sum_{n=0}^{m} a_n X^n,$$

wobei *m* die polynomische Ordnung angibt.

Matrix normieren

Normalisiert die 2D-Eingangs-**Matrix** nach dem statistischen Profil (μ, σ), wobei μ der **Mittelwert** und σ die **Standardabweichung** ist, um eine **Normalisierte Matrix** zu erhalten, deren statistisches Profil (0,1) lautet.



Das VI erhält die **Normalisierte Matrix** durch

$$B = \frac{A - \mu}{\sigma},$$

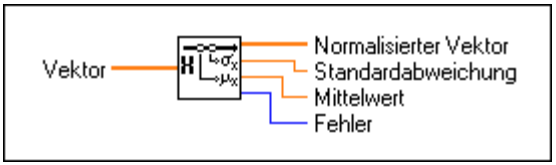
$$\mu = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} a_{ij}}{n \cdot m},$$

$$\sigma = \sqrt{\left| \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (a_{ij} - \mu)^2}{n \cdot m} \right|},$$

wobei *B* die 2D-Ausgangssequenz **Normalisierte Matrix** darstellt, *A* die 2D-Eingangssequenz **Matrix** mit *n* Zeilen und *m* Spalten darstellt und a_{ij} das Element von *A* in der *i*. Zeile und der *j*. Spalte ist.

Vektor normieren

Normalisiert den Eingangs-**Vektor** durch sein statistisches Profil (μ,σ), wobei μ der **Mittelwert** und σ die **Standardabweichung** ist, um einen **Normalisierten Vektor** zu erhalten, dessen statistisches Profil (0,1) lautet.



Das VI erhält **Normalisierter Vektor** durch

$$Y = \frac{X - \mu}{\sigma},$$

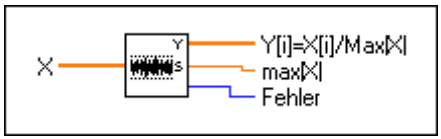
$$\mu = \frac{\sum_{i=0}^{n-1} x_i}{n},$$

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n-1} (x_i - \mu)^2}{n}},$$

wobei *Y* die Ausgangssequenz **Normalisierter Vektor** darstellt und *X* die Eingangssequenz **Vektor** von der Länge *n* darstellt und x_i das *i*. Element von *X* ist.

Schnell 1D skalieren

Findet den absoluten Höchstwert vom Eingangs-Array **X** und skaliert anschließend **X** nach diesem Wert.



Das Ausgangs-Array **Y[i] = X[i]/Max|X|** ist gegeben durch

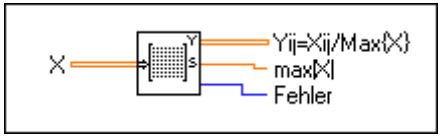
$$Y = \frac{X}{s},$$

wobei *s* der absolute Höchstwert in **X** ist.

Dieses VI kann zur Normierung von Sequenzen innerhalb des Wertebereichs [–1,1] eingesetzt werden. Dieses VI erweist sich als besonders nützlich, wenn die Matrix eine Matrix mit einem Mittelwert von Null ist.

Schnell 2D skalieren

Findet den absoluten Höchstwert vom Eingangs-Array **X** und skaliert anschließend **X** nach diesem Wert.



Das Ausgangs-Array **Yij = Xij/Max{X}** ist gegeben durch

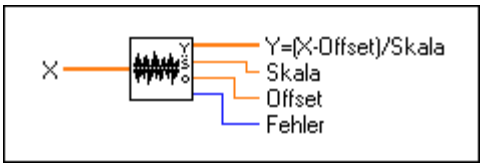
$$Y = \frac{X}{s},$$

wobei *s* den absoluten Höchstwert in **X** angibt.

Dieses VI kann zur Normierung von Sequenzen innerhalb des Wertebereichs [–1,1] eingesetzt werden. Dieses VI erweist sich als besonders nützlich, wenn die Matrix eine Matrix mit einem Mittelwert von Null ist.

1D skalieren

Findet **Skala** und **Offset** und skaliert anschließend das Eingangs-Array **X** nach diesen Werten.



Das Ausgangs-Array **Y** ist gegeben durch

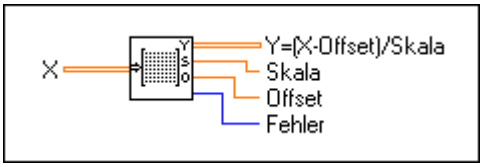
$$Y = \frac{X - Offset}{Skala},$$

Skala = 0,5(*max* – *min*) und **Offset** = *min* + **Skala**, wobei *max* den Höchstwert in **X** und *min* den Kleinstwert in **X** angibt.

Dieses VI kann zur Normierung jeder numerischen Sequenz eingesetzt werden mit der Versicherung, daß der Wertebereich der Ausgangssequenz [–1,1] ist.

2D skalieren

Findet **Skala** und **Offset** und skaliert anschließend **X** nach diesen Werten.



Das zweidimensionale Ausgangs-Array **Y = (X – Offset)/Skala** ist gegeben durch

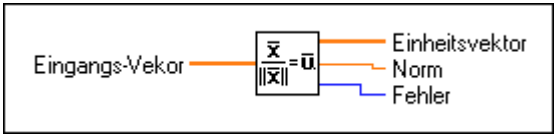
$$Y = \frac{X - Offset}{Skala},$$

Skala = 0,5(*max* – *min*) und **Offset** = *min* + 0,5 **Skala**, wobei *max* den Höchstwert in **X** und *min* den Kleinstwert in **X** darstellt.

Dieses VI kann zur Normierung jeder numerischen Sequenz eingesetzt werden mit der Versicherung, daß der Wertebereich der Ausgangssequenz [–1,1] ist.

Einheitsvektor

Findet die **Norm** vom **Eingangs-Vektor** und erhält seinen korrespondierenden **Einheitsvektor** durch Normierung vom ursprünglichen **Eingangs-Vektor** nach seiner **Norm**.



Es sei *X* der Eingang **Eingangs-Vektor**; **Norm** ist gegeben durch

$$\|X\| = \sqrt{x_0^2 + x_1^2 + \ldots + x_{n-1}^2},$$

wobei ||*X*|| die **Norm** ist und das VI den **Einheitsvektor,** *U*, durch

$$U = \frac{X}{\|X\|}$$

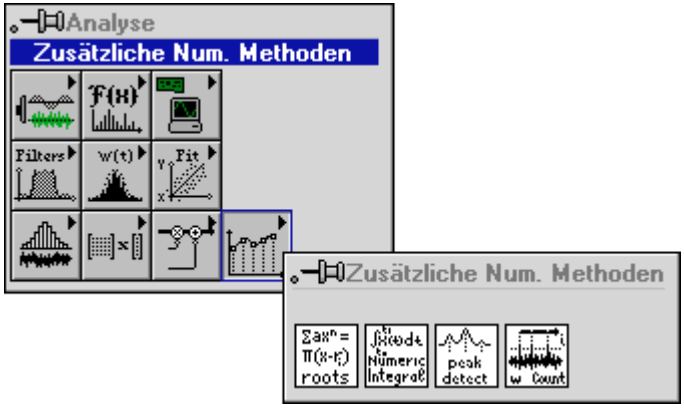
berechnet.

47

Zusätzliche Numerische Methoden-VIs

Dieses Kapitel beschreibt die VIs, die numerische Verfahren zum Wurzelziehen, zur numerischen Integration und zum Erkennen von Spitzenwerten verwenden.

Die folgende Abbildung zeigt die **Zusätzliche Num. Methoden**-Palette, auf die über das Menü **Funktionen»Analyse» Zusätzliche Num. Methoden** zugegriffen werden kann.

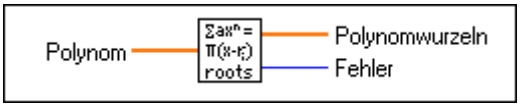


Beschreibungen der Zusätzliche Numerische Methoden-VIs

Die folgenden Zusätzliche Num. Methoden-VIs sind verfügbar.

Komplexe Nullstellen eines Polynoms

Findet die komplexen Wurzeln eines komplexen Polynoms.

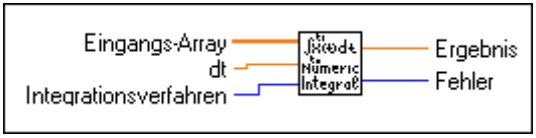


Dieses VI verwendet ein modifiziertes, komplexes Newtonsches Verfahren zur Ermittlung der *n* komplexen Wurzeln (von denen einige real sein könnten, ohne imaginären Teil) des allgemeinen komplexen Polynoms:

$$a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1} + a_nx^n.$$

Numerische Integration

Führt eine numerische Integration an dem Daten-Eingangs-Array aus und verwendet dabei eins von vier gebräuchlichen numerischen Integrationsverfahren.



☞ **Hinweis** ***Wenn die Anzahl der Punkte für ein bestimmtes ausgewähltes Verfahren keine Integralzahl von Partialsummen enthält, wird das Verfahren auf alle möglichen Punkte angewendet. Für die verbleibenden Punkte wird das nächstmögliche Verfahren niederer Ordnung verwendet. Die folgende Tabelle zeigt als Beispiel, wie das VI die verschiedenen Punkte anzahlen auswertet, wenn das Bode-Verfahren gewählt wurde:***

Punkteanzahl	Ausgeführte Partialauswertung
224	56 Bode
225	56 Bode, 1 Trapezförmig
226	56 Bode, 1 Simpsons’
227	56 Bode, 1 Simpsons’ 3/8
228	57 Bode

Wenn also 227 Punkte eingelesen werden und das Bode-Verfahren gewählt wurde, gelangt das VI zu dem Ergebnis, indem es 56 Partialauswertungen nach dem Bode-Verfahren ausführt und eine nach Simpsons’ 3/8-Methode.

Jedes der Verfahren ist vom Abtastintervall (**dt**) abhängig und berechnet das Integral, indem es zur Ausführung der Partialauswertungen, die von der Anzahl zusammenhängender Punkte abhängen, aufeinanderfolgende Applikationen einer grundlegenden Gleichung einsetzt. Die Anzahl der in jeder Partialauswertung verwendeten Punkte spiegelt die Ordnung des Verfahrens wider. Das Ergebnis stellt die Summe der aufeinanderfolgenden Partialauswertungen dar.

$$\text{Ergebnis} = \int_{t0}^{t1} f\,t\,dt \approx \sum_{j} \text{Partialsumme},$$

wobei *j* ein von der Punkteanzahl und dem Integrationsverfahren abhängiger Bereich ist.

Die grundlegenden Gleichungen von jeder Regel zur Berechnung von Partialsummen lauten in aufsteigender Folge:

Trapezförmig: $(x[i] + x[i+1])*dt$, $k = 1$

Simpsons’: $(x[2i] + 4x[2i+1] + x[2i+2])*dt/3$, $k = 2$

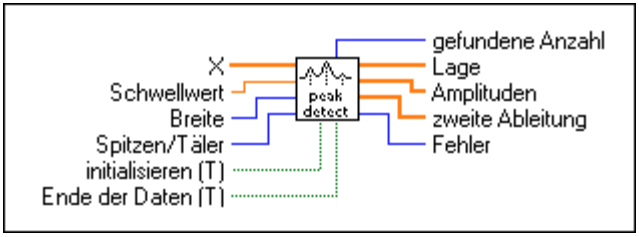
Simpsons’ 3/8: $(3x[3i] + 9x[3i+1] + 9x[3i+2] + 3x[3i+3]) * dt/8$, $k = 3$

Bode: $(14x[4i] + 64x[4i+1] + 24x[4i+2] + 64x[4i+3] + 14x[4i+4])*dt/45$, $k = 4$
für $i = 0, k, 2k, 3k, 4k,...,$ Integralteil von $[(N–1)/k]$

wobei *N* die Anzahl der Datenpunkte, *k* ein von dem Verfahren abhängiger Intergerwert und *x* das Eingangs-Array ist.

Peak Detektor

Ermittelt die Lage, die Amplitude und die zweite Ableitung von Spitzen oder Tälern im Eingangs-Array.



Der Datensatz kann in dieses VI als einzelnes Array oder in aufeinanderfolgenden Datenblöcken eingegeben werden.

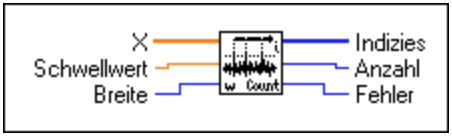
Dieses VI stützt sich auf einen Algorithmus, mit dem ein quadratisches Polynom an aufeinanderfolgende Datenpunktgruppen angepaßt wird. Die Anzahl der verwendeten Datenpunkte ist mit **Breite** vorgegeben.

Für jede Spitze oder jedes Tal wird die quadratische Anpassung gegen ein **Schwellwert**-Niveau geprüft: Spitzen mit Höhen unterhalb vom **Schwellwert** und Täler mit Tiefpunkten oberhalb vom **Schwellwert** werden ignoriert. **Spitzen/Täler** werden erst entdeckt, nachdem ungefähr die Hälfte der in **Breite** festgelegten Datenpunkte über die Stellen von **Spitzen/Täler** hinaus verarbeitet wurde. Diese Verzögerung hat nur für die Echtzeitverarbeitung Implikationen.

Das VI muß benachrichtigt werden, wenn der erste und der letzte Datenblock eingegeben wird, damit das VI die Daten initialisieren und dann intern an den Algorithmus zum Auffinden von Spitzen freigeben kann.

Schwellwert-Peak-Detektor

Untersucht die Eingangssequenz **X** auf gültige Spitzen und hält die **Anzahl** der aufgefundenen Spitzen fest und führt Aufzeichnungen über die **Indizes** der Punkte, die den **Schwellwert** in einer gültigen Spitze überschreiten. Eine Spitze gilt dort als gültig, wo die Elemente von **X** den **Schwellwert** überschreiten und dann zu einem Wert kleiner oder gleich dem **Schwellwert** zurückkehren und wo die Anzahl der **Schwellwert** überschreitenden Elemente mindestens so groß wie **Breite** ist.



Teil V

Kommunikations-VIs und Funktionen

Teil V, *Kommunikations-VIs und Funktionen*, beschreibt, wie LabVIEW Netzverbund und Kommunikationen innerhalb von Anwendungen handhabt. Des weiteren werden die Kommunikations-VIs und -Funktionen vorgestellt. Dieser Teil enthält die folgenden Kapitel:

- Kapitel 48, *TCP-VIs*, beschreibt das Internet Protocol (IP), das Transmission Control Protocol (TCP) sowie Internet-Adressen und die LabVIEW-TCP-VIs. Siehe Kapitel 21, *TCP und UDP*, des *LabVIEW Benutzerhandbuchs.* Dort finden Sie einen Überblick über TCP/IP sowie Beispiele von TCP-Client-/Server -Anwendungen.
- Kapitel 49, *UDP-VIs*, beschreibt die VIs, die Sie mit dem User Datagram Protocol (UDP), einem Protokoll in der TCP-/IP-Seite zur Kommunikation über ein einzelnes Netzwerk oder einem miteinander verbundenen Satz von Netzwerken benutzen.
- Kapitel 50, *DDE-VIs*, beschreibt die LabVIEW-VIs für Dynamic Data Exchange (DDE) für Windows 3.1, Windows 95 und Windows NT. Diese VIs führen DDE-Funktionen zur gemeinsamen Benutzung von Daten mit anderen Anwendungen, die DDE-Verbindungen akzeptieren, durch.
- Kapitel 51, *ActiveX-Automationsfunktionen*, beschreibt die Funktionen zur Unterstützung von ActiveX-Automation. Diese Funktionen ermöglichen es anderen mit ActiveX aktivierten Anwendungen, wie Microsoft Excel, Eigenschaften und Methoden von LabVIEW und einzelnen VIs anzufordern.

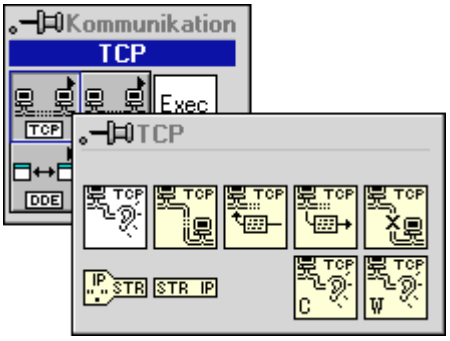
- Kapitel 52, *AppleEvent-VIs*, beschreibt die LabVIEW-VIs für AppleEvents, eine Form der Kommunikation zwischen Anwendungen (IAC, interapplication communication), mit deren Hilfe Macintosh-Anwendungen miteinander kommunizieren können.
- Kapitel 53, *Programm-zu-Programm- Kommunikations-VIs*, beschreibt die LabVIEW-VIs für Programm-zu-Programm-Kommunikation (PPC; program-to-program-communication), eine Low-Level-Form der Apple-Kommunikationen zwischen Anwendungen (IAC; interapplication communiation), durch die Macintosh-Anwendungen Datenblöcke senden und empfangen.

48

TCP-VIs

Dieses Kapitel beschreibt das Internet Protokoll (IP), das Transmission-Control-Protokoll (TCP) und Internetadressen sowie die TCP-VIs von LabVIEW. Lesen Sie im *LabVIEW Benutzerhandbuch* das Kapitel 21, *TCP und UDP*, um einen Überblick über TCP/IP zu erhalten und Beispiele von TCP-Client/Server-Anwendungen zu finden.

Die folgende Abbildung zeigt die **TCP** Palette, auf die über das Menü **Funktionen»Kommunikation»TCP** zugegriffen werden kann.



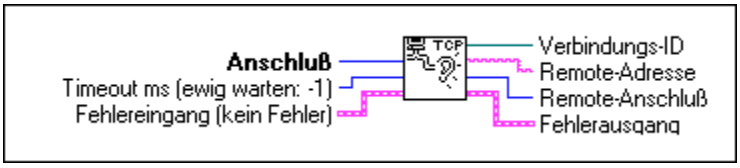
Beispiele für die Verwendung der TCP-VIs finden Sie unter den Beispielen in `examples\comm\tcpex.llb`.

Beschreibung des TCP-VIs

Das folgende TCP-VI steht zur Verfügung.

TCP hören

Stellt einen Listener (Empfänger) her und wartet auf eine zugelassene TCP-Verbindung an dem angegebenen Anschluß.



Wenn ein Hören an einem bestimmten Anschluß beginnt, kann kein zweites VI TCP hören am selben Anschluß hören. Nehmen Sie z.B. an, ein VI besitzt auf seinem Blockdiagramm zwei TCP hören-VIs. Wenn das erste VI TCP hören am Anschluß 2222 das Hören aufnimmt, schlagen alle Hörversuche eines zweiten VIs TCP hören am Anschluß 2222 fehl.

TCP/IP-Funktionen

Zusätzlich zu den vorhandenen Funktionen sind einige der TCP/IP-VIs jetzt Funktionen. Die folgenden VIs stellen in LabVIEW 5.0 nun Funktionen dar:

- IP in String
- String in IP
- TCP Verbindung öffnen
- TCP Listener erstellen
- TCP Auf Listener warten
- TCP Schreiben
- TCP Lesen
- TCP Verbindung schließen

Das VI TCP hören ist in LabVIEW 5.0 immer noch ein VI, weil seine Funktionalität von den Funktionen TCP Listener erstellen und TCP Auf Listener warten dupliziert wird.

Die Funktionen TCP Lesen, TCP Schreiben und TCP Auf Listener warten verfügen über neue Eigenschaften. Der Parameter **Daten ein** der Funktion TCP Schreiben akzeptiert jetzt Arrays aus Bytes. Die Funktion TCP Lesen verfügt über einen neuen Eingang, **Modus**, der deren Arbeitsweise beeinflußt. Die vier Modi sind Standard, Gepuffert, CLRF (Wagenrücklauf mit anschließendem Zeilenvorschub) und Sofort. TCP Auf Listener warten verfügt über einen neuen Eingang, **Auflösen der Remote-Adresse**, der angibt, ob die Remote-Adresse aufgelöst oder in Punkt-Notation belassen werden soll.

Der Modus Standard weist dasselbe Verhalten wie in früheren Versionen von LabVIEW auf. Der Modus Gepuffert ist eine Lesevorgang, in dem alles oder gar nichts gelesen wird. Wenn beim Ablauf der Zeitbegrenzung die angeforderten Bytes nicht eingetroffen sind, werden keine Bytes ausgegeben. Die nicht ausgegebenen Bytes werden für spätere Leseversuche aufgehoben. Der Modus CRLF liest, bis ein Wagenrücklauf und ein Zeilenvorschub im Eingangsstrom gefunden werden. Es muß aber dennoch eine maximale Lesegröße festgelegt werden. Wenn kein CRLF innerhalb der festgelegten Größe gefunden wird, wird nichts ausgegeben. Wenn die Zeitbegrenzung abgelaufen ist und kein CRLF gefunden wurde, wird nichts ausgegeben. Der Modus Sofort legt fest, daß unmittelbar nach Lesen von Bytes eine Ausgabe erfolgt.

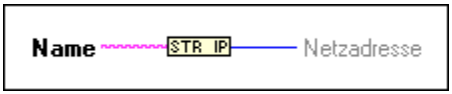
Die folgenden TCP/IP-Funktionen sind verfügbar.

IP in String

Konvertiert eine IP-Netzwerkadresse in einen String.

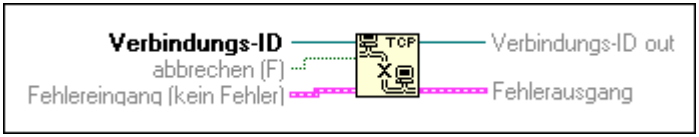
String in IP

Wandelt einen String in eine IP-Netzwerkadresse um.



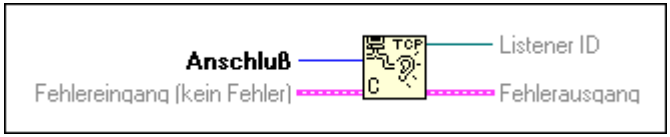
TCP Verbindung schließen

Schließt die mit der **Verbindungs-ID** verknüpfte Verbindung.



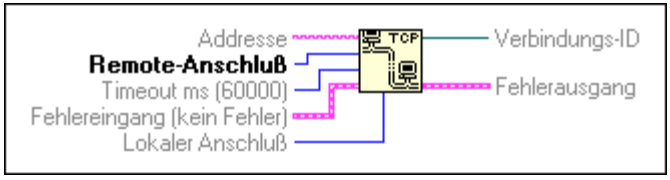
TCP Listener erstellen

Erstellt einen Listener für eine TCP-Verbindung.



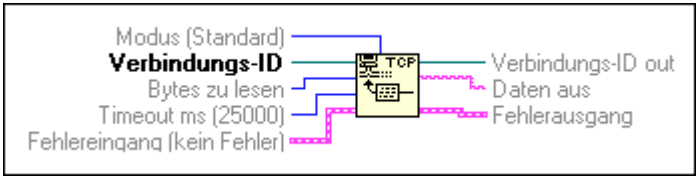
TCP Verbindung öffnen

Versucht, eine TCP-Verbindung mit der angegebenen Adresse und dem angegebenen Anschluß einzurichten.



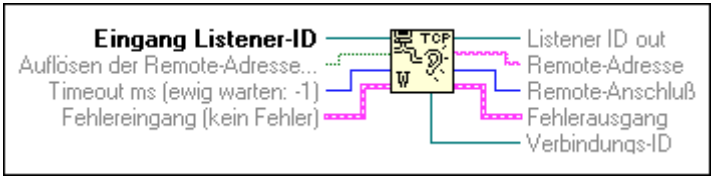
TCP Lesen

Empfängt von der angegebenen TCP-Verbindung Bytes bis zu **Bytes zu lesen** und gibt die Ergebnisse in **Daten aus** aus.



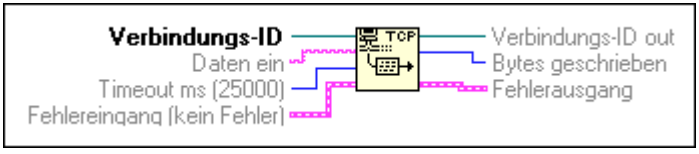
TCP Auf Listener warten

Wartet an dem angegebenen Anschluß auf eine akzeptierte TCP-Verbindung.



TCP Schreiben

Schreibt den String **Daten ein** zur angegebenen TCP-Verbindung.

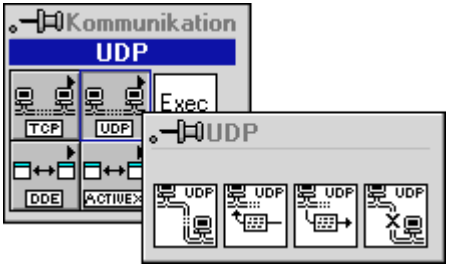


49

UDP-VIs

Dieses Kapitel beschreibt einen Satz von VIs, die mit dem User Datagram Protocol (UDP = Benutzer-Datagramm-Protokoll) verwendet werden können, einem Protokoll in der TCP/IP-Suite zur Kommunikation in einem einzelnen Netzwerk oder einem Satz miteinander verbundener Netzwerke.

Die folgende Abbildung zeigt die **UDP**-Palette, die über das Menü **Funktionen»Kommunikation»UDP** aufgerufen wird.

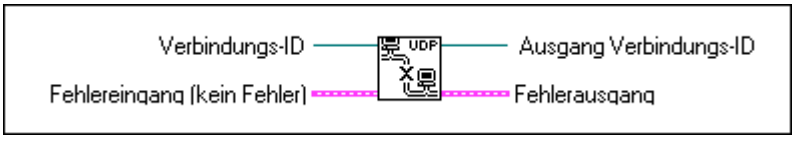


Beschreibungen der UDP-VIs

Die folgenden UDP-VIs stehen zur Verfügung.

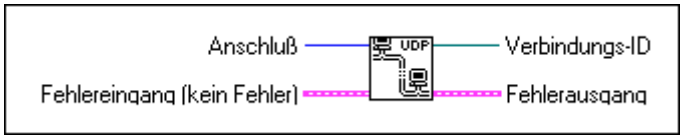
UDP schließen

Schließt die durch **Verbindungs-ID** angegebene UD-Verbindung.



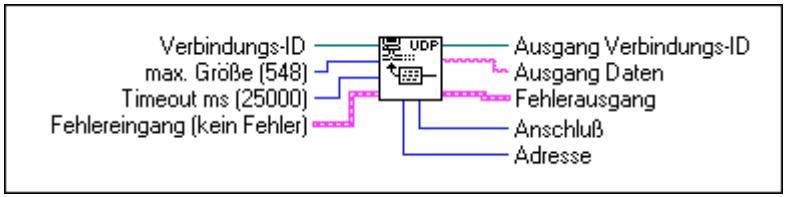
UDP öffnen

Versucht an dem angegebenen Anschluß eine UDP-Verbindung zu öffnen. Die **Verbindungs-ID** ist ein dunkles Token, das in allen nachfolgenden, mit dieser Verbindung in Zusammenhang stehenden Operationen verwendet wird.



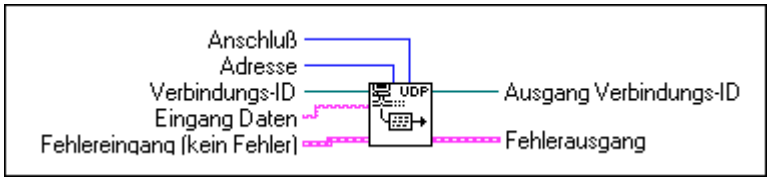
UDP lesen

Gibt in dem String **Daten aus** ein Datagramm aus, das von der mit **Verbindungs-ID** angegebenen UDP-Verbindung empfangen wurde.



UDP schreiben

Schreibt an die mit **Adresse** und **Anschluß** vorgegebene Remote-UDP- Verbindung den Datenstring **Daten ein**.

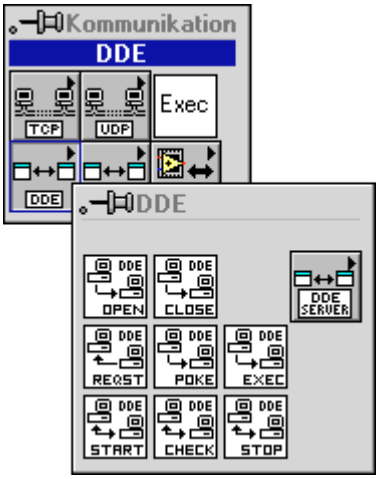


50

DDE-VIs

Dieses Kapitel beschreibt die LabVIEW-VIs für Dynamic Data Exchange (DDE = Dynamischer Datenaustausch) unter Windows 3.1, Windows 95 und Windows NT. Diese VIs führen DDE-Funktionen zur gemeinsamen Benutzung von Daten mit anderen Anwendungen aus, die DDE-Verbindungen zulassen.

Die folgende Abbildung zeigt die **DDE**-Palette, auf die über das Menü **Funktionen»Kommunikation»DDE** zugegriffen wird.



Die **DDE**-Palette umfaßt die **DDE Server**-Unterpalette.

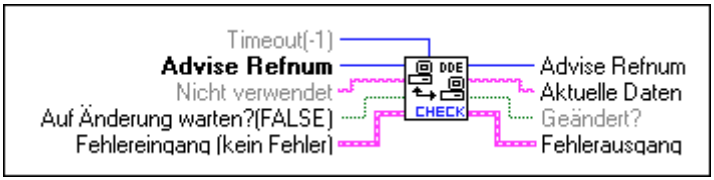
Beispiele für die Verwendung der DDE-VIs finden Sie unter den Beispielen in `examples\comm\DDEexamp.llb`.

Beschreibungen der DDE Client-VIs

Die folgenden DDE Client-VIs stehen zur Verfügung.

DDE Advise überprüfen

Überprüft einen zuvor durch DDE Advise starten eingerichteten Advise-Wert.

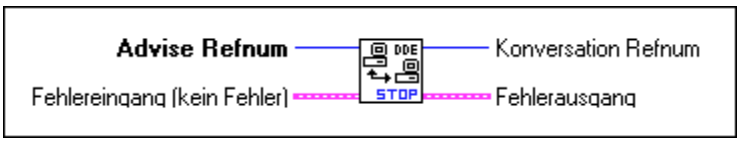


DDE Advise starten

Leitet einen Advise-Link ein.

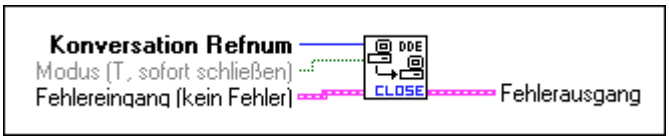
DDE Advise stopp

Kündigt einen zuvor durch DDE Advise starten eingerichteten Advise-Link.



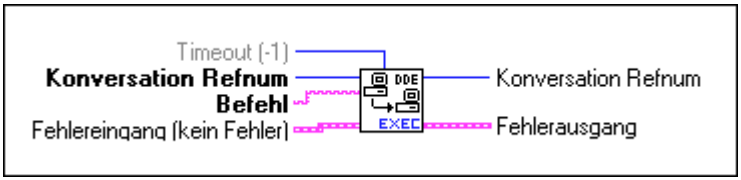
DDE Verbindung schließen

Schließt eine DDE-Verbindung.



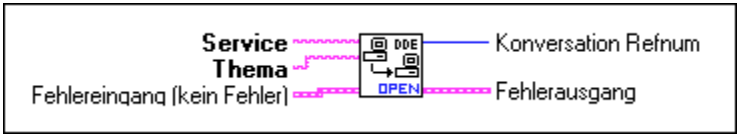
DDE ausführen

Weist den DDE-Server zur Ausführung von **Befehl** an.



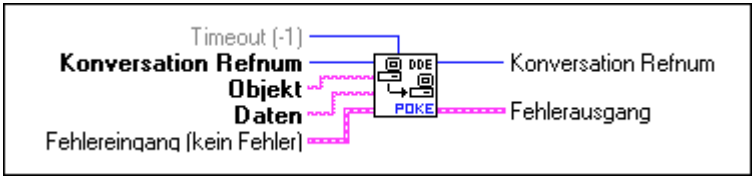
DDE Verbindung öffnen

Richtet eine Verbindung zwischen LabVIEW und einer anderen Anwendung ein. Dieses VI muß vor dem Einsatz jedes anderen DDE-VIs (außer Server-VIs) aufgerufen werden.



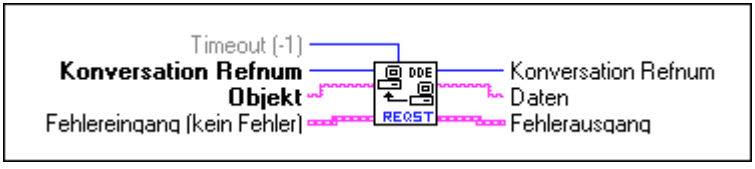
DDE Poke

Weist den DDE-Server an, den Wert von **Daten** in **Objekt** einzufügen.



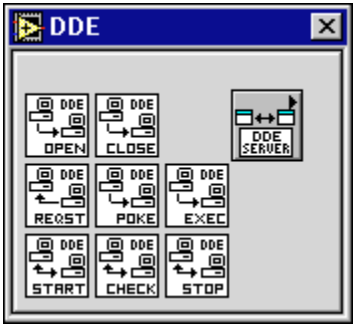
DDE Anforderung

Leitet einen DDE-Mitteilungsaustausch ein, um den aktuellen Wert von **Objekt** zu erhalten.



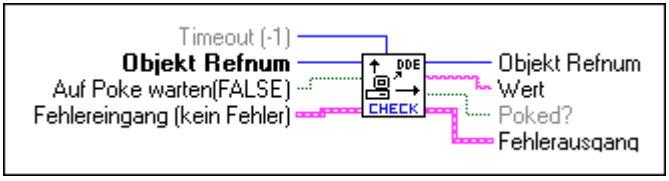
Beschreibungen der DDE-Server-VIs

Die DDE-Server-Funktionen können über das Menü **Funktionen»Kommunikation» DDE»DDE Server** aufgerufen werden.



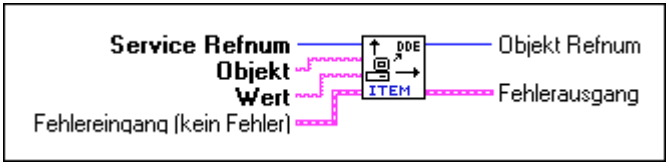
DDE-Server Elemente überprüfen

Gibt den Wert eines zuvor definierten DDE-Objekts zurück.



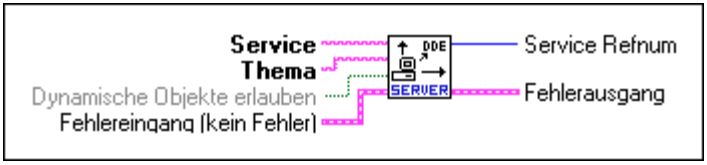
DDE-Server Element registrieren

Richtet ein DDE-Objekt für die durch **Service-Refnum** angegebene Dienstleistung ein.



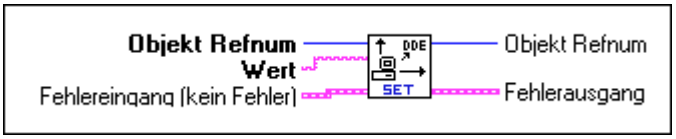
DDE-Server Service registrieren

Richtet einen DDE-Service ein, mit dem sich Clients verbinden können.



DDE-Server Element einstellen

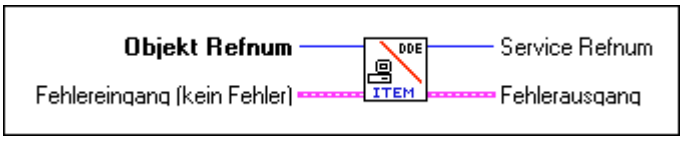
Setzt den Wert eines zuvor definierten DDE-Objekts.



DDE-Server Elementregistrierung löschen

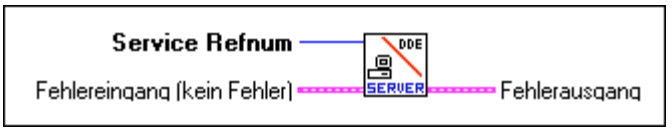
Entfernt das angegebene Objekt von seinem Service.

☞ **Hinweis** ***DDE-Clients können nach Abschluß dieses VIs auf das Objekt nicht länger zugreifen.***



DDE-Server Serviceregistrierung löschen

Entfernt den angegebenen Service. DDE-Clients können nach Ausführung dieses VIs nicht mehr mit diesem Service Verbindung aufnehmen, und alle stehenden Verbindungen werden geschlossen.

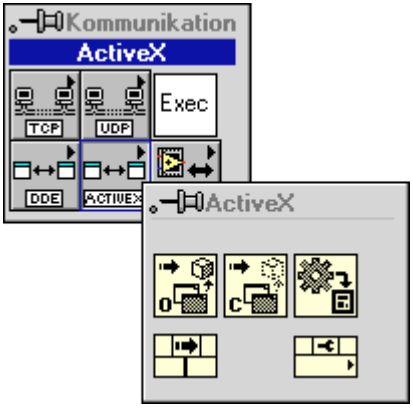


51

ActiveX-Automationsfunktionen

Dieses Kapitel beschreibt die Funktionen zur Unterstützung von ActiveX-Automation. Diese Funktionen erlauben anderen ActiveX-fähigen Anwendungen wie Microsoft Excel, Eigenschaften und Methoden von LabVIEW und individuellen VIs anzufordern.

Die ActiveX-Automationsfunktionen werden über das Menü **Funktionen»Kommunikationen»ActiveX/OLE** aufgerufen.



Die Palette **ActiveX/OLE** beinhaltet die folgenden Funktionen:

- Automation öffnen
- Automation schließen
- Knoten aufrufen
- Eigenschaftsknoten

Sie beinhaltet darüber hinaus die ActiveX-Variante zur G-Datenfunktion. Weitere Informationen zu dieser Funktion finden Sie weiter hinten in diesem Kapitel unter *Datenumwandlungsfunktion*.

National Instruments unterstützt die alten Funktionen, die Kompatibilitätsfunktionen verwenden, doch alle neuen Anwendungen sollten so erstellt sein, daß sie neue Funktionen verwenden. Die folgende Tabelle zeigt die alten Funktionszuweisungen gegenüber den neuen.

Tabelle 51-1. Neue und alte ActiveX-Automationsfunktionen

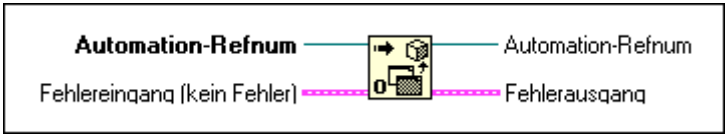
Neue ActiveX-Funktionen	Alte ActiveX-Funktionen
Automation öffnen	Refnum erstellen
Automation schließen	Refnum freigeben
Methodenknoten	Methode ausführen
Eigenschaftsknoten	Eigenschaften erhalten Eigenschaften einstellen

Beschreibungen der ActiveX-Automationsfunktionen

Die folgenden Funktionen sind verfügbar.

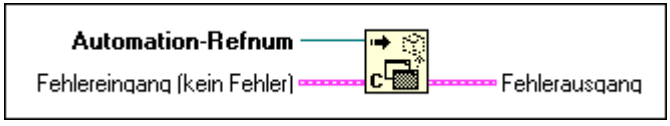
Automation-Refnum öffnen

Öffnet eine Automation-Referenznummer, die auf ein bestimmtes ActiveX-Automation-Objekt verweist. Die Objektklasse wird im Popup-Menü der Funktion unter **ActiveX-Klasse auswählen** ausgewählt. Nach dem Öffnen einer Referenznummer kann sie an andere ActiveX-Funktionen weitergegeben werden. Es sollten nur erstellbare Klassen als Eingang für die Funktion ausgewählt werden.



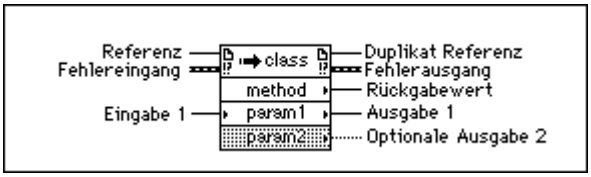
Automation-Refnum schließen

Schließt eine Automation-Referenznummer. Stellen Sie sicher, daß Sie jede offene Automation-Referenznummer schließen, wenn sie nicht länger geöffnet sein muß.



Methodenknoten

Ruft eine Methode oder Aktion an einem ActiveX-Objekt auf. Um ein Objekt der ActiveX-Klasse auszuwählen, müssen Sie das Popup-Menü aufrufen und **Auswählen» ActiveX-Klasse** auswählen oder eine Automation-Refnum mit dem Eingang verbinden. Um eine auf das Objekt bezogene Methode auszuwählen, müssen Sie das Popup-Menü des zweiten Abschnitts des Knotens (“Methode” im Diagramm) aufrufen und **Methoden** auswählen. Nach Auswahl der Methode erscheinen die verknüpften Parameter darunter. Parameterwerte können gelesen oder beschrieben werden. Bei den Parametern mit weißem Hintergrund handelt es sich um freie Eingänge und bei den mit grauem Hintergrund um wahlfreie Eingänge.



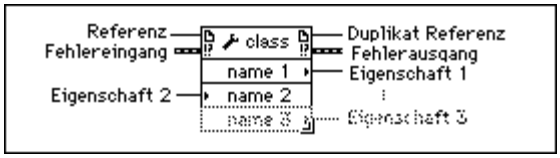
Wenn es sich bei den Eingangs-Parametern um Variant-Datentypen handelt, können G-Datentypen eingegeben werden, und diese werden automatisch in Variant-Datentypen umgewandelt und mit einem Formatumwandlungspunkt kenntlich gemacht. Wenn es sich beim Ausgang um einen Variant-Typen handelt, muß ggf. die Funktion ActiveX-Datentyp in G-Funktion zur Umwandlung in den G-Typ eingesetzt werden.

Eigenschaftsknoten

Setzt (schreibt) und erhält (liest) Eigenschaftsinformationen von ActiveX-Objekten. Um ein Objekt der ActiveX-Klasse auszuwählen, müssen Sie das Popup-Menü aufrufen und **Auswählen»ActiveX-Klasse** auswählen oder eine Automation-Refnum mit dem Eingang verbinden. Um eine auf das Objekt bezogene Eigenschaft auszuwählen, müssen Sie das Popup-Menü der zweiten Zeile des Knotens aufrufen und **Eigenschaften** auswählen. Um Eigenschaftsinformationen zu setzen, müssen Sie das Popup-Menü aufrufen und **In Schreiben ändern** aufrufen; um Eigenschaftsinformationen zu erhalten, müssen Sie das Popup-Menü aufrufen und **In Lesen ändern** auswählen. Einige Eigenschaften lassen sich entweder nur lesen oder nur schreiben, wodurch **In Schreiben ändern** oder **In Lesen ändern** im Popup-Menü abgedunkelt erscheinen.

Der Eigenschaftsknoten arbeitet auf dieselbe Weise wie der Attributknoten. Wenn Objekte zum Knoten hinzugefügt werden sollen, müssen Sie das Popup-Menü aufrufen und **Element hinzufügen** auswählen oder den Knoten anklicken und auseinanderziehen, um die Anzahl der Objekte im Knoten zu erhöhen. Die Eigenschaften werden in der Reihenfolge von oben nach unten geändert. Achten Sie darauf, daß Sie, wenn sich der kleine Richtungspfeil

auf einer Eigenschaft auf der linken Seite befindet, den Wert der Eigenschaft einstellen. Wenn sich der kleine Richtungspfeil auf einer Eigenschaft auf der rechten Seite befindet, erhalten Sie den Eigenschaftswert.



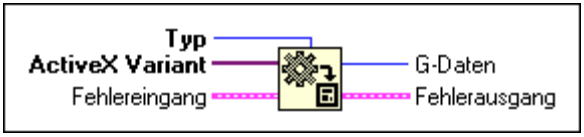
Wenn die zu schreibende Eigenschaft dem ActiveX-Datentyp angehört, kann sie in die G-Datentypen eingegeben werden und wird automatisch in Variant-Datentypen umgewandelt, was durch einen Formatumwandlungspunkt angezeigt wird. Wenn die Eigenschaft dem ActiveX-Datentyp angehört, muß ggf. die Funktion ActiveX-Datentyp in G-Datentyp eingesetzt werden, um sie in den G-Datentyp umzuwandeln.

Datenumwandlungsfunktion

Einige Anwendungen stellen ActiveX-Daten in der Form eines selbstbeschreibenden Datentyps bereit, dem sogenannten *ActiveX* oder *OLE-Variant*. Um die Daten einzusehen oder zu verarbeiten, müssen sie in einen entsprechenden G-Datentyp umgewandelt werden. Um Daten der ActiveX-Variante in G-Datentypen umzuwandeln, muß die unten beschriebene Funktion ActiveX-Datentyp in G-Datentyp verwendet werden.

ActiveX-Datentyp in G-Datentyp

Konvertiert ActiveX-Daten in Daten um, die in LabVIEW angezeigt werden können.



52

AppleEvent-VIs

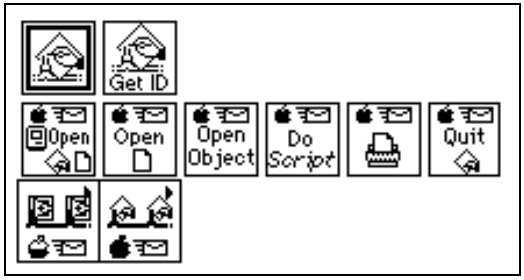
☞ **Hinweis** ***Dieses Kapitel ist nur für die LabVIEW-Benutzer von Belang, die LabVIEW auf der Plattform Macintosh System 7 einsetzen.***

Dieses Kapitel beschreibt die LabVIEW VIs für AppleEvents, einer Form von Interapplication Communication (IAC = Kommunikation von Anwendungen untereinander), durch die Macintosh-Anwendungen miteinander kommunizieren können. Außerdem kann LabVIEW mit einer Low-Level-Form von IAC verwendet werden, die als Program-to-Program Communication (PPC = Programm-zu-Programm-Kommunikation) bezeichnet wird.

AppleEvents stellen eine High-Level-Kommunikationsmethode dar, in der Anwendungen mit Nachrichten andere Anwendungen zur Ausführung von Aktionen oder zur Ausgabe von Informationen auffordern. Eine Anwendung kann diese Nachrichten an sich selbst schicken, an andere Anwendungen auf demselben Computer oder an andere Anwendungen irgendwo in einem Netzwerk. Apple hat ein großes *Vokabular* für die Nachrichten definiert, um zur Standardisierung dieser Form der Kommunikation der Anwendungen miteinander beizutragen. *Wörter* aus diesem Vokabular können zur Bildung komplexer Nachrichten kombiniert werden. Einzelheiten dieses Vokabulars sind in der *AppleEvents-Registrierung*, einer von der Apple Computer Inc. erhältlichen Dokumentation, beschrieben. Die Mehrzahl der für System 7 geschriebenen Anwendungen, LabVIEW eingeschlossen, reagieren auf eine Teilmenge von AppleEvents.

PPC stellt eine Low-Level-Form von IAC dar, durch die Anwendungen Datenblöcke senden und empfangen. PPC liefert eine höhere Leistungsfähigkeit als AppleEvents, weil der Verwaltungsaufwand zur Übertragung der Informationen niedriger ist. Da jedoch PPC die Art der übertragbaren Informationen nicht definiert, wird PPC von vielen Anwendungen nicht unterstützt. PPC ist der beste Weg, große Informationsmengen zwischen Anwendungen, die PPC unterstützen, hin und her zu senden. Lesen Sie Kapitel 53, *Programm-zu-Programm-Kommunikations-VIs*, für weitere Informationen über PPC.

Die folgende Abbildung zeigt die Palette **AppleEvent-VI**, die über das Menü **Funktionen»Kommunikation»AppleEvent** aufgerufen wird.



☞ **Hinweis** ***Damit Anwendungen über IAC kommunizieren können, muß der Computer System 7.0 oder höher verwenden und Program-Linking aktiviert sein.***

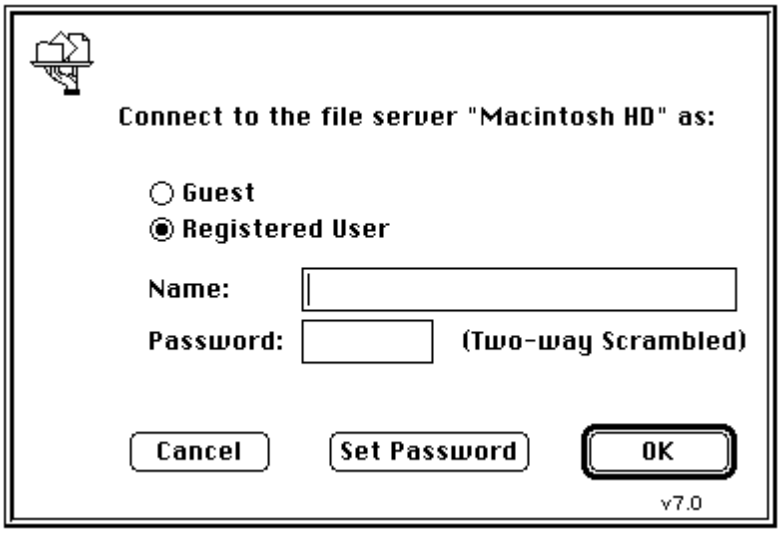
Beispiele für die Verwendung der AppleEvent-VIs finden Sie unter den Beispielen in `examples\comm\AE Examples.llb`.

Allgemeines Verhalten der AppleEvent-VIs

Beim Senden eines AppleEvents muß die *Ziel*-Anwendung für das Event angegeben werden. Damit ein AppleEvents empfangen werden kann, muß die Anwendung geöffnet sein. Zum Öffnen einer Anwendung können Sie das VI AESend Finder Open einsetzen.

Das Dialogfeld Benutzerkennung

Vor dem Senden eines AppleEvents an einen anderen Computer, müssen Sie das Kontrollfeld-Dienstprogramm Benutzer & Gruppen einsetzen, um für sich einen Benutzernamen und ein Paßwort einzurichten. Beim erstmaligen Senden eines AppleEvents an eine Anwendung oder den Finder auf dem Zielcomputer fordert Sie ein Dialogfeld zur Eingabe Ihres Namens und Paßwortes auf. Das System vergleicht diese Informationen mit der Konfiguration vom Kontrollfeld-Dienstprogramm Benutzer & Gruppen auf dem Zielcomputer.



Das aktuelle Design des AppleEvent-Managers schließt keine programmierte Methode für das Umgehen dieses Dialogfelds ein; diese Tatsache sollte beim Entwerfen von VIs, die IAC verwenden, in Betracht gezogen werden. Sie können z.B. keinen unbeaufsichtigten, entfernten Computer anweisen, ein AppleEvent an einen dritten Computer zu schicken; jemand muß die Nutzerinformationen im Dialogfeld Benutzerkennung eingeben, das auf diesem entfernten Computer erscheint. Das VI PPC erlaubt *nicht-autorisierte* Sitzungen, wenn Gastzugriff auf dem Computer, mit dem kommuniziert werden soll, aktiviert ist. Sie könnten daher u.U. die PPC-VIs für bestimmte LabVIEW-zu-LabVIEW-Kommunikationen nützlicher finden.

Ziel-ID

Die Mehrzahl der AppleEvents sendenden VIs benötigen eine Beschreibung der das AppleEvent erhaltenen Zielanwendung. Bei der **Ziel-ID** handelt es sich um einen komplexen Cluster aus Informationen, die von Apple Computer Inc. definiert wurden und die die Zielanwendung und dessen Speicherplatz angeben. Die folgenden VIs erzeugen **Ziel-ID**, so daß Sie diesen Cluster nicht auf dem Diagramm zu erstellen brauchen.

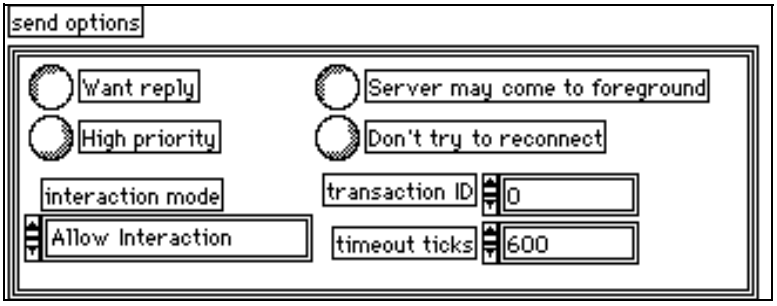
- PPC-Browser erstellt die **Ziel-ID** durch Anzeigen eines Dialogfeldes, in dem Sie AppleEvent-fähige Anwendungen auf dem Netzwerk interaktiv auswählen.
- Das VI Ziel-ID erhalten erstellt die **Ziel-ID** programmgemäß anhand des Anwendungsnamens und Netzwerkspeicherplatzes.

Diese VIs werden in Einzelheiten im Abschnitt *Beschreibungen der Zielauswahl-VIs*, in diesem Kapitel besprochen.

Der Cluster **Ziel-ID** muß nur eingesehen werden, wenn Zielinformationen von einem VI an ein anderes weitergegeben werden sollen. Um ein **Ziel-ID**-Cluster für das Frontpanel eines VIs zu erstellen, daß Zielinformationen an ein anderes VI oder ein AppleEvent weitergibt, kann der **Ziel-ID**-Cluster von dem Frontpanel eines der AppleEvent-VIs kopiert werden.

Optionen senden

Viele der VIs, die ein AppleEvent senden, besitzen einen Eingang **Optionen senden**, welcher angibt, ob die Zielanwendung mit dem Benutzer und der Länge der Zeitbegrenzung des AppleEvents in Wechselwirkung steht.



Beschreibungen der Zielauswahl-VIs

Die folgenden Zielauswahl-VIs sind verfügbar.

Ziel-ID erhalten

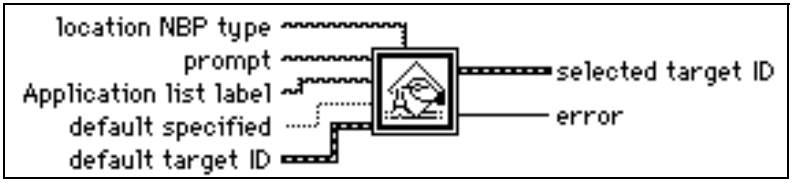
Gibt eine Ziel-ID für eine angegebene Anwendung anhand ihres Namens und Speicherplatzes aus. Sie geben entweder den Anwendungsnamen und Speicherplatz an, oder das VI sucht das gesamte Netzwerk nach der Anwendung ab.

Die folgende Tabelle faßt die Operation **Gesamtes Netzwerk absuchen**, **Zone** und **Server** zusammen:

Um die folgenden Speicherplätze abzusuchen:	sind die folgenden Parameter einzusetzen:
Den aktuellen Computer	**Zone** und **Server** dürfen nicht verbunden sein. **Gesamtes Netzwerk absuchen** muß FALSE sein.
Einen speziellen Computer im Netzwerk	**Zone** und **Server** müssen die Zone und den Server des Zielcomputers angeben. (Wenn **Zone** nicht verbunden wird, sucht das VI die aktuelle Zone ab.) **Gesamtes Netzwerk absuchen** muß auf FALSE stehen.
Eine bestimmte Zone	**Zone** muß die abzusuchende Zone angeben. **Server** darf nicht verbunden sein. **Gesamtes Netzwerk absuchen** muß auf FALSE stehen.
Das gesamte Netzwerk	**Gesamtes Netzwerk absuchen** muß auf TRUE stehen. Das VI ignoriert **Zone** und **Server**.

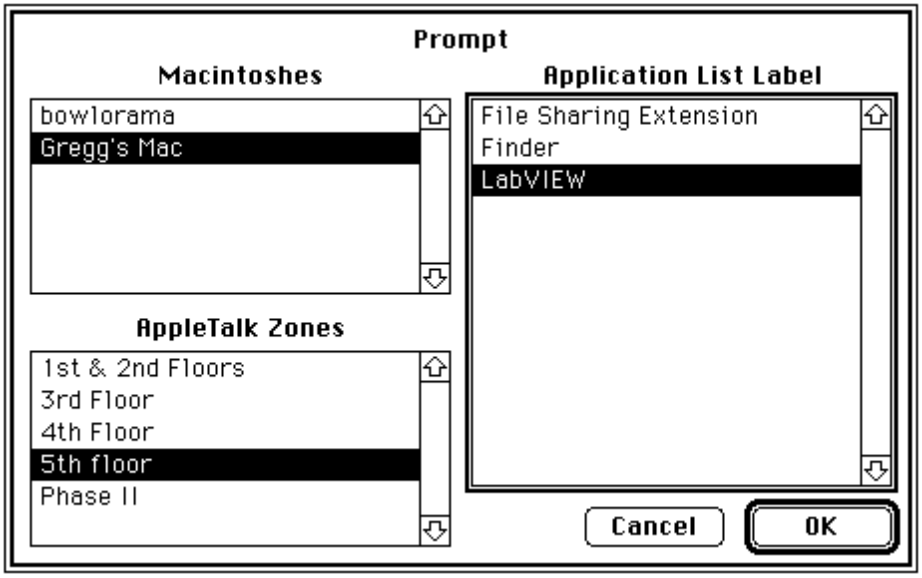
PPC-Browser

Ruft das Dialogfeld vom PPC-Browser zur Auswahl einer Anwendung im Netzwerk oder auf demselben Computer auf.



Dieses Standarddialogfeld von Macintosh kann zur Auswahl einer Zone (Unterteilung) des Netzwerks, eines Objektes in dieser Zone (im System 7 ist dies normalerweise der Name

eines Computers einer Person) und einer Anwendung verwendet werden. Das VI gibt dann den Cluster **Ziel-ID** aus.

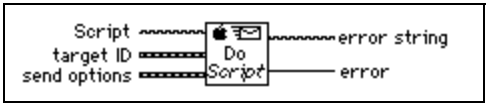


Beschreibungen der AppleEvent-VIs

Die folgenden AppleEvent-VIs sind verfügbar.

Sendet AE Auftragsskript

Sendet das AppleEvent Auftragsskript (Do Script AppleEvent) an eine vorgegebene Zielanwendung.



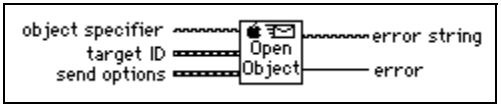
Sendet AE Finder-öffnen

Sendet das AppleEvent zum Öffnen angegebener Anwendungen oder Dokumente für den Finder von System 7 auf dem angegeben Computer.

☞ **Hinweis** ***Apple könnte u.U. den Satz AppleEvents, auf den der Finder reagiert, ändern, damit diese enger mit dem für AppleEvents gesetzten Standard übereinstimmen. Als Konsequenz könnte es passieren, daß das vom sendet AE Finder-öffnen gesendete AppleEvent in zukünftigen Versionen der Systemsoftware nicht länger unterstützt wird.***

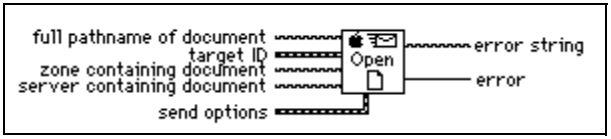
Sendet AE Öffnen

Sendet das AppleEvent Öffnen an eine vorgegebene Zielanwendung.



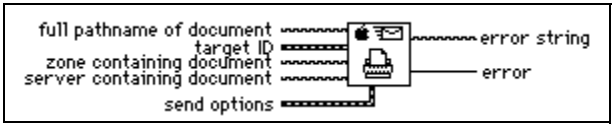
Sendet AE Dokument-öffnen

Sendet das AppleEvent Dokument-öffnen an die angegebene Anwendung, d.h., weist die Anwendung an, das vorgegebene Dokument zu öffnen.



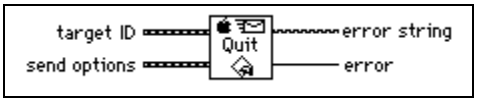
Sendet AE Dokument-drucken

Sendet das AppleEvent Dokument-drucken an die angegebene Zielanwendung, d.h., weist die Anwendung an, das vorgegebene Dokument zu drucken.



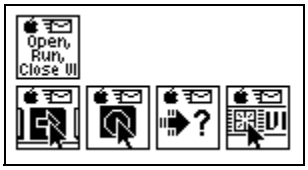
Sendet AE Anwendung-schließen

Sendet das AppleEvent Anwendung-schließen an eine angegebene Zielanwendung.



LabVIEW-spezifische AppleEvent-VIs

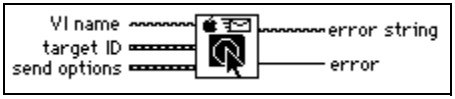
LabVIEW-spezifische AppleEvent-VIs senden Meldungen, die nur von LabVIEW-Anwendungen (Standard- und Laufzeitsysteme) erkannt werden. Die LabVIEW-spezifischen AppleEvent-VIs werden über das Menü **Funktionen»Kommunikation»LabVIEW-spezifische Apple Events** aufgerufen.



Diese VIs sollten nur bei der Kommunikation mit LabVIEW-Anwendungen eingesetzt werden. Diese Meldungen können an aktuelle LabVIEW-Anwendungen oder an LabVIEW-Anwendungen auf einem Netzwerk gesendet werden. Fehlerinformationen hierzu finden Sie in der Tabelle A-5, *AppleEvent-Fehlercodes*, im Anhang A, *Fehlercodes*.

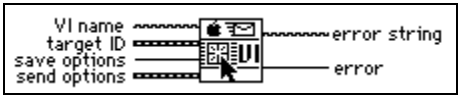
Sendet AE VI-abbrechen

Sendet das AppleEvent VI-abbrechen an die vorgegebene LabVIEW-Zielanwendung.



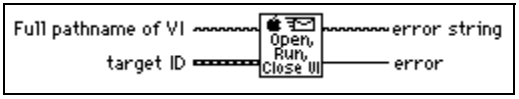
Sendet AESend VI-schließen

Sendet das AppleEvent VI-schließen an die vorgegebene LabVIEW-Zielanwendung.



Sendet AE VI-öffnen-ausführen-schließen

Verwendet die AppleEvent-VIs Dokument-öffnen, VI-ausführen, VI-aktiv? und VI-schließen, um ein VI durch eine vorgegebene LabVIEW-Anwendung öffnen, ablaufen und schließen zu lassen.



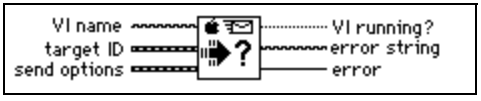
Bei diesem VI muß der vollständige Pfadname des auszuführenden VIs angegeben werden. Lesen Sie Kapitel 12, *Bedien- und Anzeigeelemente vom Typ Pfad und Refnum*, in Ihrem *Referenzhandbuch zur Programmierung in G* für eine Beschreibung der in der Palette Bedienelemente verfügbaren Bedienelemente und Anzeigen.

Sendet AE VI-ausführen

Sendet das AppleEvent VI-ausführen an die LabVIEW-Zielanwendung.

Sendet AE VI-aktiv?

Sendet das AppleEvent VI-aktiv? an die vorgegebene LabVIEW-Zielanwendung. **VI running?** ist ein Boolescher Wert, der anzeigt, ob das VI gerade abläuft.



Fortgeschrittene Themen

Dieser Abschnitt beschreibt einen Teil der fortgeschrittenen Programmierung, die mittels AppleEvent-VIs vorgenommen werden kann.

Konstruieren und Senden anderer AppleEvents

Neben den VIs, die gängige AppleEvents senden, können Lower-Level-VIs zum Senden von beliebigen AppleEvents eingesetzt werden. Der Einsatz dieser VIs erfordert mehr Kenntnisse als der Einsatz der vorangegangenen VIs in diesem Kapitel. Wenn Sie am Einsatz dieser VIs interessiert sind, sollten Sie mit der Besprechung von AppleEvents in den Unterlagen *Inside Macintosh, Volume VI* und *AppleEvents Registry* vertraut sein.

Beim Senden von AppleEvents müssen verschiedene Informationen eingeschlossen werden. Die Event-Klasse und die Event-ID kennzeichnen das AppleEvent, das gesendet wird. Bei der Event-Klasse handelt es sich um einen Code aus vier Buchstaben, der die AppleEvent-Gruppe angibt. Ein AppleEvent von der Klasse `core` (Kern) z.B. kennzeichnet ein AppleEvent als dem Satz core-AppleEvents anghörig. Bei der Event-ID handelt es sich um einen anderen vierbuchstabigen Code, der das spezielle, zu sendende AppleEvent kennzeichnet. Bei `odoc` handelt es sich um den Code mit vier Buchstaben für das AppleEvent Dokumente-öffnen, einem der core-AppleEvents. Um ein AppleEvent mit Hilfe des VIs Sendet AE zu senden, sind die Event-Klasse und die Event-ID zu einem String aus acht Buchstaben zu verketten. Um z.B. das AppleEvent Dokumente-öffnen zu senden, ist an das VI Sendet AE der Code aus acht zeichen `coreodoc` einzugeben.

Wenn ein AppleEvent an eine andere Anwendung gesendet wird, muß die **Ziel-ID** und **Optionen senden**, wie weiter vorn in diesem Kapitel beschrieben, angegeben werden.

Darüber hinaus kann ein Array aus Parametern angegeben werden, wenn die Zielanwendung zur Ausführung des angegebenen AppleEvents Zusatzinformationen benötigt. Da sich die Datenstruktur der Parameter für AppleEvents für den Einsatz im LabVIEW-Diagramm nicht gut eignet, akzeptiert das VI Sendet AE diese Parameter als ASCII-Strings. Diese Strings müssen die im nächsten Abschnitt beschriebene Grammatik befolgen. Diese Grammatik kann zur Beschreibung eines beliebigen AppleEvent-Parameters verwendet werden. Das VI Sendet AE interpretiert diesen String, um die angemessene Datenstruktur für ein AppleEvent zu erzeugen, und sendet anschließend das Event an das vorgegebene Ziel.

AppleEvent-Parameter erstellen

In den meisten Fällen besteht ein AppleEvent-Parameter aus einem einzigen Wert, kann jedoch auch recht komplex sein und eine hierarchische Struktur besitzen, die weitere Komponenten enthält, welche ihrerseits andere Komponenten enthalten können. In LabVIEW ist ein Parameter als String konstruiert, der einer einfachen Grammatik folgt, mit der alle Datenarten, aus denen ein AppleEvent-Parameter bestehen kann, einschließlich komplexer Strukturen, beschrieben werden können.

Ein AppleEvent-Parameter beginnt mit einem Schlüsselwort, einem vierbuchstabigen Code, der die Bedeutung des Parameters beschreibt. Wenn z.B. ein Parameter ein direkter Parameter ist (einer der gebräuchlichsten Parametertypen), muß mit Hilfe von ---- (vier Bindestrichen) in dem vierbuchstabigen Code angegeben werden, daß das Schlüsselwort ein `keyDirectObject` ist. Andere Beispiele von Schlüsselworten beinhalten `savo`, kurz für Sichern, welches beim Senden von AppleEvent VI-schließen an LabVIEW verwendet wird. Die Dokumentation mit Einzelheiten der von einer Anwendung unterstützten AppleEvents sollte die für jeden Parameter verwendeten Schlüsselworte aufführen. Lesen Sie den Abschnitt *AppleEvents von anderen Anwendungen an LabVIEW senden* in diesem Kapitel für eine Auflistung der in LabVIEW einsetzbaren AppleEvents.

Nach dem Schlüsselwort müssen die Parameterdaten als String angegeben werden. AppleEvents können mit vielen unterschiedlichen Datentypen, Strings und Zahlen eingeschlossen, verwendet werden. Wenn der Datenstring angegeben wird, wird er vom VI Sendet AE entsprechend der Art der Datenformatierung und optionaler Richtlinien, die in den String eingebettet werden können, in den gewünschten Datentyp konvertiert. Jeder Datenabschnitt ist mit einem vierbuchstabigen Code verknüpft, wodurch sein Datentyp angezeigt wird. Die Zielanwendung interpretiert die Daten anhand dieses Codes. Wenn z.B. durch Kommata abgeteilte Objekte in eckigen Klammern eingeschlossen sind, wird eine Liste mit *AE-Deskriptoren* aufgestellt und an die Liste der Datentyp `list` vergeben; hierbei könnte jedes durch komma getrennte Objekt u.U. wiederum andere Objekte, einschließlich Listen, darstellen.

Eine Anzahl von VIs auf der Palette **AppleEvents-VI** kann zur Erzeugung von gebräuchlicheren Parameterstrings eingesetzt werden, Beinamen (Aliases) und Deskriptor-Listen eingeschlossen, welche zum Verweisen auf Dateien in Parametern bzw. zur Kennzeichnung einer Liste mit Objekten als Parameter verwendet werden. Diese Strings können verkettet oder kaskadenförmig aneinandergereiht werden, um einen komplexeren Parameter herzustellen.

Tabelle 52-1 beschreibt das Format von AppleEvent-Deskriptorstrings und gibt VIs an, die die Deskriptoren, wo zweckmäßig, erzeugen.

Tabelle 52-1. Formate von AppleEvent-Deskriptorstrings

Um Daten zu senden als:	ist der String zu formatieren als:	Der Parameter gehört zum Codetyp:	Beispiele:	VI, das den String konstruieren kann:
Einen Integer-Wert	Eine Reihe von dezimalen Zahlenzeichen, wahlweise mit einem negativen Vorzeichen.	lang oder kurz	`1234` `–5678`	nicht zutreffend

Aufgezählte Daten	Ein Code aus vier Zeichen. Falls zu lang, wird er gekürzt; falls zu kurz, wird er mit Leerzeichen aufgefüllt. Falls in einfachen Anführungsstrichen (’), kann er jedes beliebige Zeichen enthalten; ansonsten kann er nicht mit einer Zahl beginnen oder die folgenden Zeichen enthalten: `@ ' : - , ( [ { } ] )`.	enum (aufgezählt)	`whos`
			`'@all'`
			`long`
			`>=`
			`'86it'`
				nicht zutreffend

Einen String	Ist die gewünschte Zeichenfolge in geschweifte, doppelte, öffnende bzw. schließende Anführungszeichen einzuschließen (welche mit <Option-[> bzw. <Option-Umschalt-[> eingegeben werden). Beachten Sie, daß der String nicht mit einer Null abgeschlossen wird.	TEXT	`“put x into card field 5”` `“Hi There”`	nicht zutreffend

Tabelle 52-1. Formate von AppleEvent-Deskriptorstrings (Fortsetzung)

Um Daten zu senden als:	ist der String zu formatieren als:	Der Parameter gehört zum Codetyp:	Beispiele:	VI, das den String konstruieren kann:
Eine AE-Aufzeichnung	Eine durch komma geteilte Liste mit Elementen, die von geschweiften Klammern eingeschlossen ist, wobei jedes Element aus einem Schlüsselwort (einem Typencode) besteht, das von einem Doppelpunkt gefolgt ist, dem ein Wert folgt; es kann jedem Typen in dieser Liste angehören.	reco (Aufzeichnung)	`{x:100, y:–100}` `{'origin': {x:100, y:–100}, extent: {x:500, y:500}, cont:[1,5, 25]}`	AE Aufzeichnung -erstellen
Eine AE-Deskriptorliste	Eine durch komma geteilte Liste mit Deskriptoren in eckigen Klammern.	list (Liste)	`[123, –58, “test”]`	AE-Deskriptorliste -erstellen
hexadezimale Daten	Eine gerade Anzahl von hexadezimalen Zeichen von französischen Anführungszeichen eingeschlossen (« wird durch <Option-\> und » durch <Option-Umschalt-\> eingegeben).	?? (muß erzwungen werden - siehe nächsten Eintrag)	`«01 57 64 fe AB C1»`	(Hex-Daten bilden eine Komponente des mit Beinamen erstellen produzierten Strings)

Tabelle 52-1. Formate von AppleEvent-Deskriptorstrings (Fortsetzung)

Um Daten zu senden als:	ist der String zu formatieren als:	Der Parameter gehört zum Codetyp:	Beispiele:	VI, das den String konstruieren kann:

Einen anderen Datentyp	Die Daten, die nach einem der in dieser Tabelle aufgelisteten Typen erstellt sind, sind in Klammern zu setzen und der gewünschte Typencode voranzustellen. Wenn es sich um numerische Daten handelt, zwingt LabVIEW die Daten, wenn möglich, in den vorgegebenen Datentyp und gibt, wenn nicht möglich, den Fehlercode `errAECoercionFail` aus. Wenn die Daten einem anderen Typen angehören, ersetzt LabVIEW den alten Typencode mit dem vorgegebenen Typencode.	Dem angegebenen Code-Typen	`sing(1234)`
			`alias(«hex dump of an alias»)`
			
			`type(line)`
			`rang{star: 5, stop: 6}`
				nicht zutreffend
				Beinamen erstellen
				erstellt einen hexadezimalen Ausdruck einer Dateibeschreibung
				nicht zutreffend
				nicht zutreffend

Keine Daten	Ein leerer String braucht nicht in einen Typen gezwungen zu werden.	null (Null)	`( )`	nicht zutreffend

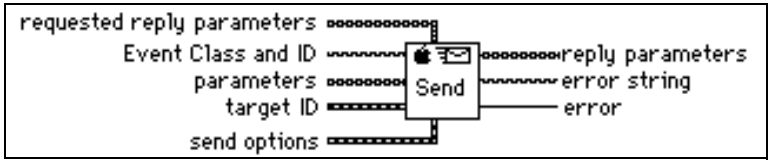
Low-Level-AppleEvent-VIs

Low-Level-AppleEvent-VIs können verwendet werden, um AppleEvent-Parameter zu konstruieren und AppleEvents zu senden. Die High-Level-VIs zum Senden von AppleEvents, die weiter vorn in diesem Kapitel beschrieben sind, basieren auf dem VI Sendet AE und stellen gute Beispiele für die Erstellung von AppleEvents und deren Parameter dar.

Auf die Palette **Low-Level Apple Events** wird über das Popup-Menü vom Icon **Low-Level Apple Events** zugegriffen.

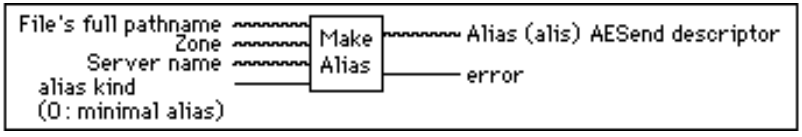
Sendet AE

Sendet ein AppleEvent in den vorgegebenen Parametern an die angegebene Zielanwendung.



Beinamen erstellen (Alias erstellen)

Erstellt von der Pfadangabe und dem Speicherplatz auf dem Netzwerk eine einmalige Beschreibung einer Datei in Form eines Beinamens. Diese Beschreibung kann mit dem VI Sendet AE zum Senden eines AppleEvents, das sich auf eine Datei bezieht, verwendet werden.



Bei einem Beinamen (Alias) handelt es sich um eine Datenstruktur, die von Macintoshs Toolbox zur Beschreibung von Objekten (Dateien, Verzeichnissen/Ordnern und Laufwerken/Datenträgern) verwendet wird. Er ist nicht mit einer Alias-Datei des Finders zu verwechseln. Ein minimaler Beiname enthält eine vollständige Pfadangabe zu einer Datei und möglicherweise der Zone (Unterteilung) und des Servers, wo sich die Datei befindet. Ein voller Beiname enthält weitere Informationen wie Erstellungsdatum, Dateityp und Ersteller. (Die komplette Beschreibung der Beinamen-Struktur unterliegt der Geheimhaltung von Apple Computer Inc.) Beinamen bilden die gebräuchlichste Methode, einem AppleEvent ein Objekt im Dateisystem zuzuweisen.

AE-Parameter mittels Objekt-Bezeichner erstellen

Apple hat zur Erstellung von AppleEvents ein High-Level-Interface aufgebaut, das als Object Support Library (Bibliothek zur Objektunterstützung) bezeichnet wird. Dieses Interface liegt als Schicht über den weiter vorn in diesem Kapitel beschriebenen Datenstrukturen für AppleEvent-Parameter. Dieses Interface hilft bei der Erstellung gebräuchlicher Parametertypen, einschließlich Bereichsangaben. LabVIEWs VIs Objektunterstützung befinden sich auf der Popup-Palette **Low-Level Apple Events**.

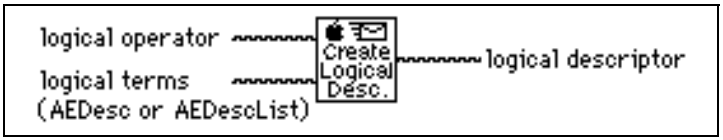
AE-Vergleichsdeskriptor erstellen

Erstellt einen String, der eine Aufzeichnung eines AppleEvent-Vergleichs beschreibt und festlegt, wie AppleEvent-Objekte mit anderen AppleEvent-Objekten oder einer Deskriptoraufzeichnung zu vergleichen sind.

Der Ausgangsstring Vergleichsdeskriptor kann z.B. als ein Argument für das VI Sendet AE eingesetzt werden oder als ein Argument für das VI AE-Objekt-Bezeichner erstellen eingesetzt werden, um einen komplexeren Deskriptorstring aufzustellen. Ein Beispiel für die Verwendung dieses VIs finden Sie im Abschnitt *Beispiele für Objektunterstützungs-VIs* in diesem Kapitel.

AE logischen Deskriptor erstellen

Erstellt einen String, der einen logischen AppleEvent-Deskriptor beschreibt und der mit dem VI Sendet AE eingesetzt wird.



Logische AppleEvent-Aufzeichnungen beschreiben logische oder Boolesche Ausdrücke mehrfacher Glieder wie dem UND zweier AppleEvent-Vergleichsaufzeichnungen. Der logische Deskriptorstring kann z.B. als ein Argument für das VI Sendet AE oder als ein Argument für das VI AE-Objekt-Bezeichner erstellen eingesetzt werden, um einen komplexeren Deskriptorstring aufzustellen. Ein Beispiel für die Verwendung dieses VIs finden Sie im Abschnitt *Beispiele für Objektunterstützungs-VIs* in diesem Kapitel.

AE Objekt-Bezeichner erstellen

Erstellt einen String, der ein AppleEvent-Objekt beschreibt und mit dem VI Sendet AE eingesetzt wird.

Ein Objekt-Bezeichner ist eine AppleEvent-Aufzeichnung vom Typ `obj` und beschreibt ein bestimmtes Objekt. Er besitzt vier Elemente: die Objektklasse, das enthaltene Objekt, einen Code, der die Form der Beschreibung anzeigt, und die Beschreibung des Objekts.

AE Bereichsdeskriptor erstellen

Erstellt einen String, der eine Aufzeichnung eines AppleEvent-Bereichsdeskriptors beschreibt und mit dem VI Sendet AE eingesetzt wird.

Bereichsdeskriptoraufzeichnungen werden in Objektbezeichnern verwendet, deren Schlüsselform die FormRange (`rang`) ist. Sie beschreiben einen Objektbereich mit zwei Objektbezeichnern: der Anfang und das Ende des Bereichs.

AE Deskriptorliste erstellen

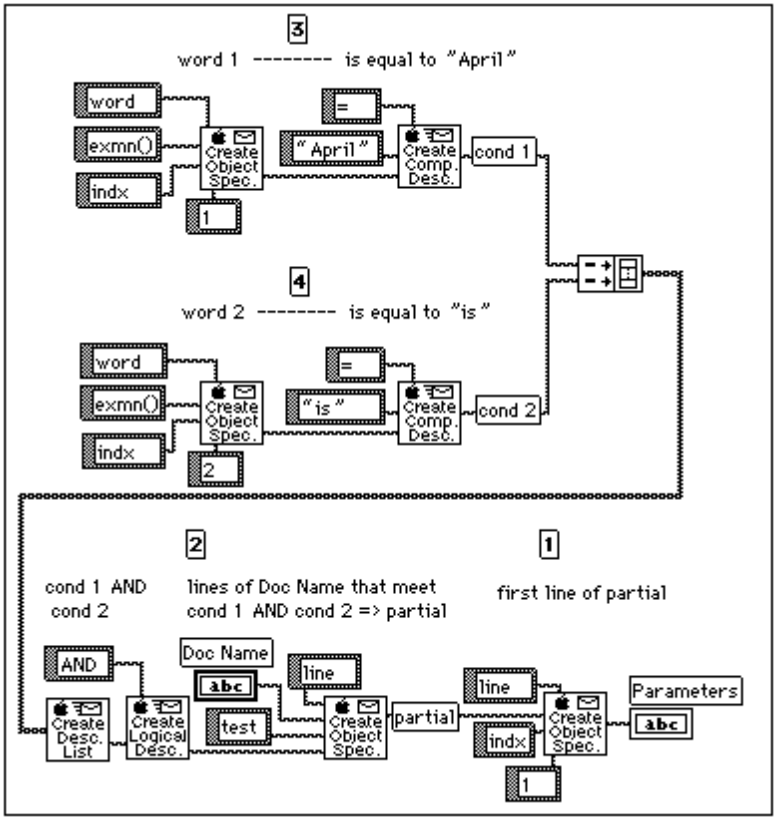
Erstellt einen String, der eine Auflistung von AppleEvent-Deskriptoren beschreibt und mit dem VI Sendet AE eingesetzt werden kann. Deskriptorlisten werden häufig bei der Erstellung von Operanden für einen logischen Deskriptor verwendet.

AE Aufzeichnung erstellen

Erstellt einen String, der eine Aufzeichnung eines AppleEvent-Deskriptors beschreibt und dann mit dem VI Sendet AE eingesetzt werden kann. Ein Aufzeichnungsdeskriptor kann zur Bündelung von Deskriptoren verschiedener Typen verwendet werden. Jeder Deskriptor verfügt über sein eigenes Schlüsselwort oder eigenen Namen und Wert.

Beispiele für Objektunterstützungs-VIs

Im folgenden Beispiel wird ein AppleEvent-Parameter mit Hilfe der Objektunterstützungs-VIs erstellt. Dieses Beispiel erstellt einen AppleEvent-Parameter, der an ein Textverarbeitungsprogramm gesendet wird und das Textverarbeitungsprogramm auffordert, die erste Zeile eines vorgegebenen Dokumentes zurückzugeben, deren erstes Wort **April** ist und deren zweites Wort **ist** ist.



Der folgende String, den obenstehendes Diagramm erstellt, ist recht kompliziert; es sind Einrückungen hinzugefügt worden, um den String lesbarer zu gestalten. Schlagen Sie in der *AppleEvent Registry* nach, um weitere Informationen über die Object Support Library zu erhalten.

```
obj
   want: type('line'),
   from: obj {
      want: type('line'),
      from: Doc Name,
      form: test,
      seld: logi {
         term:[
            cmpd{
               relo:=,
                  obj1:"April",
                  obj2:obj {
                     want: type('word'),
                     from: exmn( ),
                     form: indx,
                     seld: 1
                  }
               },
            cmpd{
               relo:=,
                  obj1:"is",
                  obj2:obj {
                     want: type('word'),
                     from: exmn( ),
                     form: indx,
                     seld: 2
                  }
               }
            ],
             logc: AND
         }
      },
   form: indx,
   seld: 1
```

AppleEvents von anderen Anwendungen an LabVIEW senden

LabVIEW reagiert auf die geforderten AppleEvents, die von allen Anwendungen unter System 7 nach Apples Erwartung unterstützt werden sollen, sowie auf LabVIEW-spezifische AppleEvents, die speziell für LabVIEW entworfen wurden. Beide Kategorien sind in den folgenden Abschnitten beschrieben.

Geforderte AppleEvents

LabVIEW reagiert auf die geforderten AppleEvents, d.h., Anwendung öffnen, Dokument öffnen, Dokument drucken und Anwendung schließen. Diese Events sind in *Inside Macintosh, Volume VI* beschrieben.

LabVIEW-spezifische AppleEvents

LabVIEW reagiert gleichfalls auf die LabVIEW-spezifischen AppleEvents VI ausführen, VI abbrechen, VI Aktiv? und VI schließen. Mit diesen Events und dem Event AE Dokument-öffnen können andere Anwendungen LabVIEW programmatisch anweisen, ein VI zu öffnen, auszuführen und nach der Ausführung zu schließen. Ein umfassendes Verständnis von AppleEvents, wie sie in *Inside Macintosh, Volume VI* und der AppleEvent Registry beschrieben sind, bilden eine Voraussetzung für das Senden dieser AppleEvents von anderen Anwendungen an LabVIEW. Diese Events können zwischen zwei oder mehreren LabVIEW-Anwendungen mit Hilfe der Utility-VIs, wie im Abschnitt *Events an andere Anwendungen senden* im Kapitel 24, *AppleEvents*, im *LabVIEW Benutzerhandbuch* beschrieben, versendet werden.

Die LabVIEW-spezifischen AppleEvents sind in späteren Abschnitten in einer der AppleEvent Registry ähnlichen Form beschrieben.

Antworten auf AppleEvents

Wenn LabVIEW nicht fähig ist, ein AppleEvent auszuführen, enthält die Antwort einen Fehlercode. Wenn es sich bei dem Fehler nicht um einen standardmäßigen AppleEvent-Fehler handelt, enthält die Antwort auch einen String, der den Fehler beschreibt. Anhang A, *Fehlercodes*, faßt die LabVIEW-spezifischen Fehler, die in einer Antwort auf ein AppleEvent zurückgegeben werden können, zusammen.

Event: VI ausführen

Beschreibung

Weist LabVIEW an, das angegebene VI oder die angegebenen VIs auszuführen. Vor der Ausführung dieses Events muß die LabVIEW-Anwendung laufen und das VI geöffnet sein (das VI kann mit Hilfe des AppleEvents Dokument-öffnen geöffnet werden).

Event-Klasse

`LBVW` (Benutzerspezifische Events verwenden den Typ Anwendungsersteller als Event-Klasse)

Event-ID

`GoVI ----`

Event-Parameter

Beschreibung	Schlüsselwort	Standardtyp
VI oder Liste mit VIs	`keyDirectObject (----)`	typeChar (char) (erforderlich) oder Liste mit typeChar (list)

Antwort-Parameter

Beschreibung	Schlüsselwort	Standardtyp
Keine		

Mögliche Fehler

Fehler	Wert	Beschreibung
kLVE_InvalidState (ungültiger Zustand)	1000	Das VI befindet sich in einem Zustand, der seine Ausführung nicht zuläßt.
kLVE_FPNotOpen (FP nicht geöffnet)	1001	Das Frontpanel ist nicht geöffnet.
kLVE_CtrlErr (Strg-Fehler)	1002	Bedienelemente des VIs auf dem Frontpanel befinden sich in einem Fehlerzustand.
kLVE_VIBad (schlechtes VI)	1003	Das VI ist kaputt.
kLVE_NotInMem (nicht im Speicher)	1004	Das VI befindet sich nicht im Speicher.

Event: VI abbrechen

Beschreibung

Weist LabVIEW an, das angegebene VI oder die angegebenen VIs abzubrechen. Vor der Ausführung dieses Events muß die LabVIEW-Anwendung laufen und das VI geöffnet sein (das VI kann mit Hilfe des AppleEvents Dokument-öffnen geöffnet werden). Diese Meldung kann nur an VIs gesendet werden, die auf der obersten Ebene ablaufen (SubVIs werden nur abgebrochen, wenn das aufrufende VI abgebrochen wird).

Event-Klasse

LBVW (Benutzerspezifische Events verwenden den Typ Anwendungsersteller als Event-Klasse)

Event-ID

RsVI

Event-Parameter

Beschreibung	Schlüsselwort	Standardtyp
VI oder Liste mit VIs	`keyDirectObject (----)`	typeChar (char) (erforderlich) oder Liste mit typeChar (list)

Antwort-Parameter

Beschreibung	Erforderliches? Schlüsselwort	Standardtyp
Keine		

Mögliche Fehler

Fehler	Wert	Beschreibung
`kLVE_InvalidState` (ungültiger Zustand)	`1000`	Das VI befindet sich in einem Zustand, der seine Ausführung nicht zuläßt.
`kLVE_FPNotOpen` (FP nicht geöffnet)	`1001`	Das Frontpanel ist nicht geöffnet.
`kLVE_NotInMem` (nicht im Speicher)	`1004`	Das VI befindet sich nicht im Speicher.

Event: VI aktiv?

Beschreibung

Fordert die Auskunft an, ob ein bestimmtes VI gerade abläuft. Vor der Ausführung dieses Events muß die LabVIEW-Anwendung laufen und das VI geöffnet sein (das VI kann mit Hilfe des AppleEvents Dokument-öffnen geöffnet werden). Die Antwort zeigt an, ob das VI gegenwärtig ausgeführt wird.

Event-Klasse

LBVW (Benutzerspezifische Events verwenden den Typ Anwendungsersteller als Event-Klasse.)

Event-ID

VIAc

Event-Parameter

Beschreibung	Schlüsselwort	Standardtyp
VI Name (erforderlich)	keyDirectObject (----)	typeChar (char)

Antwort-Parameter

Beschreibung	Schlüsselwort	Standardtyp
Active? (erforderlich)	keyDirectObject (----)	Booleschen Wert (bool) eingeben

Mögliche Fehler

Fehler	Wert	Beschreibung
kAEvtErrFPNotOpen (FP nicht geöffnet)	1001	Das Frontpanel des VIs ist nicht geöffnet.
kLVE_NotInMem (nicht im Speicher)	1004	Das VI befindet sich nicht im Speicher.

Event: VI schließen

Beschreibung

Weist LabVIEW an, das angegebene VI oder die angegebenen VIs zu schließen. Vor der Ausführung dieses Events muß die LabVIEW-Anwendung laufen und das VI geöffnet sein (das VI kann mit Hilfe des AppleEvents Dokument-öffnen geöffnet werden).

Event-Klasse

`LBVW` (Benutzerspezifische Events verwenden den Typ Anwendungsersteller als Event-Klasse)

Event-ID

`ClVI`

Event-Parameter

Beschreibung	Schlüsselwort	Standardtyp
VI oder Liste mit VIs	`keyDirectObject` (`----`)	typeChar (char) (required) oder Liste mit typeChar (list)
Save Options (Speicher-optionen - nicht erforderlich)	`keyAESaveOptions` (`savo`)	typeEnum (enum) mögliche Werte: yes (ja) und no (nein)

Antwort-Parameter

Beschreibung	Schlüsselwort	Standardtyp
Keine		

Mögliche Fehler

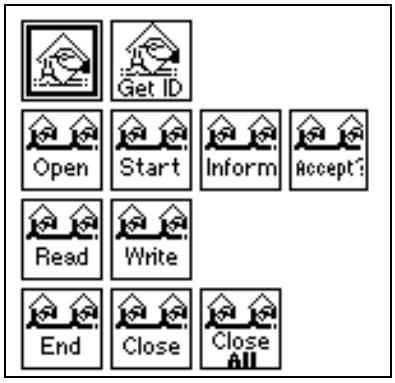
Fehler	Wert	Beschreibung
`kAEvtErrFPNotOpen (FP nicht geöffnet)`	`1001`	Das Frontpanel des VIs ist nicht geöffnet.
`kLVE_NotInMem (nicht im Speicher)`	`1004`	Das VI befindet sich nicht im Speicher.
`cancelError (Abbruch-Fehler)`	`43`	Der Nutzer hat die Operation Schließen abgebrochen.

53

Programm-zu-Programm-Kommunikations-VIs

Dieses Kapitel beschreibt die LabVIEW VIs für die Program-To-Program Communication (PPC = Programm-zu-Programm-Kommunikation), eine Low-Level-Form von Apple Interapplication Communication (IAC = Kommunikation von Apple-Anwendungen untereinander), durch die Macintosh-Anwendungen Datenblöcke senden und empfangen.

Die folgende Abbildung zeigt die **PPC-VI**-Palette, die über das Menü **Funktionen»Kommunikation»PPC** aufgerufen wird.



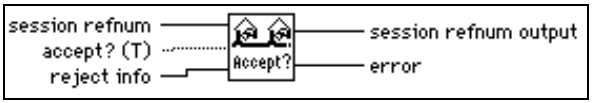
Beispiele für die Verwendung der PPC-VIs finden Sie unter den Beispielen in `examples\comm\PPC Examples.llb`.

Beschreibungen der PPC-VIs

Die folgenden PPC-VIs sind verfügbar.

PPC-Sitzung akzeptieren

Akzeptiert eine Anforderung für eine PPC-Sitzung oder weist sie zurück, je nach dem Booleschen Wert **Accept?**.



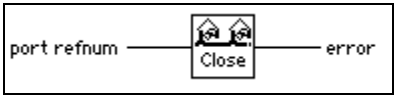
Die das VI PPC-Sitzung akzeptieren verwendenden Anfragen sollten unverzüglich akzeptiert oder zurückgewiesen werden, weil der andere Computer solange wartet (an der Leitung hängt), bis das VI seinen Versuch, eine Sitzung einzuleiten, akzeptiert oder zurückweist oder bis ein Fehler eintritt.

PPC-Browser

Für Informationen über den PPC-Browser lesen Sie bitte das Kapitel 52, *AppleEvent-VIs*, in diesem Handbuch.

Alle PPC-Anschlüsse schließen

Schließt alle von dem VI PPC-Anschluß öffnen geöffneten PPC-Anschlüsse.

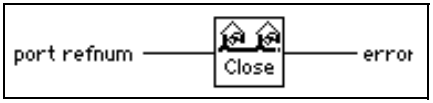


Das Schließen eines Anschlusses beendet alle anstehenden Aufrufe, die mit diesem Anschluß verknüpft sind, mit einem portClosedErr (Fehler - 916).

Das VI Alle PPC-Anschlüsse schließen kann eingesetzt werden, um mit abnormen Zuständen umzugehen, die Anschlüsse offen lassen. Ein Beispiel für einen abnormen Zustand finden Sie, wenn ein VI abgebrochen wird, ehe es die Ausführung normal beenden und den PPC-Anschluß schließen konnte. Sie können das VI Alle PPC-Anschlüsse schließen während der Entwicklung von VIs nutzen, wenn solche Fehler mit gewisser Wahrscheinlichkeit gemacht werden, oder als Vorsichtsmaßnahme zu Beginn eines Programms, das Anschlüsse öffnet.

PPC-Anschluß schließen

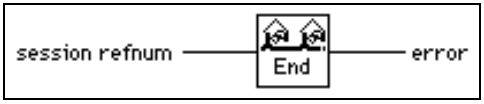
Schließt den angegeben PPC-Anschluß.



Das Schließen eines Anschlusses beendet alle anstehenden Aufrufe, die mit diesem Anschluß verknüpft sind, mit einem portClosedErr (Fehler - 916).

PPC-Sitzung beenden

Beendet die angegebene PPC-Sitzung.



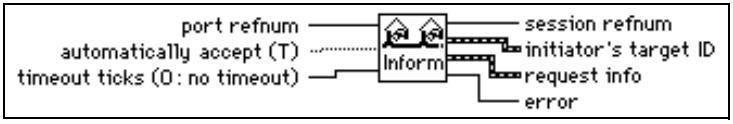
Das Beenden einer Sitzung verursacht, daß alle anstehenden Aufrufe, die mit der Sitzung verknüpft sind (die Aufrufe PPC lesen und PPC schreiben), mit einem sessClosedErr (Fehler 917) abschließen.

Ziel-ID erhalten

Für Informationen über das VI Ziel-ID erhalten lesen Sie bitte Kapitel 52, *AppleEvent-VIs*, in diesem Handbuch.

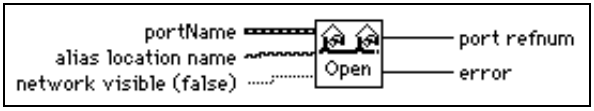
PPC-Sitzung benachrichtigen

Wartet auf eine Anforderung für eine PPC-Sitzung.

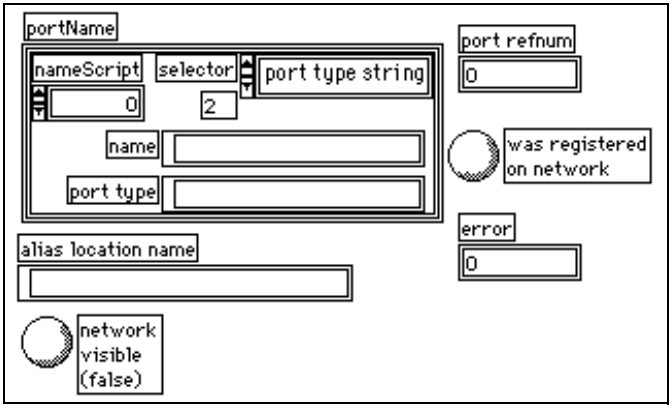


PPC-Anschluß öffnen

Öffnet einen Anschluß für eine PPC-Kommunikation und gibt eine einmalige Anschlußreferenznummer in **Anschluß-Refnum** aus. Ein einziger Anschluß kann für mehrere Sitzungen verwendet werden.



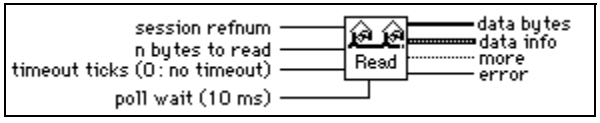
Beim Öffnen eines Anschlusses mit dem VI PPC-Anschluß öffnen muß ein Cluster **portName** vorgegeben werden.



Lesen Sie die Online-Hilfe von LabVIEW für weitere Informationen über dieses VI.

PPC lesen

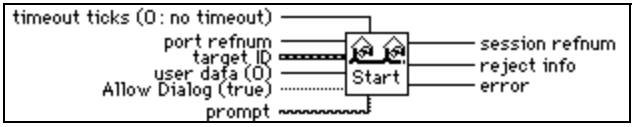
Liest von einer angegebenen Sitzung einen Informationsblock ein. Wenn eine Zeitüberschreitung auftritt oder das VI vor Abschluß der Ausführung abbricht, schließt sich der von **Anschluß-Refnum** dargestellte Anschluß.



Das VI PPC lesen arbeitet asynchron, indem es mit dem Lesen der angegebenen Daten beginnt und weiterhin abfragt, bis das Lesen beendet ist.

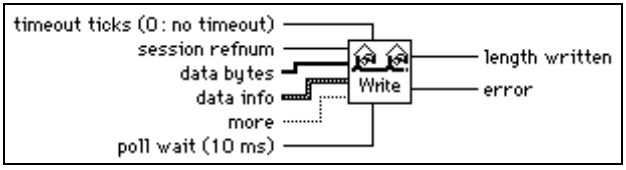
PPC-Sitzung aufnehmen

Versucht eine Sitzung mit der durch **Ziel-ID** angegebenen Anwendung aufzunehmen. Wenn eine Zeitüberschreitung auftritt oder das VI vor Abschluß der Ausführung abbricht, schließt sich der von **Anschluß-Refnum** dargestellte Anschluß.



PPC schreiben

Schreibt einen Informationsblock an die angegebene Sitzung. Wenn eine Zeitüberschreitung auftritt oder das VI vor Abschluß der Ausführung abbricht, ist der von **Anschluß-Refnum** dargestellte Anschluß geschlossen. Das VI PPC schreiben arbeitet asynchron, indem es mit dem Schreiben der angegebenen Daten beginnt und weiterhin abfragt, bis das Schreiben beendet ist.



A

Fehlercodes

In diesem Dokument sind Tabellen enthalten, in denen alle numerischen Fehlercodes für LabVIEW aufgeführt sind.

Verbinden Sie Error-Handler-VIs mit anderen VIs, um die Beschreibung eines eventuell vorkommenden Fehlers zu erhalten. Error-Handler-VIs können auch ein Dialogfeld mit der Beschreibung einer Fehlermeldung und mit Schaltflächen anzeigen, mit denen Sie die Ausführung stoppen oder fortsetzen können. Siehe Ausführungen zum Thema *Fehlerbehandlung* in der LabVIEW *Online Reference*.

☞ **Hinweis** ***Alle Beschreibungen von Fehlercodes sind auch in den Konfigurations-Utility-Hilfepanelen in Windows- und Macintosh-Plattformen enthalten.***

Numerische Fehlercodes

Die Tabellen sind in ungefähr absteigender Reihenfolge, von den negativen zu den positiven Werten, angeordnet. Tabellen mit negativen Zahlenwerten sind mit dem kleinsten absoluten Wert beginnend bis zum größten absoluten Wert angeordnet. Bitte beachten Sie, daß die Fehlercodes 5000 bis 9999 für Ihre Benutzung reserviert sind.

Tabelle A-1. Numerische Fehlercodes

	Fehlercodebereich			Tabelle
*	−1073807360	bis	−1073807231	VISA-Fehlercodes
	−20001	bis	−20065	Analyse-VI-Fehlercodes
	−10001	bis	−10943	Datenerfassungs-VI-Fehlercodes
	−1700	bis	−1719	AppleEvent-Fehlercodes
*	−1200	bis	−13xx	Instrumententreiber-Fehlercodes
	−900	bis	−932	PPC-Fehlercodes
*	0	bis	85	LabVIEW-Funktions-Fehlercodes
*	0	bis	32	GPIB-Fehlercodes
	1	bis	5	LabVIEW Spezifische PPC-Fehlercodes
*	53	bis	66	TCP/IP- und UDP-Fehlercodes

Tabelle A-1. Numerische Fehlercodes (Fortsetzung)

Fehlercodebereich				Tabelle
*	61	bis	65	Serielle-Anschluß-Fehlercodes
	1000	bis	1004	LabVIEW Spezifische AppleEvent-Fehlercodes
	14001	bis	14020	DDE-Fehlercodes

* In diesen Tabellen sind, von der Fehlerquelle abhängig, einige Fehlercodes mit überlappenden numerischen Werten, aber unterschiedlichen Bedeutungen enthalten.

Tabelle A-2. VISA-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
–1073807360	VI_ERROR_SYSTEM_ERROR	Unbekannter Systemfehler (verschiedene Fehler).
–1073807346	VI_ERROR_INV_OBJECT VI_ERROR_INV_SESSION	Die angegebene Sitzungs- oder Objektreferenz ist ungültig.
-1073807345	VI_ERROR_RSRC_LOCKED	Angegebener Sperrungstyp kann nicht erlangt oder angegebene Operation kann nicht durchgeführt werden, da die Ressource gesperrt ist.
–1073807344	VI_ERROR_INV_EXPR	Angabe eines ungültigen Ausdrucks für die Suche.
–1073807343	VI_ERROR_RSRC_NFOUND	Ungenügende Angaben über den Standort, oder das Gerät oder die Ressource sind im System nicht vorhanden.
–1073807342	VI_ERROR_INV_RSRC_NAME	Angabe einer ungültigen Ressourcenreferenz. Fehleruntersuchung.
–1073807341	VI_ERROR_INV_ACC_MODE	Ungültiger Zugriffsmodus.
–1073807339	VI_ERROR_TMO	Zeitbegrenzung vor Abschluß der Operation abgelaufen.
–1073807338	VI_ERROR_CLOSING_FAILED	Unfähig, die Zuweisung der bereits zugewiesenen Datenstrukturen, die mit dieser Session- oder Objektreferenz korrespondieren, rückgängig zu machen.
-1073807332	VI_ERROR_INV_JOB_ID	Angegebener Job identifizierer ist ungültig.
–1073807331	VI_ERROR_NSUP_ATTR	Das angegebene Attribut wird von der Ressource, auf die Bezug genommen wurde, nicht identifiziert oder unterstützt.
–1073807330	VI_ERROR_NSUP_ATTR_STATE	Der angegebene Status des Attributs ist nicht gültig oder wird nicht, wie von der Ressource definiert, unterstützt..
–1073807329	VI_ERROR_ATTR_READONLY	Das angegebene Attribut ist schreibgeschützt.

Tabelle A-2. VISA-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–1073807322	VI_ERROR_INV_EVENT	Der angegebene Ereignistyp wird von der Ressource nicht unterstützt.
–1073807321	VI_ERROR_INV_MECH	Angabe eines ungültigen Mechanismus'.
–1073807320	VI_ERROR_HNDLR_NINSTALLED	Es war kein Handler installiert.
–1073807319	VI_ERROR_INV_HNDLR_REF	Die angegebene Handlerreferenz ist ungültig.
–1073807318	VI_ERROR_INV_CONTEXT	Der angegebene Ereigniskontext ist ungültig.
–1073807308	VI_ERROR_RAW_WR_PROT_VIOL	Während der Übertragung ist eine Verletzung des Ursprungsschreibprotokolls vorgekommen.
–1073807307	VI_ERROR_RAW_RD_PROT_VIOL	Während der Übertragung ist eine Verletzung des Ursprungsleseprotokolls vorgekommen.
–1073807306	VI_ERROR_OUTP_PROT_VIOL	Gerät hat einen Ausgabeprotokollfehler während der Übertragung gemeldet.
–1073807305	VI_ERROR_INP_PROT_VIOL	Gerät hat einen Eingabeprotokollfehler während der Übertragung gemeldet.
–1073807304	VI_ERROR_BERR	Während der Übertragung ist ein Busfehler vorgekommen.
–1073807302	VI_ERROR_INV_SETUP	Operation kann wegen ungültiger Einrichtung nicht gestartet werden (aufgrund von Attributen, die auf einen nicht konsistenten Status eingestellt wurden).
–1073807300	VI_ERROR_ALLOC	Ungenügende Systemressourcen zur Durchführung notwendiger Speicherzuweisung.
–1073807299	VI_ERROR_INV_MASK	Angabe ungültiger Puffermaske.
–1073807298	VI_ERROR_IO	Lese-/Schreiboperation konnte wegen eines I/O -Fehlers nicht durchgeführt werden.
–1073807297	VI_ERROR_INV_FMT	Ein Bezeichner im Format-String ist ungültig.
–1073807295	VI_ERROR_NSUP_FMT	Ein Bezeichner im Format-String wird nicht unterstützt.
–1073807294	VI_ERROR_LINE_IN_USE	Die angegebene Trigger-Leitung ist belegt.
–1073807286	VI_ERROR_SRQ_NOCCURRED	Für diese Sitzung ist keine Serviceanforderung eingegangen.
–1073807282	VI_ERROR_INV_SPACE	Angabe eines ungültigen Adressenbereichs.
–1073807279	VI_ERROR_INV_OFFSET	Angabe eines ungültigen Offsets.
–1073807276	VI_ERROR_NSUP_OFFSET	Auf angegebenen Offset kann von dieser Hardware aus nicht zugegriffen werden.

Tabelle A-2. VISA-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–1073807273	VI_ERROR_WINDOW_NMAPPED	Die angegebene Session ist derzeit nicht eingetragen.
–1073807265	VI_ERROR_NLISTENERS	Kein Listener-Zustand entdeckt (sowohl NRFD als auch NDAC sind verlassen).
–1073807264	VI_ERROR_NCIC	Die mit dieser Session verbundene Schnittstelle ist derzeit nicht der Controller.
–1073807257	VI_ERROR_NSUP_OPER	Die jeweilige Session- oder Objektreferenz unterstützt diese Operation nicht.
–1073807242	VI_ERROR_NSUP_WIDTH	Die angegebene Breite wird von dieser Hardware nicht unterstützt.
–1073807239	VI_ERROR_INV_PROT	Das angegebene Protokoll ist ungültig.
–1073807237	VI_ERROR_INV_SIZE	Die angegebene Fenstergröße ist ungültig.
–1073807232	VI_ERROR_WINDOW_MAPPED	Die angegebene Session enthält bereits ein angelegtes Fenster.
–1073807231	VI_ERROR_NIMPL_OPER	Die jeweilige Operation ist nicht implementiert.

Tabelle A-3. Analyse-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
0	NoErr	Kein Fehler; der Aufruf war erfolgreich.
–20001	OutOfMemErr	Für die Durchführung der angegebenen Routine ist keine ausreichende Speicherkapazität vorhanden.
–20002	EqSamplesErr	Die Größe der Eingabesequenzen muß gleich sein.
–20003	SamplesGTZeroErr	Die Anzahl der Abtastwerte muß größer als Null sein.
–20004	SamplesGEZeroErr	Die Anzahl der Abtastwerte muß größer als Null oder gleich Null sein.
–20005	SamplesGEOneErr	Die Anzahl der Abtastwerte muß größer als oder gleich Eins sein.
–20006	SamplesGETwoErr	Die Anzahl der Abtastwerte muß größer als oder gleich Zwei sein.
–20007	SamplesGEThreeErr	Die Anzahl der Abtastwerte muß größer als oder gleich Drei sein.
–20008	ArraySizeErr	Die Eingabearrays enthalten nicht die korrekte Anzahl von Datenwerten für dieses VI.
–20009	PowerOfTwoErr	Die Größe des Eingabearrays muß eine Potenz von zwei sein: Größe= 2^m, $0 < m < 23$.

Tabelle A-3. Analyse-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–20010	MaxXformSizeErr	Die maximale Transformationsgröße wurde überschritten.
–20011	DutyCycleErr	Das Tastverhältnis muß der Bedingung entsprechen: 0 ≤ Tastverhältnis ≤ 100.
–20012	CyclesErr	Die Anzahl der Zyklen muß größer als Null oder kleiner als oder gleich der Anzahl der Abtastwerte sein.
–20013	WidthLTSamplesErr	Die Breite muß die Bedingung erfüllen: 0 < Breite < Abtastwert.
–20014	DelayWidthErr	Die Verzögerung muß die Bedingung erfüllen: 0 ≤ (Verzögerung + Breite) < Abtastwerte.
–20015	DtGEZeroErr	dt muß größer als oder gleich Null sein.
–20016	DtGTZeroErr	dt muß größer als Null sein.
–20017	IndexLTSamplesErr	Der Index muß die Bedingung erfüllen: 0 ≤ Index < Abtastwerte.
–20018	IndexLengthErr	Der Index muß die Bedingung erfüllen: 0 ≤ (Index + Länge) < Abtastwerte.
–20019	UpperGELowerErr	Der obere Wert muß größer oder gleich dem unteren Wert sein.
–20020	NyquistErr	Die Grenzfrequenz, f_c, muß die Bedingung erfüllen: $0 \le f_c \le \frac{f_s}{2}$.
–20021	OrderGTZeroErr	Die Reihe muß größer als Null sein.
–20022	DecFactErr	Der Dezimierfaktor muß die Bedingung erfüllen: 0 < Dezimier ≤ Abtastwerte.
–20023	BandSpecErr	Die Bandspezifikationen müssen die Bedingungen erfüllen $0 \le f_{low} \le f_{high} \le \frac{f_s}{2}$.
–20024	RippleGTZeroErr	Die Amplitude der Welligkeit muß größer als Null sein.
–20025	AttenGTZeroErr	Die Dämpfung muß größer als Null sein.
–20026	WidthGTZeroErr	Die Breite muß größer als Null sein.
–20027	FinalGTZeroErr	Der Endwert muß größer als Null sein.
–20028	AttenGTRippleErr	Die Dämpfung muß größer als die Ripple-Amplitude sein.
–20029	StepSizeErr	Die Stufengröße, μ, muß die Bedingung erfüllen: 0 ≤ μ ≤ 0.1.

Tabelle A-3. Analyse-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–20030	LeakErr	Der Streufaktor muß die Bedingung erfüllen: $0 \le leak \le \mu$.
–20031	EqRplDesignErr	Der Filter kann mit den angegebenen Eingabewerten nicht entworfen werden.
–20032	RankErr	Der Rang des Filters muß folgende Bedingung erfüllen: $1 \le (2\times Rang + 1) \le Größe$.
–20033	EvenSizeErr	Die Anzahl der Koeffizienten muß für diesen Filter ungerade sein.
–20034	OddSizeErr	Die Anzahl der Koeffizienten muß für diesen Filter gerade sein.
–20035	StdDevErr	Die Standardabweichung muß für die Normalisierung größer als Null sein.
–20036	MixedSignErr	Die Elemente für das **Y-Werte**-Array dürfen nicht Null sein; sie müssen entweder alle positiv oder negativ sein.
–20037	SizeGTOrderErr	Die Anzahl der Datenpunkte im **Y-Werte**-Array muß größer als Zwei sein.
–20038	IntervalsErr	Die Anzahl der Intervalle muß größer als Null sein.
–20039	MatrixMulErr	Die Anzahl der Spalten in der ersten Matrix ist nicht gleich der Anzahl der Reihen in der zweiten Matrix oder Vektor.
–20040	SquareMatrixErr	Bei der Eingabematrix muß es sich um eine Quadratmatrix handeln.
–20041	SingularMatrixErr	Das Gleichungssystem kann nicht gelöst werden, da die Eingabematrix singulär ist.
–20042	LevelsErr	Die Anzahl der Stufen ist außerhalb des Bereichs.
–20043	FactorErr	Die Stufe der Faktoren ist für einige Daten außerhalb des Bereichs.
–20044	ObservationsErr	Auf einer Faktorstufe wurden keine Beobachtungen gemacht.
–20045	DataErr	Die Gesamtzahl der Datenpunkte muß gleich dem Produkt der Ebenen für jeden Faktor und der Beobachtungen pro Zelle sein.
–20046	OverflowErr	Im berechneten F-Wert ist ein Überlauf vorhanden.
–20047	BalanceErr	Die Daten sind unausgeglichen. Alle Zellen müssen dieselbe Anzahl von Beobachtungen enthalten.

Tabelle A-3. Analyse-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–20048	ModelErr	Das Random-Effect-Modell wurde angefordert, wenn Fixed-Effect-Modell erforderlich war.
–20049	DistinctErr	Die *x*-Werte müssen typisch sein.
–20050	PoleErr	Die Interpolationsfunktion hat am angeforderten Wert einen Pol.
–20051	ColumnErr	Alle Werte in der ersten Spalte in der X-Matrix müssen 1 betragen.
–20052	FreedomErr	Es müssen einer oder mehr Freiheitsgrade vorhanden sein.
–20053	ProbabilityErr	Die Wahrscheinlichkeit muß zwischen Null und Eins liegen.
–20054	InvProbErr	Die Wahrscheinlichkeit muß größer oder gleich Null und kleiner als Eins sein.
–20055	CategoryErr	Die Anzahl der Kategorien oder Abtastwerte muß größer als Eins sein.
–20056	TableErr	Die Kontingenztafel darf keine negative Zahl enthalten.
–20061	InvSelectionErr	Eine der Eingabeauswahlen ist ungültig.
–20062	MaxIterErr	Die maximalen Iterationen wurden überschritten.
–20063	PolyErr	Die Polynominalkoeffizienten sind ungültig.
–20064	InitStateErr	Dieses VI ist nicht korrekt initialisiert worden.
–20065	ZeroVectorErr	Der Vektor kann nicht Null betragen.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
–10001	syntaxError	Im Eingangs-String wurde ein Fehler entdeckt; die Anordnung oder Reihenfolge der Zeichen im String stimmt nicht mit der erwarteten Anordnung überein.
–10002	semanticsError	Im Eingangs-String wurde ein Fehler entdeckt; der Syntax des Strings ist korrekt, aber bestimmte im String angegebene Werte stimmen nicht mit anderen im String angegebenen Werten überein.
–10003	invalidValueError	Der Wert eines numerischen Parameters ist ungültig.
–10004	valueConflictError	Der Wert eines numerischen Parameters stimmt nicht mit einem anderen Parameter überein, und die Kombination ist daher ungültig.
–10005	badDeviceError	Der Geräteparameter ist ungültig.
–10006	badLineError	Der Leitungsparameter ist ungültig.
–10007	badChanError	Ein Kanal für den Kartentyp oder die Eingabekonfiguration befindet sich außerhalb des Bereich; die Kombination der Kanäle ist unzulässig, oder die Abtastreihenfolge muß umgekehrt werden, so daß Kanal 0 am Ende ist.
–10008	badGroupError	Die Gruppe ist ungültig.
–10009	badCounterError	Der Counter ist ungültig.
–10010	badCountError	Die Zählung ist für den angegebenen Counter zu groß oder zu klein, oder die jeweilige I/O-Übertragungszählung ist für die aktuelle Puffer- oder Kanalkonfiguration ungeeignet.
–10011	badIntervalError	Die Analogeingangs-Abtastrate ist für die Anzahl der Kanäle und die Kanaltaktfrequenz zu schnell; oder die angegebene Taktfrequenz wird vom verbundenen Counterkanal oder I/O-Kanal nicht unterstützt.
–10012	badRangeError	Der Analogeingangs- oder Analogausgangs-Spannungsbereich ist für den angegebenen Kanal ungültig.
–10013	badErrorCodeError	Der Treiber hat einen unerkannten oder nicht aufgeführten Fehlercode zurückgegeben
–10014	groupTooLargeError	Die Gruppengröße ist für die Karte zu groß.
–10015	badTimeLimitError	Die Zeitbegrenzung ist ungültig.
–10016	badReadCountError	Die Lesezählung ist ungültig.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10017	badReadModeError	Der Lesemodus ist ungültig.
–10018	badReadOffsetError	Das Offset ist unerreichbar.
–10019	badClkFrequencyError	Die Frequenz ist ungültig.
–10020	badTimebaseError	Die Zeitbasis ist ungültig.
–10021	badLimitsError	Die Grenzen befinden sich außerhalb des Kartenbereichs.
–10022	badWriteCountError	In Ihrem Datenarray ist eine unvollständige Aktualisierung enthalten, oder es wird versucht, über den Rand des internen Puffers hinauszuschreiben, oder die Ausgangsoperation ist fortlaufend und die Länge des Arrays ist nicht ein Vielfaches der Hälfte der internen Puffergröße.
–10023	badWriteModeError	Der Schreibmodus ist außerhalb des Bereichs oder deaktiviert.
–10024	badWriteOffsetError	Durch das Addieren des Schreib-Offsets zur Schreibmarke gerät die Schreibmarke außerhalb des internen Puffers.
–10025	limitsOutOfRangeError	Die angeforderten Eingabegrenzen übersteigen die Fähigkeiten oder die Konfiguration der Karte. Es wurden alternative Grenzen ausgewählt.
–10026	badBufferSpecificationError	Die angeforderte Anzahl der Puffer oder die Puffergröße sind nicht gestattet; die Grenze des Lab-PC-Puffers ist z.B. 64K-Abtastwerte, oder die Karte unterstützt keine Mehrfachpuffer.
–10027	badDAQEventError	Für DAQEvents 0 muß der allgemeine Wert A größer als 0 und kleiner als die interne Puffergröße sein. Wenn DMA für DAQEvent 1 benutzt wird, muß der allgemeine Wert A die interne Puffergrüße in gleiche Teile teilen. Wenn TIO-10 für DAQEvent 4 benutzt wird, muß der allgemeine Wert A 1 oder 2 betragen.
–10028	badFilterCutoffError	Die angegebene Grenzfrequenz ist für dieses Gerät nicht gültig.
–10029	obsoleteFunctionError	Die von Ihnen aufgerufene Funktion wird von dieser Treiberversion nicht mehr unterstützt.
–10030	badBaudRateError	Die angegebene Baudrate für die Kommunikation mit dem seriellen Anschluß ist auf dieser Plattform ungültig.
–10031	badChassisIDError	Das angegebene SCXI-Chassis stimmt nicht mit einem konfigurierten SCXI-Chassis überein.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10032	badModuleSlotError	Der angegebene Steckplatz für das SCXI-Modul ist ungültig oder ist leer.
–10033	invalidWinHandleError	Der an die Funktion weitergegebene Fensterhandle ist ungültig.
–10034	noSuchMessageError	Keine konfigurierte Meldung stimmt mit der überein, die Sie versucht haben zu löschen.
–10080	badGainError	Die Verstärkung ist ungültig.
–10081	badPretrigCountError	Die Pretrigger-Abtastzählung ist ungültig.
–10082	badPosttrigCountError	Die Postrigger-Abtastzählung ist ungültig.
–10083	badTrigModeError	Der Triggermodus ist ungültig.
–10084	badTrigCountError	Die Triggerzählung ist ungültig.
–10085	badTrigRangeError	Der Triggerbereich oder das Trigger-Hysteresefenster sind ungültig.
–10086	badExtRefError	Der externe Referenzwert ist ungültig.
–10087	badTrigTypeError	Der Trigger-Typ-Parameter ist ungültig.
–10088	badTrigLevelError	Der Trigger-Level ist ungültig.
–10089	badTotalCountError	Die Gesamtzählung stimmt nicht mit der Puffergröße und der Pretrigger-Abtastzählung oder dem Kartentyp überein.
–10090	badRPGError	Der individuelle Bereich, die Polarität und Verstärkungseinstellungen sind gültig, aber die angegebene Kombination ist nicht zulässig.
–10091	badIterationsError	Sie haben versucht, für die Iterationsparameter eine ungültige Einstellung zu verwenden. Der Iterationswert muß 0 oder größer sein. Ihr Gerät ist u.U. auf nur zwei Werte begrenzt, 0 und 1.
–10092	lowScanIntervalError	Bei einigen Geräten ist eine Zeitlücke zwischen dem letzten Abtastwert in einem Scan und dem Start des nächsten Scan notwendig. Das von Ihnen angegebene Scanintervall bietet für die Karte keine ausreichend große Zeitlücke. Siehe die Funktion `SCAN_Start` in der Sprachschnittstelle API wegen einer Erklärung.
–10093	fifoModeError	Die Erzeugung des FIFO-Modulsignalverlaufs kann nicht verwendet werden, da mindestens eine Bedingung nicht erfüllt ist.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10100	badPortWidthError	Die angeforderte digitale Anschlußbreite ist kein Vielfaches der Hardware-Anschlußbreite oder kann von der DAQ-Hardware nicht erlangt werden.
–10120	gpctrBadApplicationError	Benutzung einer ungültigen Anwendung.
–10121	gpctrBadCtrNumberError	Benutzung von ungültiger counterNumber.
–10122	gpctrBadParamValueError	Benutzung von ungültigem paramValue.
–10123	gpctrBadParamIDError	Benutzung von ungültiger paramID.
–10124	gpctrBadEntityIDError	Benutzung von ungültiger entityID.
–10125	gpctrBadActionError	Benutzung von ungültiger Aktion.
–10200	EEPROMreadError	Unfähig, Daten von EEPROM zu lesen.
–10201	EEPROMwriteError	Unfähig, Daten zu EEPROM zu schreiben.
–10240	noDriverError	Die Treiber-Schnittstelle konnte den Treiber nicht finden oder öffnen.
–10241	oldDriverError	Eine der Treiberdateien oder die Konfigurations-Utility sind überaltert.
–10242	functionNotFoundError	Die angegebene Funktion befindet sich nicht im Treiber.
–10244	deviceInitError	Der Treiber ist während des Versuchs, das angegebene Gerät zu konfigurieren, auf einen Hardware-Initialisierungsfehler gestoßen.
–10245	osInitError	Der Treiber ist während des Versuchs, eine Operation durchzuführen, auf einen Fehler des Betriebssystems gestoßen, oder das Betriebssystem unterstützt eine vom Treiber durchgeführte Operation nicht.
–10246	communicationsError	Der Treiber kann nicht mit dem angegebenen externen Gerät kommunizieren.
–10248	dupAddressError	Die Basisadressen für zwei oder mehrere Geräte sind identisch; daher ist der Treiber nicht in der Lage, auf das angegebene Gerät zuzugreifen.
–10249	intConfigError	Die Interrupt-Konfiguration ist angesichts der Fähigkeiten des Computers oder des Geräts inkorrekt.
–10250	dupIntError	Die Interrupt-Level für zwei oder mehr Geräte sind die gleichen.
–10251	dmaConfigError	Die DMA-Konfiguration ist angesichts der Fähigkeiten des Computers/DMA-Controllers- oder Geräts unrichtig.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10252	dupDMAError	Die DMA-Kanäle für zwei oder mehr Geräte sind dieselben.
–10253	jumperlessBoardError	Unfähig, eine oder mehr steckbrückenlose Karten zu finden, die mit Hilfe der NI-DAQ-Konfigurations-Utility konfiguriert wurden.
–10254	DAQCardConfError	DAQCard kann aus einem der folgenden Gründe nicht konfiguriert werden: 1. Die korrekte Version der Karten- und Buchsenservice-Software ist nicht installiert. 2. Die Karte in der PCMCIA-Buchse ist keine DAQCard. 3. Die angeforderte Basisadresse und/oder Interrupt-Level sind nach Aussage des Karten- und Buchsenservice-Ressourcenmanagers nicht verfügbar. 4. Der Kartenservice konnte aufgrund unzureichender verfügbarer Speicherkapazität unter 1 MB in Windows 3.1 nicht geladen werden. Versuchen Sie unterschiedliche Einstellungen oder benutzen Sie AutoAssign im NI-DAQ-Konfigurations-Utility. Speicherkapazität unterhalb von 1 MB muß zur Konfiguration von DAQCard in Win 3.*x* verfügbar sein.
–10255	remoteChassisDriverInitError	Bei der Initialisierung des Treibers für Remote SCXI ist ein Fehler vorgekommen.
–10256	comPortOpenError	Beim Öffnen des angegebenen COM-Anschlusses ist ein Fehler vorgekommen.
–10257	baseAddressError	Angabe einer inkorrekten Basisadresse im Konfigurations-Utility.
–10258	dmaChannel1Error	Angabe von einem inkorrekten DMA Kanal 1 im Konfigurations-Utility oder durch das Betriebssystem.
–10259	dmaChannel2Error	Angabe von einem inkorrekten DMA Kanal 2 im Konfigurations-Utility oder durch das Betriebssystem.
–10260	dmaChannel3Error	Angabe von einem inkorrekten DMAKanal 3 im Konfigurations-Utility oder durch das Betriebssystem.
–10261	userModeToKernelModeCallError	Der Benutzermoduscode hat beim Aufruf des Kernmodus versagt.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10340	noConnectError	Es ist kein RTSI-Signal/Leitung verbunden, oder das angegebene Signal und die angegebene Leitung sind nicht verbunden.
–10341	badConnectError	Das RTSI-Signal/Leitung kann nicht wie angegeben verbunden werden.
–10342	multConnectError	Das angegebene RTSI-Signal wird bereits von einer RTSI-Leitung angetrieben, oder die angegebene RTSI-Leitung wird bereits von einem RTSI-Signal angetrieben.
–10343	SCXIConfigError	Die angegebenen SCXI-Konfigurationsparameter sind ungültig, oder die Funktion kann mit der aktuellen SCXI-Konfiguration nicht ausgeführt werden.
–10344	chassisSynchedError	Die Remote-SCXI-Einheit ist nicht mit dem Host synchronisiert. Setzen Sie das Chassis zurück, um es mit dem Host neu zu synchronisieren.
–10345	chassisMemAllocError	Die erforderliche Speicherkapazität kann der Remote-SCXI-Einheit für die angegebene Operation nicht zugewiesen werden.
–10346	badPacketError	Das von der Remote-SCXI-Einheit erhaltene Paket ist ungültig. Überprüfen Sie Ihre seriellen Anschlußkabel-Verbindungen.
–10347	chassisCommunicationError	Beim Senden eines Pakets an das Remote-Chassis ist ein Fehler vorgekommen. Überprüfen Sie Ihre seriellen Anschlußkabel-Verbindungen.
–10348	waitingForReprogError	Die Remote-SCXI-Einheit ist im Neuprogrammierungsmodus und wartet auf die Neuprogrammierungsbefehle vom Host (NI-DAQ Konfigurations-Utility).
–10349	SCXIModuleTypeConflictError	Das Modul ID lesen vom SCXI-Modul steht mit dem konfigurierten Modultyp im Widerspruch.
–10370	badScanListError	Die Scanliste ist ungültig; Sie mischen z.B. AMUX-64T-Kanäle und Onboard-Kanäle, tasten SCXI-Kanäle außerhalb der Reihenfolge ab oder haben für das gleiche SCXI-Modul einen anderen Startkanal angegeben. Der Treiber versucht, komplizierte Verstärkungsverteilungen über SCXI-Kanäle am selben Modul durch Manipulation der Scanliste zu erlangen und gibt diesen Fehler zurück, wenn dies fehlschlägt.
–10400	userOwnedRsrcError	Die angegebene Ressource gehört dem Benutzer, und der Treiber kann nicht auf sie zugreifen oder sie modifizieren.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10401	unknownDeviceError	Beim angegebenen Gerät handelt es sich nicht um ein Produkt von National Instruments oder der Treiber unterstützt das Gerät nicht (z.B. wurde der Treiber freigegeben, bevor das Gerät unterstützt wurde).
–10402	deviceNotFoundError	Im angegebenen Steckplatz oder an der angegebenen Adresse konnte kein Gerät gefunden werden.
–10404	noLineAvailError	Es ist keine Leitung verfügbar.
–10405	noChanAvailError	Es ist kein Kanal verfügbar.
–10406	noGroupAvailError	Es ist keine Gruppe verfügbar.
–10407	lineBusyError	Die angegebene Leitung ist in Benutzung.
–10408	chanBusyError	Der angegebene Kanal ist in Benutzung.
–10409	groupBusyError	Die angegebene Gruppe ist in Benutzung.
–10410	relatedLCGBusyError	Eine zusammenhängende Leitung, Kanal oder Gruppe ist in Benutzung; wenn der Treiber die angegebene Leitung, Kanal oder Gruppe konfiguriert, werden Konfiguration, Daten oder Handshaking-Leitungen für die zusammenhänge Leitung, Kanal oder Gruppe gestört sein.
–10411	counterBusyError	Der angegebene Counter ist in Benutzung.
–10412	noGroupAssignError	Es wurde keine Gruppe zugewiesen, oder die angegebene Leitung oder Kanal kann keiner Gruppe zugewiesen werden.
–10413	groupAssignError	Es wurde bereits eine Gruppe zugewiesen, oder die angegebene Leitung oder Kanal wurden bereits einer Gruppe zugewiesen.
–10414	reservedPinError	Für das ausgewählte Signal ist ein Pin notwendig, der nur von NI-DAQ reserviert und konfiguriert wird. Sie können diesen Pin nicht selbst konfigurieren.
–10415	externalMuxSupportError	Diese Funktion unterstützt dieses Gerät nicht, wenn ein externer Multiplexer (wie z.B. ein AMUX-64T oder SCXI) damit verbunden ist.
–10440	sysOwnedRsrcError	Die angegebene Resource gehört dem Treiber, und der Benutzer kann auf diese nicht zugreifen oder sie modifizieren.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10441	memConfigError	Zur Unterstützung des aktuellen Datenübertragungsmodus ist kein Speicher konfiguriert oder der konfigurierte Speicher unterstützt den aktuellen Datenübertragungsmodus nicht. (Bei Benutzung von Blockübertragung muß der Speicher Blockübertragungen vornehmen können.)
–10442	memDisabledError	Der angegebene Speicher ist deaktiviert oder ist mit dem aktuellen Adressiermodus nicht verfügbar.
–10443	memAlignmentError	Der Übertragungspuffer ist nicht ordnungsgemäß auf den aktuellen Datenübertragungsmodus ausgerichtet. Z.B. befindet sich der Puffer an einer ungeraden Adresse, ist nicht auf eine 32-Bit-Grenze oder eine 512-Bit-Grenze ausgerichtet usw. Daher kann der Treiber den Puffer nicht ausrichten, weil der Puffer zu klein ist.
–10445	memLockError	Der Übertragungspuffer kann nicht auf physischen Speicher gesperrt werden. An PC-AT-Maschinen können Teile des DMA- Datenerfassungspuffers in einem ungültigen DMA -Bereich sein, z.B. über 16 MB.
–10446	memPageError	Im Übertragungspuffer ist ein Seitenumbruch enthalten. Die Systemressourcen machen bei Feststellen des Seitenumbruchs evtl. eine Neuprogrammierung notwendig.
–10447	memPageLockError	Die Betriebsumgebung kann keine Seitensperre gewähren.
–10448	stackMemError	Der Treiber kann aufgrund von Stapelbegrenzungen die Stringeingabe nicht weiter untersuchen.
–10449	cacheMemError	Ein mit dem Cache verbundener Fehler ist vorgekommen, oder Caches werden vom aktuellen Modus nicht unterstützt.
–10450	physicalMemError	Im physischen Speicher ist ein Hardwarefehler vorgekommen, oder an der angegebenen Adresse ist kein Speicher vorhanden.
–10451	virtualMemError	Der Treiber kann keinen Übertragungspuffer im virtuellen Speicher zusammenhängen und kann daher den Puffer nicht in einen physischen Speicher sperren. Deshalb können Sie den Puffer nicht für DMA-Übertragungen benutzen.
–10452	noIntAvailError	Es steht kein Interrupt-Level zur Benutzung zur Verfügung.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10453	intInUseError	Der angegebene Interrupt-Level wird bereits von einem anderen Gerät benutzt.
–10454	noDMACError	Für das System ist kein DMA-Kontroller verfügbar.
–10455	noDMAAvailError	Es ist kein DMA-Kanal zur Benutzung verfügbar.
–10456	DMAInUseError	Der angegebene DMA-Kanal wird bereits von einem anderen Gerät benutzt.
–10457	badDMAGroupError	DMA kann für die angegebene Gruppe nicht konfiguriert werden, weil es zu klein, zu groß oder nicht richtig ausgerichtet ist. Sehen Sie im Benutzerhandbuch des betreffenden Geräts nach, um hinsichtlich des DMA Gruppenverzweigungen festzulegen.
–10458	diskFullError	Beim Versuch, in eine Datei zu schreiben, kam ein Laufwerk-Überlauf vor.
–10459	DLLInterfaceError	Wegen eines Schnittstellenfehlers konnte DLL nicht aufgerufen werden.
–10460	interfaceInteractionError	Sie haben VIs von der DAQ-Bibliothek und der _DAQ Kompatibilitäts-Bibliothek gemischt (LabVIEW 2.2 VIs). Sie können zwischen den beiden Bibliotheken nur durch Ausführung von DAQ IV Device Reset vor Aufruf der _DAQ-Kompatibilitäts-VIs oder durch Ausführung von Kompatibilitäts-VI Board Reset vor Aufruf der DAQ-VIs wechseln.
–10480	muxMemFullError	Die Scanliste ist zu lang, um in den Mux-Verstärkungsspeicher auf der Karte zu passen.
–10481	bufferNotInterleavedError	Sie müssen einen einzelnen Puffer versetzter Daten bereitstellen und die Kanäle müssen sich in absteigender Reihenfolge befinden. Sie können DMA nicht zur Übertragung von Daten von zwei Puffern verwenden. Sie können jedoch u.U. Interrupts benutzen.
–10540	SCXIModuleNotSupportedError	Mindestens eines der angegebenen SCXI-Module ist für die Operation nicht unterstützt.
–10541	TRIG1ResourceConflict	CTRB1 wird COUTB1 antreiben. CTRB1 wird jedoch auch TRIG1 antreiben. Durch diesen Konflikt können evtl. unvorhersehbare Ergebnisse verursacht werden, wenn das Chassis abgetastet wird.
–10600	noSetupError	Für die angegebene Ressource wurde keine Einstellungsoperation durchgeführt. Oder einige Ressourcen verlangen für eine ordnungsgemäße Einstellung eine bestimmte Reihenfolge der Aufrufe.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10601	multSetupError	Die angegebenen Ressourcen wurden bereits durch eine Einstellungsoperation konfiguriert.
–10602	noWriteError	In den Übertragungspuffer wurden keine Ausgabedaten geschrieben.
–10603	groupWriteError	Die mit einer Gruppe verbundenen Ausgabedaten müssen für einen einzelnen Kanal oder für aufeinanderfolgende Kanäle sein.
–10604	activeWriteError	Sobald mit der Datenerzeugung begonnen wurde, können nur die Übertragungspuffer, zu denen ursprünglich geschrieben wurde, aktiviert werden. Wenn DMA aktiv ist und ein einzelner Übertragungspuffer versetzte Kanaldaten enthält, müssen für alle Ausgabekanäle, die derzeit den DMA-Kanal benutzen, neue Daten zur Verfügung gestellt werden.
–10605	endWriteError	Zum Übertragungspuffer wurden keine Daten geschrieben, da der endgültige Datenblock bereits geladen wurde.
–10606	notArmedError	Die angegebene Ressource ist nicht verstärkt.
–10607	armedError	Die angegebene Ressource ist bereits verstärkt.
–10608	noTransferInProgError	Für die angegebene Ressource wird derzeit keine Übertragung durchgeführt.
–10609	transferInProgError	Für die angegebene Ressource wird bereits eine Übertragung durchgeführt, oder die Operation ist nicht gestattet, weil das Gerät gerade eine Übertragung - möglicherweise mit anderen Ressourcen - durchführt.
–10610	transferPauseError	Ein einzelner Ausgabekanal in einer Gruppe kann nicht angehalten werden, wenn die Ausgabendaten für die Gruppe überführt sind.
–10611	badDirOnSomeLinesError	Einige Leitungen im angegebenen Kanal sind nicht für die angegebene Übertragungsrichtung vorgesehen. Für eine Schreibübertragung wurden einige Leitungen zur Eingabe konfiguriert. Für eine Leseübertragung wurden einige Leitungen für Ausgabe konfiguriert.
–10612	badLineDirError	Die angegebene Leitung unterstützt nicht die angegebene Übertragungsrichtung.
–10613	badChanDirError	Der angegebene Kanal unterstützt nicht die angegebene Übertragungsrichtung.
–10614	badGroupDirError	Die angegebene Gruppe unterstützt nicht die angegebene Übertragungsrichtung.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10615	masterClkError	Die Takt-Konfiguration für den Takt-Master ist ungültig.
–10616	slaveClkError	Die Takt-Konfiguration für den Takt-Slave ist ungültig.
–10617	noClkSrcError	Für die Takt-Ressource wurde kein Quellensignal zugewiesen.
–10618	badClkSrcError	Das angegebene Quellsignal kann der Takt-Ressource nicht zugewiesen werden.
–10619	multClkSrcError	Der Takt-Ressource wurde bereits ein Quellsignal zugewiesen.
–10620	noTrigError	Der Trigger-Ressource wurde kein Triggersignal zugewiesen.
–10621	badTrigError	Das angegebene Trigger-Signal kann der Trigger-Ressource nicht zugewiesen werden.
–10622	preTrigError	Der Pretrigger-Modus wird nicht unterstützt oder ist in der aktuellen Konfiguration nicht verfügbar, oder es wurde keine Pretrigger-Quelle zugewiesen.
–10623	postTrigError	Es wurde keine Posttrigger-Quelle zugewiesen.
–10624	delayTrigError	Der verzögerte Triggermodus wird nicht unterstützt oder ist in der aktuellen Konfiguration nicht verfügbar oder es wurde keine Verzögerungsquelle zugewiesen.
–10625	masterTrigError	Die Trigger-Konfiguration für den Trigger-Master ist ungültig.
–10626	slaveTrigError	Die Trigger-Konfiguration für den Trigger-Slave ist ungültig.
–10627	noTrigDrvError	Der Trigger-Ressource wurde kein Signal zugewiesen.
–10628	multTrigDrvError	Der Trigger-Ressource wurde bereits ein Signal zugewiesen.
–10629	invalidOpModeError	Der angegebene Betriebsmodus ist ungültig, oder die Ressourcen wurden für den angegebenen Betriebsmodus nicht konfiguriert.
–10630	invalidReadError	Die zur Datenablesung angegebenen Parameter waren im Kontext mit der Erfassung ungültig. Es wurde z.B. ein Versuch unternommen, vom Übertragungspuffer 0 Bytes zu lesen, oder es wurde der Versuch unternommen, über das Ende des Übertragungspuffers hinaus zu lesen.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10631	noInfiniteModeError	Im aktuellen Betriebsmodus sind keine fortlaufenden Eingabe- oder Ausgabeübertragungen gestattet.
–10632	someInputsIgnoredError	Gewisse Eingaben wurden ignoriert, da sie für den aktuellen Betriebsmodus nicht relevant sind.
–10633	invalidRegenModeError	Der angegebene Analogausgabe-Wiedererzeugungsmodus ist für diese Karte nicht gestattet.
–10634	noContTransferInProgressError	Für die angegebene Ressource wird keine fortlaufende (doppelt gepufferte) Übertragung durchgeführt.
–10635	invalidSCXIOpModeError	Entweder ist der SCXI-Betriebsmodus, der in einem Konfigurationaufruf angegeben wurde, ungültig, oder ein Modul befindet sich im falschen Betriebsmodus, um den Funktionsaufruf durchzuführen.
–10636	noContWithSynchError	Sie können keine fortlaufende (doppelt gepufferte) Operation mit einem synchronen Funktionsaufruf starten.
–10637	bufferAlreadyConfigError	Es wurde der Versuch unternommen, einen Puffer zu konfigurieren, nachdem dieser bereits konfiguriert worden war. Sie können einen Puffer nur einmal konfigurieren.
–10680	badChanGainError	Alle Kanäle auf dieser Karte müssen dieselbe Verstärkung haben.
–10681	badChanRangeError	Alle Kanäle auf dieser Karte müssen denselben Bereich haben.
–10682	badChanPolarityError	Alle Kanäle auf dieser Karte müssen dieselbe Polung haben.
–10683	badChanCouplingError	Alle Kanäle auf dieser Karte müssen dieselbe Kopplung haben.
–10684	badChanInputModeError	Alle Kanäle auf dieser Karte müssen denselben Eingabemodus haben.
–10685	clkExceedsBrdsMaxConvRateError	Die ausgewählte Taktfrequenz übersteigt die empfohlene maximale Taktfrequenz für diese Karte.
–10686	scanListInvalidError	Die Scanliste wurde durch eine Konfigurationsänderung ungültig gemacht.
–10687	bufferInvalidError	Der Erfassungspuffer wurde durch eine Konfigurationsänderung ungültig gemacht, oder es wurde noch kein Erfassungspuffer konfiguriert.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10688	noTrigEnabledError	Die Gesamtzahl der Scans und Pretrigger-Scans deutet darauf hin, daß ein Trigger-Start vorgesehen, aber kein Trigger aktiviert ist.
–10689	digitalTrigBError	Der digitale Trigger B ist für die gesamten angegebenen Scans und Pretrigger-Scans illegal.
–10690	digitalTrigAandBError	Diese Karte gestattet die gleichzeitige Aktivierung der digitalen Trigger A und B nicht.
–10691	extConvRestrictionError	Diese Karte gestattet keinen externen Abtasttakt mit einer externen Abtastung, Start-Trigger oder Stopp-Trigger.
–10692	chanClockDisabledError	Die Erfassung kann nicht gestartet werden, da der Kanaltakt deaktiviert ist.
–10693	extScanClockError	Bei der Durchführung einer einzelnen Abtastung eines einzelnen Kanals kann kein externer Abtasttakt verwendet werden.
–10694	unsafeSamplingFreqError	Die Abtastfrequenz übersteigt die sichere maximale Rate für die benutze Hardware, Verstärkung und Filter.
–10695	DMANotAllowedError	Sie haben eine Operation eingestellt, die die Benutzung von Interrupts erforderlich macht. DMA ist nicht gestattet. Für einige DAQ-Ereignisse, wie Mitteilungen und LabVIEW-Occurrences, sind z.B. Interrupts notwendig.
–10696	multiRateModeError	Die Abtastung mit Mehrfachtakt kann nicht mit AMUX-64, SCXI oder Pretrigger-Erfassungen durchgeführt werden.
–10697	rateNotSupportedError	NI-DAQ konnte Ihr Zeitbasis-/Intervall-Paar nicht konvertieren, um den tatsächlichen Hardware-Fähigkeiten auf der angegebenen Karte zu entsprechen.
–10698	timebaseConflictError	Für die angegebene Karte können Sie diese Kombination von Scan- und Abtasttakt-Zeitbasen nicht benutzen.
–10699	polarityConflictError	Für diese Operation und Karte können Sie die Kombination von Scan- und Abtasttakt-Quellpolaritäten nicht benutzen.
–10700	signalConflictError	Für diese Operation und Karte können Sie die Kombination von Scan- und Konvertierungs-Taktsignalquellen nicht benutzen.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10701	noLaterUpdateError	Der Aufruf hatte keine Auswirkung, da der angegebene Kanal nicht auf spätere interne Aktualisierungen eingestellt wurde.
–10702	prePostTriggerError	Pretriggering und Posttriggering kann auf den Lab- und Serie-1200-Geräten nicht gleichzeitig benutzt werden.
–10710	noHandshakeModeError	Der angegebene Anschluß wurde nicht für den Handshake konfiguriert.
–10720	noEventCtrError	Der angegebene Counter ist nicht zur Ereigniszählungsoperation konfiguriert.
–10740	SCXITrackHoldError	Der SCXI-Track- und Halte-Triggerleitung wurde bereits ein Signal zugewiesen oder ein Kontrollaufruf war ungeeignet, weil das angegebene Modul nicht für eine Einkanal-Operation konfiguriert ist.
–10780	sc2040InputModeError	Wenn Sie Ihr Gerät mit einem SC-2040 verbunden haben, müssen alle analogen Eingabekanäle für den differentialen Eingabemodus konfiguriert werden.
–10781	outputTypeMustBeVoltageError	Die Polung des Ausgabekanals kann bei der Stromausgabe nicht bipolar sein.
–10782	sc2040HoldModeError	Die angegebene Operation kann nicht durchgeführt werden, wenn SC-2040 im Haltemodus konfiguriert ist.
–10783	calConstPolarityConflictError	Kalibrierkonstanten im Ladebereich haben eine von der aktuellen Stromkonfiguration unterschiedliche Polung. Deshalb sollten Sie die Konstanten vom Werk laden.
–10800	timeOutError	Die Operation konnte nicht innerhalb der Zeitbegrenzung abgeschlossen werden.
–10801	calibrationError	Während des Kalibriervorgangs ist ein Fehler vorgekommen.
–10802	dataNotAvailError	Die angeforderte Datenmenge wurde noch nicht erfaßt.
–10803	transferStoppedError	Die Übertragung wurde gestoppt, um die Wiedererzeugung von Ausgabedaten zu verhindern.
–10804	earlyStopError	Die Übertragung hat vor Erreichen vom Endes des Übertragungspuffers gestoppt.
–10805	overRunError	Die Taktquelle für den Eingabetakt ist schneller als die der maximalen Taktfrequenz, die das Gerät unterstützt. Wenn Sie dem Treiber gestatten, die maximale analoge Eingabekanal-Taktfrequenz zu berechnen, stützt der Treiber die Taktfrequenz auf den Kartentyp. Sie sollten daher prüfen, daß Ihr Kartentyp im Konfigurations-Utility korrekt ist.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10806	noTrigFoundError	Im Eingabeübertragungspuffer wurde kein Trigger-Wert gefunden
–10807	earlyTrigError	Der Trigger fand statt, ehe ausreichend Trigger-Daten erfaßt werden konnten.
–10808	LPTCommunicationError	In der Kommunikation mit dem DAQ-Gerät ist ein Fehler über den parallelen Anschluß vorgekommen.
–10809	gateSignalError	Es wurde versucht, eine Pulsbreitenmessung mit dem Puls in der zu messenden Phase zu starten (z.B. hohe Phase für High-Level-Gatterung).
–10840	internalDriverError	Im Treiber ist bei der Ausführung der gegebenen Operation ein unerwarteter Fehler vorgekommen.
–10841	firmwareError	Die Firmware unterstützt die angegebene Operation nicht, oder die Firmware-Operation konnte aufgrund eines Datenintegritätsproblems nicht abschließen.
–10842	hardwareError	Die Hardware antwortet auf die angegebene Operation nicht oder die Antwort von der Hardware stimmt nicht mit den Gebrauchseigenschaften der Hardware überein.
–10843	underFlowError	Aufgrund von Systembegrenzungen konnte der Treiber Daten nicht schnell genug zum Gerät schreiben, um mit dem Durchsatz des Geräts Schritt halten zu können.
–10844	underWriteError	Es wurden keine neue Daten an den Ausgabeübertragungspuffer geschrieben, ehe der Treiber die Datenübertragung an das Gerät versucht hat.
–10845	overFlowError	Aufgrund von Systembeschränkungen konnte der Treiber Daten vom Gerät nicht schnell genug lesen, um mit dem Durchsatz des Geräts Schritt halten zu können. Der Onboard-Gerätespeicher hat einen Überlauffehler gemeldet.
–10846	overWriteError	Der Treiber hat neue Daten zum Eingabeübertragungspuffer geschrieben, bevor die vorher erfaßten Daten gelesen wurden.
–10847	dmaChainingError	Zum Zeitpunkt des DMA-Verkettungsinterrupts war keine Pufferinformation verfügbar; DMA-Übertragungen werden am Ende des derzeit aktiven Übertragungspuffers beendet
–10848	noDMACountAvailError	Der Treiber konnte vom Übertragungszähl-Register im DMA-Kontroller keine gültige Ablesung erhalten.
–10849	openFileError	Die Konfigurationsdatei konnte nicht geöffnet werden.
–10850	closeFileError	Eine Datei kann nicht geschlossen werden.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10851	fileSeekError	Innerhalb einer Datei kann keine Suche vorgenommen werden.
–10852	readFileError	Aus einer Datei kann nicht gelesen werden.
–10853	writeFileError	Zu einer Datei kann nicht geschrieben werden.
–10854	miscFileError	Beim Zugriff auf eine Datei kam ein Fehler vor.
–10855	osUnsupportedError	NI-DAQ unterstützt die aktuelle Operation auf dieser bestimmten Version des Betriebssystems nicht.
–10856	osError	Vom Betriebssystem ist bei der Durchführung der gegebenen Operation ein unerwarteter Fehler vorgekommen.
–10857	internalKernelError	Während der Durchführung dieser Operation ist im Kern ist ein unerwarteter Fehler vorgekommen.
–10880	updateRateChangeError	Derzeit ist es aus einem der folgenden Gründe nicht möglich, die Updaterate zu ändern: 1. Bei Durchführen der Signalverlauferzeugung können Sie die Intervall-Zeitbasis nicht ändern. 2. Wenn Sie mehrere Änderungen nacheinander vornehmen, müssen Sie jeder Änderung ausreichend Zeit geben, wirksam zu werden, bevor Sie weitere Änderungen anfordern.
–10881	partialTransferCompleteError	Nach einer erfolgreichen Teilübertragung können Sie keine weitere Übertragung vornehmen.
–10882	daqPollDataLossError	Die auf dem Remote-SCXI-Gerät erfaßten Daten wurden überschrieben, bevor sie zum Puffer im Host übertragen werden konnten. Versuchen Sie, wenn möglich, eine langsamere Datenerfassungsrate zu verwenden.
–10883	wfmPollDataLossError	Zum Signalverlaufpuffer konnten keine neue Daten übertragen werden, um mit dem Signalverlauf-Update auf dem laufenden zu bleiben. Versuchen Sie, wenn möglich, eine langsamere Signalverlauf-Updaterate zu verwenden.
–10884	pretrigReorderError	Daten konnten nach abgeschlossener Pretrigger-Erfassung nicht neu angeordnet werden.
–10920	gpctrDataLossError	Während gepufferter GPCTR-Operationen sind u.U. ein oder mehrere Datenpunkte aufgrund der Geschwindigkeitsbeschränkungen auf Ihrem System verloren gegangen.

Tabelle A-4. Datenerfassungs-VI-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–10940	chassisResponseTimeoutError	Innerhalb der angegebenen Zeitbegrenzung ist von der Remote-SCXI-Einheit keine Antwort eingegangen.
–10941	reprogrammingFailedError	Die Neuprogrammierung der SCXI-Einheit war erfolglos. Bitte versuchen Sie es noch einmal.
–10942	invalidResetSignatureError	Vom Host wurde eine ungültige Zurücksetzen-Signatur an die Remote-SCXI-Einheit gesandt.
–10943	chassisLockupError	Die Interrupt-Serviceroute auf der Remote-SCXI-Einheit dauert länger als notwendig. Sie brauchen Ihre Remote-SCXI-Einheit nicht zurückzusetzen; jedoch müssen Sie Ihre Datenerfassung löschen und neu starten.

Tabelle A-5. AppleEvent-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
–1700	errAECoercionFail	Die Formatumwandlung der Daten konnte nicht an den angeforderten Deskriptortyp erfolgen.
–1701	errAEDescNotFound	Descriptor-Aufzeichnung wurde nicht gefunden.
–1702	errAECorruptData	Daten in einem AppleEvent konnten nicht gelesen werden.
–1703	errAEWrongDataType	Falscher Deskriptortyp.
–1704	errAENotAEDesc	Keine gültige Deskriptoraufzeichnung.
–1705	errAEBadListItem	Operation mit einem Listenelement fehlgeschlagen.
–1706	errAENewerVersion	Neuere Version vom AppleEvent Manager notwendig.
–1707	errAENotAppleEvent	Das Ereignis ist kein AppleEvent.
–1708	errAEReplyNotValid	An den AEResetTimer wurde ein ungültiger Antwortparameter weitergegeben.
–1709	errAEERReplyNotValid	An den AEResetTimer wurde ein ungültiger Antwortparameter weitergegeben.
–1710	errAEUnknownSendMode	Ungültiger Sendemodus wurde weitergegeben.
–1711	errAEWaitcanceled	Benutzer hat die Position in der Warteschleife zum Erhalt einer Antwort oder einer Bestätigung beendet.
–1712	errAETimeout	AppleEvent-Zeitablauf erreicht.
–1713	errAENoUserInteraction	Benutzerinteraktion nicht erlaubt.
–1714	errAENotASpecialFunction	Falsches Schlüsselwort für eine spezielle Funktion.

Tabelle A-5. AppleEvent-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
–1715	errAEParamMissed	Handler hat nicht alle notwendigen Parameter erhalten.
–1716	errAEUnknownAddressType	Unbekannter AppleEvent-Adressentyp.
–1717	errAEHandlerNotFound	Kein Handler in den Dispatchtabellen entspricht den Parametern von AEGetEventHandler oder AEGetCoercionHandler.
–1718	errAEReplyNotArrived	Der Inhalt der Antwort, auf die Sie zugreifen, ist noch nicht eingetroffen.
–1719	errAEIllegalIndex	Index ist in einer gegebenen Operation außerhalb des Bereichs.

Tabelle A-6. Instrumententreiber-Fehlercodes

Status	Statusbeschreibung	Eingestellt von
0	Kein Fehler; der Aufruf war erfolgreich	—
–1200	Ungültiger Syntaxstring	VISA Transition Library
–1201	Fehler beim Finden des Instruments	VISA Transition Library
–1202	Unfähig, Schnittstelle oder Instrument zu initialisieren	VISA Transition Library
–1205	Ungültiges Instrumenten-Handle	VISA Transition Library
–1210	Parameter außerhalb des Bereichs	Instrumententreiber-VI
–1215	Fehler beim Schließen des Instruments	VISA Transition Library
–1218	Fehler bei Erlangung des Attributs	VISA Transition Library
–1219	Fehler bei Einstellen des Attributs	VISA Transition Library
–1223	Instrumenten-ID-Abfrage fehlgeschlagen	Instrumententreiber-Initialisierungs-VI
–1224	Fehler bei Löschung von Instrument	VISA Transition Library
–1225	Fehler bei Triggerung von Instrument	VISA Transition Library
–1226	Fehler bei Abfrage an Instrument	VISA Transition Library
–1230	Fehler beim Schreiben an Instrument	VISA Transition Library
–1231	Fehler beim Lesen von Instrument	VISA Transition Library
–1236	Fehler bei Interpretation von Instrumentenantwort	Instrumententreiber-VI
–1240	Instrument-Zeitablauf erreicht	VISA Transition Library

Tabelle A-6. Instrumententreiber-Fehlercodes (Fortsetzung)

Status	Statusbeschreibung	Eingestellt von
–1250	Fehler beim Zuordnen der VXI-Adresse	VISA Transition Library
–1251	Fehler bei der Aufhebung der Skalierung der VXI-Adresse auf	VISA Transition Library
–1252	Fehler beim Zugreifen auf VXI-Adresse	VISA Transition Library
–1260	Funktion von Kontroller nicht unterstützt	VISA Transition Library
–1300	Instrumenentspezifischer Fehler	—
–13xx	Entwicklerspezifizierte Fehlercodes	—

Tabelle A-7. PPC-Fehlercodes

Code	Name	Beschreibung
–900	notInitErr	PPC-Toolbox wurde nicht initialisiert.
–902	nameTypeErr	Ungültiger oder unpassender locationKindSelector in locationName.
–903	noPortErr	Ungültiger Anschlußname. Unfähig, Anschluß oder ungültige portRefNum zu öffnen.
–904	noGlobalsErr	Das System kann keinen Speicher zuweisen. Dies ist ein kritischer Fehler; Sie sollten neu starten.
–905	localOnlyErr	Netzwerk-Aktivität ist derzeit deaktiviert.
–906	destPortErr	Anschluß existiert am Ziel nicht.
–907	sessTableErr	PPC-Toolbox ist unfähig, eine Session zu erstellen.
–908	noSessionErr	Ungültige Session-Referenznummer.
–909	badReqErr	Ungültiger Parameter oder ungültiger Status für diese Operation.
–910	portNameExistsErr	Ein anderer Anschluß mit diesem Namen ist bereits geöffnet (vielleicht in einer anderen Anwendung).
–911	noUserNameErr	Benutzername auf Zielmaschine unbekannt.
–912	userRejectErr	Ziel hat diese Sitzungsanforderung abgelehnt.
–913	noMachineNameErr	Benutzer hat seinen Macintosh im Network Setup Control Panel nicht benannt.
–914	noToolboxNameErr	Eine Systemressource fehlt.
–915	noResponseErr	Unfähig, Zielanwendung zu kontaktieren.
–916	portClosedErr	Der Anschluß war geschlossen.

Tabelle A-7. PPC-Fehlercodes (Fortsetzung)

Code	Name	Beschreibung
–917	sessClosedErr	Die Session war geschlossen.
–919	badPortNameErr	PPCPortRec ist ungültig.
–922	noDefaultUserErr Macintosh: Eigentümer	Benutzer hat im Sharing Setup Control Panel keinen Inhabernamen angegeben.
–923	notLoggedInErr	Die Standardvorgabe userRefNum existiert noch nicht.
–924	noUserRefErr	Unfähig, eine neue userRefNum zu erstellen.
–925	networkErr	Im Netzwerk ist ein Fehler vorgekommen.
–926	noInformErr	PPCStart fehlgeschlagen, da Ziel keine Information erwartet.
–927	authFailErr	Paßwort des Benutzers ist falsch.
–928	noUserRecErr	Ungültige Benutzerreferenznummer.
–930	badServiceMethodErr	Die angewandte Servicemethode ist nicht ppcServiceRealTime.
–931	badLocNameErr	Der Positionsname ist ungültig.
–932	guestNotAllowedErr	Für Zielanschluß ist eine Überprüfung der Identität erforderlich.

Tabelle A-8. GPIB-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
0	EDVR	Fehler bei Verbindung mit Treiber.
1	ECIC	Bei Befehl ist es erforderlich, daß GPIB-Controller ein CIC (controller-in-charge) ist.
2	ENOL	Schreiben hat keine Listeners festgestellt.
3	EADR	GPIB-Controller nicht richtig adressiert.
4	EARG	Ungültiges Argument oder Argumente.
5	ESAC	Bei Befehl ist es erforderlich, daß GPIB-Controller ein SCist.
6	EABO	I/O-Operation abgebrochen.
7	ENEB	Nichtexistente Karte.
8	EDMA	DMA-Hardware-Fehler entdeckt.
9	EBTO	DMA-Hardware µP Bus-Zeitablauf.
11	ECAP	Keine Fähigkeiten.

Tabelle A-8. GPIB-Fehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
12	EFSO	Dateisystem-Operationsfehler.
13	EOWN	Gemeinsam benutzbare Karte in ausschließlichem Besitz.
14	EBUS	GPIB-Busfehler.
15	ESTB	Serieller Poll-Byte-Queue-Überlauf.
16	ESRQ	SRQ anhaftend.
17	ECMD	Unerkannter Befehl.
19	EBNP	Karte nicht vorhanden.
20	ETAB	Tabellenfehler.
30	NADDR	Keine GPIB-Adresseneingabe.
31	NSTRG	Keine Stringeingabe (schreiben).
32	NCNT	Keine Zähleingabe (lesen).

Tabelle A-9. LabVIEW-Funktionsfehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
0	—	Kein Fehler.
1	—	Manager-Argumentfehler.
2	—	Argumentfehler.
3	—	Außerhalb der Zone.
4	—	Ende der Datei.
5	—	Datei bereits offen.
6	—	Allgemeiner Datei-I/O-Fehler.
7	—	Datei nicht gefunden.
8	—	Dateibefugnisfehler.
9	—	Speichermedium voll.
10	—	Duplizierter Pfad.
11	—	Es sind zu viele Dateien offen.
12	—	Systemfunktion nicht aktiviert.
13	—	Ressourcendatei nicht gefunden.

Tabelle A-9. LabVIEW-Funktionsfehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
14	—	Ressource kann nicht hinzugefügt werden.
15	—	Ressource nicht gefunden.
16	—	Bild nicht gefunden.
17	—	Bildspeicherfehler.
18	—	Stift existiert nicht.
19	—	Konfigurationstyp ungültig.
20	—	Konfigurationstoken nicht gefunden.
21	—	Konfigurations-Untersuchungsfehler.
22	—	Konfigurations-Speicherfehler.
23	—	Ungültige externe Codeformatierung.
24	—	Ungültiges externes Code-Offset.
25	—	Externer Code nicht vorhanden.
26	—	Null-Fenster.
27	—	Fensterfehler zerstören.
28	—	Null-Menü.
29	—	Drucken abgebrochen.
30	—	Ungültige Druckaufzeichnungen.
31	—	Druckertreiberfehler.
32	—	Windows-Fehler während des Druckvorgangs.
33	—	Speicherfehler während des Druckvorgangs.
34	—	Druckdialogfehler.
35	—	Allgemeiner Druckfehler.
36	—	Ungültige Geräte-Refnum.
37	—	Gerät nicht gefunden
38	—	Geräteparameterfehler.
39	—	Geräteeinheitenfehler.
40	—	Gerät kann nicht geöffnet werden.
41	—	Geräteaufruf abgebrochen.
42	—	Allgemeiner Fehler.

Tabelle A-9. LabVIEW-Funktionsfehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
43	—	Vom Benutzer abgebrochen.
44	—	Objekt-ID zu lang.
45	—	Objekt-ID zu hoch.
46	—	Objekt nicht im Binärbaum.
47	—	Unbekannter Binärbaum.
48	—	Unbekanntes Objekt (ungültiger DefProc).
49	—	Unbekanntes Objekt (DefProc nicht in Tabelle).
50	—	Nachricht außerhalb des Bereichs.
51	—	Ungültige (null) Methode.
52	—	Unbekannte Nachricht.
53	—	Manageraufruf nicht unterstützt.
54	—	Ungültige Adresse.
55	—	Verbindung wird vorgenommen.
56	—	Verbindung durch Zeitablauf beendet.
57	—	Verbindung wird bereits vorgenommen.
58	—	Netzwerkattribute nicht unterstützt.
59	—	Netzwerkfehler.
60	—	Adresse in Benutzung.
61	—	System hat keinen Speicherplatz mehr.
62	—	Verbindung abgebrochen.
63	—	Verbindung abgelehnt.
64	—	Nicht verbunden.
65	—	Bereits verbunden.
66	—	Verbindung geschlossen.
67	—	Initialisierungsfehler (Interapplication-Manager).
68	—	Ungültige Occurrence.
69	—	Auf unbegrenzten Occurrence-Handler warten.
70	—	Occurrence-Queue-Überlauf.
71	—	Konflikt vom Datenprotokolltyp.

Tabelle A-9. LabVIEW-Funktionsfehlercodes (Fortsetzung)

Fehlercode	Fehlerbezeichnung	Beschreibung
72	—	Unbenutzt.
73	—	Unerkannter Typ (Interapplication-Manager).
74	—	Speicher korrumpiert.
75	—	Erstellung temporärer DLL fehlgeschlagen.
76	—	Alte CIN-Version.
77	—	Unbekannter Fehlercode.
81	—	Nicht übereinstimmender Formatidentifizierertyp.
82	—	Unbekannter Formatidentifizierer.
83	—	Zu wenige Formatspezifizierer.
84	—	Zu viele Formatidentifizierer.
85	—	Scan fehlgeschlagen.

Tabelle A-10. LabVIEW-Spezifische PPC-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
1	errNoPPCToolBox	Die PPC ToolBox existiert entweder nicht (benötigt System 7.0 oder später) oder konnte nicht initialisiert werden.
2	errNoGlobals	Der CIN im PPC-VI konnte seine globalen Variablen nicht erlangen.
3	errTimedOut	Die PPC-Operation hat seine Zeitablaufbegrenzung überschritten.
4	errAuthRequired	Für das im PPC Start Session VI angegebene Ziel ist eine Genehmigung notwendig, aber der Genehmigungsdialog war unzulässig.
5	errbadState	Das PPC Start Session VI fand sich in einem unerwarteten Status.

Tabelle A-11. TCP- und UDP-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
53	mgNotSupported	LabVIEW Manager-Aufruf nicht unterstützt.
54	ncBadAddressErr	Die Netzadresse war schlecht aufgebaut.
55	ncInProgressErr	Operation wird durchgeführt.
56	ncTimeOutErr	Operation hat die vom Benutzer festgelegte Zeitgrenze überschritten.
57	ncBusyErr	Die Verbindung war belegt.
58	ncNotSupportedErr	Funktion nicht unterstützt.
59	ncNetErr	Das Netzwerk ist außer Betrieb, kann nicht erreicht werden oder wurde zurückgesetzt.
60	ncAddrInUseErr	Die angegebene Adresse ist derzeit in Benutzung.
61	ncSysOutOfMemErr	System konnte keinen notwendigen Speicher zuweisen.
62	ncSysConnAbortedErr	System hat den Abbruch der Verbindung veranlaßt.
63	ncConnRefusedErr	Verbindung ist nicht eingerichtet.
65	ncAlreadyConnectedErr	Verbindung ist bereits eingerichtet.
66	ncConnClosedErr	Verbindung wurde durch gleichrangiges Gerät geschlossen.

Tabelle A-12. Serielle Anschluß-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
61	EPAR	Paritätsfehler des seriellen Anschlusses.
62	EORN	Überlauffehler des seriellen Anschlusses.
63	EOFL	Serieller Anschluß erhält Pufferüberlauf.
64	EFRM	Rahmenbildungsfehler des seriellen Anschlusses.
65	SPTMO	Zeitablauf des seriellen Anschlusses, Bytes nicht beim seriellen Anschluß eingegangen.

Tabelle A-13. LabVIEW-spezifischer Fehlercodes für AppleEvent-Nachrichten

Fehlercode	Fehlerbezeichnung	Beschreibung
1000	kLVE_InvalidState	Das VI befindet sich in einem Zustand, der dessen Ausführung nicht erlaubt.
1001	kLVE_FPNotOpen	Das VI-Frontpanel ist nicht offen.
1002	kLVE_CtrlErr	Das VI hat Bedienelemente in seinem Frontpanel, die sich in einem Fehlerzustand befinden.
1003	kLVE_VIBad	Das VI ist beschädigt.
1004	kLVE_NotInMem	Das VI befindet sich nicht im Speicher.

Tabelle A-14. DDE-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
00000	—	Kein Fehler.
14001	DDE_INVALID_REFNUM	Ungültige Refnum.
14002	DDE_INVALID_STRING	Ungültiger String.
14003	DDEML_ADVACKTIMEOUT	Zeitablauf für Anforderung der synchronen Advise-Action wurde erreicht.
14004	DDEML_BUSY	DDE_FBUSY-Bit von Antwort eingestellt.
14005	DDEML_DATAACKTIMEOUT	Zeitablauf für Anforderung der synchronen Datenübertragung wurde erreicht.
14006	DDEML_DDL_NOT_INITIALIZED	Aufruf von DDEML, ohne zuerst DdeInitialize aufzurufen, oder es wurde ein ungültiger Instanzidentifizierer weitergegeben.
14007	DDEML_DLL_USAGE	Eine Anzeige- oder Nur-Client-Anwendung hat den Versuch einer DDE-Übertragung unternommen.
14008	DDEML_EXECACKTIMEOUT	Zeitablauf für Anforderung einer synchronen Übertragungsausführung wurde erreicht.
14009	DDEML_INVALIDPARAMETER	Parameter von DDML nicht bestätigt.
14010	DDEML_LOW_MEMORY	Serveranwendung hat den Clienten überholt und verbraucht einen großen Teil des Speichers.
14011	DDEML_MEMORY_ERROR	Eine Speicherzuweisung ist fehlgeschlagen.
14012	DDEML_NOTPROCESSED	Anforderung oder Poke ist für ungültiges Objekt.
14013	DDEML_NO_CONV_ESTABLISHED	Client-Konversationsversuch fehlgeschlagen.
14014	DDEML_POKEACTIMEOUT	Übertragung fehlgeschlagen.

Tabelle A-14. DDE-Fehlercodes

Fehlercode	Fehlerbezeichnung	Beschreibung
14015	DDEML_POSTMSG_FAILED	Zeitablauf für Anforderung der synchronen Pokeübertragung wurde erreicht.
14016	DDEML_REENTRANCY	Eine Anwendung mit einer in Durchführung befindlichen synchronen Übertragung hat den Versuch unternommen, eine andere Übertragung zu initiieren oder eine DDEML-Rückruffunktion hat `DdeEnableCallback` aufgerufen.
14017	DDEML_SERVER_DIED	Übertragung von Seiten des Servers bei von Clienten beendeter Konversation versucht, oder der Service wurde vor Abschluß der Übertragung beendet.
14018	DDEML_SYS_ERROR	Interner Fehler im DDMEML.
14019	DDEML_UNADVACKTIMEOUT	Zeitablauf für Anforderung, Advise zu beenden, wurde erreicht.
14020	DDEML_UNFOUND_QUEUE_ID	Ungültiger Übertragungsidentifizierer an DDEML-Funktion weitergeleitet.
14021	—	Ungültiger Befehlscode.
14022	—	Zeitablauf der Occurrence; Zeitablauf der Übertragung vor Fertigstellung erreicht.

B

DAQ-Hardware-Leistungsfähigkeit

In diesem Anhang sind Tabellen enthalten, in denen die analoge und digitale I/O-Leistungsfähigkeit der Datenerfassungsgeräte (DAQ) von National Instruments zusammengefaßt ist. Die in diesem Anhang aufgeführten Geräte sind in Kategorien eingeteilt. Die DAQ-Gerätekategorien für diese Tabellen umfassen:

- MIO- und AI-Geräte
- Lab- und Serie-1200-Geräte sowie tragbare Geräte
- Serie-54*xx-Geräte*
- SCXI-Module
- Dynamische Signalerfassungsgeräte
- Geräte nur für Analogausgabe
- Nur Digitalgeräte
- Nur Timing-Geräte
- Hardware-Fähigkeiten der 5102-Geräte

☞ **Hinweis** **(Macintosh)** ***Wenn ein NuBus-Gerät anzeigt, daß es DMA-Übertragungen unterstützt, ist auch ein DMA-Gerät (z.B. ein NB-DMA2800) notwendig.***

Hardware-Leistungsfähigkeit von MIO- und AI-Geräten

Tabelle B-1. Konfigurationsprogrammierbarkeit der Analogeingabe—MIO- und AI-Geräte

Gerät	Verstärkung	Bereich	Polung	SE/DIFF	Kopplung
Alle Geräte der MIO-E-Serie Alle Geräte der AI-E-Serie	Nach Kanal	Nach Kanal	Nach Kanal	Nach Kanal	DC AC, DC (für PCI-6110E, PCI-6111E)
AT-MIO-16F-5	Nach Kanal	Nach Gruppe	Nach Gruppe	Nach Gruppe	DC
AT-MIO-64F-5 AT-MIO-16X	Nach Kanal	Nach Kanal	Nach Kanal	Nach Kanal	DC
AT-MIO-16/16D NB-MIO-16 NB-MIO-16X	Nach Kanal	Nach Gerät	Nach Gerät	Nach Gerät	DC

Nach Gerät bedeutet, daß Sie den Wert eines Parameters mit Hardware-Jumpern (Steckbuchsen) auswählen, wobei sich die Auswahl auf jede Kanalgruppe auf dem Gerät auswirkt. *Nach Gruppe* bedeutet, daß Sie die Auswahl mittels Software programmieren, wobei sich die Auswahl auf alle gleichzeitig benutzten Kanäle auswirkt. *Nach Kanal* bedeutet, daß Sie die Auswahl mit Hardware-Jumpern oder mittels Software auf für je einen Kanal programmieren. Wenn für einen Parameter ein spezifischer Wert angezeigt wird, ist dieser Wert fixiert.

Tabelle B-2. Analogeingangs-Eigenschaften—MIO- und AI-Geräte (Teil 1)

Gerät	Anzahl der Kanäle	Auflösung	Verstärkung[1]	Bereich(V)[1]	Eingabe-FIFO (Worte)	Scanning[2]
AT-MIO-16E-1 AT-MIO-16E-2 AT-MIO-16E-10 AT-MIO-16DE-10 NEC-MIO-16E-4 PCI-MIO-16E-1 PCI-MIO-16E-4 NEC-AI-16E-4	16SE, 8DI	12 Bits	0,5; 1; 2; 5; 10; 20; 50; 100	±5, 0 bis10	512; E-1: 8,192; E-2 und E4: 2,048	Bis zu 512
PCI-6110E PCI-6111E	4 DI 2 DI	12 Bits 12 Bits	0,2; 0,5; 1,2; 5; 10; 20; 50	±10	512	Bis zu 4 Bis zu 2

Tabelle B-2. Analogeingangs-Eigenschaften—MIO- und AI-Geräte (Teil 1) (Fortsetzung)

Gerät	Anzahl der Kanäle	Auflösung	Verstärkung[1]	Bereich(V)[1]	Eingabe-FIFO (Worte)	Scanning[2]
AT-MIO-64E-3*	64SE, 32DI	12 Bits	0,5; 1; 2; 5; 10; 20; 50; 100	±5, 0 bis10	2,048	Bis zu 512
PCI-MIO-16XE-10	16SE, 8DI	16 Bits	1; 2; 5; 10; 20; 50;100	±10, 0 bis 10	512	Bis zu 512
NEC-MIO-16XE-50 NEC-AI-16XE-50 AT-MIO-16XE-50 DAQPad-MIO-16XE-50 PCI-MIO-16XE-50	16SE, 8DI	16 Bits	1; 2;10; 100	±10, 0 bis 10	512	Bis zu 512
AT-MIO-16F-5 AT-MIO-64F-5**	16SE, 8DI 64SE, 32DI	12 Bits	0,5; 1; 2; 5; 10; 20; 50; 100	±5; ±10, 0 bis 10	16F-5: 256; 64F-5: 512	Bis zu 512
AT-MIO-16X	16SE, 8DI	16 Bits	1; 2; 5; 10; 20; 50; 100	±10, 0 bis10	512	Bis zu 512
AT-MIO-16(L) AT-MIO-16(H) AT-MIO-16D(L) AT-MIO-16D(H)	16SE, 8DI	12 Bits	(L) 1; 10; 100; 500; (H): 1; 2; 4, 8	±5, ±10, 0 bis 10	16 (L,H); 512 (DL, DH)	Bis zu 16
NB-MIO-16 NB-MIO-16X	16SE, 8DI	MIO-16: 12; MIO-16X: 16	(L) 1; 10; 100; 500; (H) 1; 2; 4; 8	±10; ±5, 0 bis 10, 0 bis 5	16; MIO-16, Rev. G: 512	Bis 16; MIO-16: Gruppe von 2, 4, 8 und 16

[1] Sie können die Grenzeinstellungen für Ihr Gerät bestimmen, indem Sie die Bereichs- und die Spannungswerte miteinander multiplizieren. Weitere Informationen zu Grenzeinstellungen in LabVIEW finden Sie in Kapitel 3, *Grundlegende Konzepte zur Datenerfassung*, im *Grundlagen der Datenerfassung mit LabVIEW.*

[2] Scanning = Kanäle, in beliebiger Reihenfolge.

* Die gültigen Kanäle für AT-MIO-64E-3 im Differential-Modus sind 0–7, 16–23, 32–39 und 48–55.

** Die gültigen Kanäle für AT-MIO-64F-5 im Differential-Modus sind 0–7 und 16–39.

Tabelle B-3. Analogeingangs-Eigenschaften—MIO- und AI-Geräte (Teil 2)

Gerät	Trigger[1]	Max. Abtastrate(S/s)	Übertragungsmethode
AT-MIO-16E-1 AT-MIO-16E-2 AT-MIO-64E-3 AT-MIO-16E-10 AT-MIO-16DE-10 PCI-MIO-16E-1 PCI-MIO-16XE-10 NEC-AI-16E-4 NEC-MIO-16E-4 PCI-MIO-16E-4 PCI-6110E PCI-6111E	SW, Pre, Post (und Analog zu E-1, E-2, E-3, E-4, PCI-6110E, und PCI-6111E)	E-1: 1 M, E-2 und E-3: 500 k, E-4: 250 k, E-10 und DE-10: 100 k PCI-6110E und PCI-6111E: 5M	DMA, Interrupts
All MIO-16XE-50 Devices NEC-AI-16XE-50	SW, Pre, Post	20 k	DMA, (Interrupts auf DAQPad-MIO-16XE-50)
AT-MIO-16F-5 AT-MIO-64F-5	SW, Pre, Post	200 k	DMA, Interrupts
AT-MIO-16X AT-MIO-16/16D	SW, Pre, Post	100 k	DMA, Interrupts
NB-MIO-16	SW, Post	111 k (L-9 oder H-9), 67 k (L-15 oder H-15), 40 k (L-25 oder H-25)	DMA, Interrupts
NB-MIO-16X	SW, Post	55 k (L-18 oder H-18), 24 k (L-42 oder H-42)	DMA, Interrupts

[1] SW=Software-Trigger (auch bedingter Abruf genannt), Pre=Pretrigger, Post=Posttrigger.

☞ **Hinweis** ***Bei NB-MIO-Geräten wird das Software-Triggering in der Interrupt-Serviceroutine durchgeführt (nur Interrupts) und unterscheidet sich vom bedingten Abruf.***

Tabelle B-4. Interne Kanalunterstützung—MIO- und AI-Geräte

Geräte	Interne Kanäle
AT-MIO-16XE10, AT-MIO-16XE-50, NEC-MIO-16XE-50, DAQPad-MIO-16XE-50	IntAIGnd, IntRef5V, IntAOGndVsAIGnd, IntAOCh0, IntAOCh0VsRef5V, IntA0Ch1, IntAOCh1VsRef5V
DAQCard-AI-16E-4, NEC-AI-16E-4	IntAIGnd, IntRef5V, IntA0GndVsAIGnd
PCI-MIO-16XE-10, PCI-MIO-16XE-50, PXI-6030E, PXI-6011E, PCI-6031, CPCI-6030E, CPCI-6011E, VXI-MIO-64XE-10	IntAIGnd, IntRef5V, IntAOGndVsAIGnd, IntAOch0, IntAOCh0VsRef5V, IntAOCh1, IntAOCh1VsRef5V, IntAOChVsAOch0, IntDevTemp
PCI-MIO-16E-1, PCI-MIO-16E-4, PXI-6070E, PXI-6040E, PCI-6071E, CPCI-6070E, CPCI-6040E, VXI-MIO-64E-1	IntAIGnd, IntRef5V, IntCmRef5V, IntAOGndVsAIGnd, IntAOCh0, IntAOCh0VsRef5V, IntAOCh1, IntAOCh1VsRef5V, IntAOCh1VsAOCh0, IntDevTemp
AT-AI-16XE-10, PCI-6032E, PCI-6033E, DAQCard-AI-16XE-50, NEC-AI-16XE-50	IntAIGnd, IntRef5V, IntA0GndVsAIGnd
AT-MIO-16E-1, AT-MIO-16E-2, AT-MIO-16E-3, AT-MIO-16DE-10, AT-MIO-16E-10, DAQPad-6020E, NEC-MIO-16E-4	IntAIGnd, IntRef5V, IntCmRef5V, IntAOGndVsAIGnd, IntAOCh0, IntAOCh0VsRef5V, IntAOCh1, IntAOCh1VsRef5V

Tabelle B-5. Analogausgangs-Eigenschaften—MIO- und AI-Geräte

Gerät	Anzahl der Kanäle	DAC-Typ	FIFO-Größe	Ausgabegrenzen (V)	Update-Takt	Übertragungsmethode
AT-MIO-16E-1 AT-MIO-16E-2 AT-MIO-64E-3 NEC-MIO-15E-4 VXI-MIO-64E-1	0, 1	12-Bit Doppelpuffer	2048	0 bis 10, ±10, ±Vref, 0 bis Vref	Update-Takt 1 oder externer Update.	DMA, Interrupts
AT-MIO-16E-10	—	12-Bit Doppelpuffer	0	±10, 0 bis 10	Update-Takt 1 oder externer Update.	DMA, Interrupts
AT-MIO-XE-50 NEC-MIO-16XE-50	—	12-Bit Doppelpuffer	0	±10	Update-Takt 1 oder externer Update.	DMA, Interrupts

Tabelle B-5. Analogausgangs-Eigenschaften—MIO- und AI-Geräte (Fortsetzung)

Gerät	Anzahl der Kanäle	DAC-Typ	FIFO-Größe	Ausgabegrenzen (V)	Update-Takt	Übertragungsmethode
DAQPad-MIO-16XE-50	—	12-Bit Doppelpuffer	0	±10	Update-Takt 1 oder externer Update.	Interrupts
AT-MIO-16XE-10 VXI-MIO-64XE-10	—	16-Bit Doppelpuffer	0	±10, 0 bis 10	Update-Takt 1 oder externer Update.	DMA, Interrupts
OCI-MIO-16E-1 CPCI-6070E PXI-6070E PCI-6071E	—	12-bit Doppelpuffer	2048	±10, 0 bis10	Update-Takt 1 oder externer Update.	DMA, Interrupts
PCI-MIO-16E-4 CPCI-6040E PXI-6040E	—	12-Bit Doppelpuffer	512	±10, 0 bis10	Update-Takt 1 oder externer Update.	DMA, Interrupts
PCI-MIO-16XE-50 CPCI-6011E PXI-6011E	—	12-Bit Doppelpuffer	0	±10	Update-Takt 1 oder externer Update.	DMA, Interrupts
PCI-MIO-16XE-10 CPCI-6030E PXI-6030E PCI-6031E	—	16-Bit Doppelpuffer	5048	±10, 0 bis10	Update-Takt 1 oder externer Update.	DMA, Interrupts
DAQPad-6020E	—	12-Bit Doppelpuffer	0	±10, 0 bis10	Update-Takt 1 oder externer Update.	DMA, Interrupts

Tabelle B-5. Analogausgangs-Eigenschaften—MIO- und AI-Geräte (Fortsetzung)

Gerät	Anzahl der Kanäle	DAC-Typ	FIFO-Größe	Ausgabegrenzen (V)	Update-Takt	Übertragungsmethode
PCI-6110E PCI-6111E	—	16-Bit Doppelpuffer	4096	±5	Update-Takt 1 oder externer Update.	DMA, Interrupts
AT-MIO-16F-5 AT-MIO-64F-5	0, 1	12-Bit Doppelpuffer (64F-5: 2K FIFO)	—	0 to 10, ±10, ±Vref, 0 bis Vref	Update-Takt 1 ist zuerst verfügbar von ctr 5, 2, 1 oder externem Update. Standard ist 5. Zeitbasis-Signalbereich ist 5 000 000, 1 000 000, 100 000, 10 000, 1 000, und 100.	DMA, Interrupts
AI-MIO-16X	0, 1	16-Bit Doppelpuffer (2K FIFO)	—	±10, 0 to 10, ±Vref, 0 bis Vref	Update-Takt 1 ist zuerst verfügbar auf ctr 5, 2, 1 oder externem Update. Zeitbasis-Signalbereich ist 5 000 000, 1 000 000, 100 000, 10 000, 1 000, 100.	DMA, Interrupts

Tabelle B-5. Analogausgangs-Eigenschaften—MIO- und AI-Geräte (Fortsetzung)

Gerät	Anzahl der Kanäle	DAC-Typ	FIFO-Größe	Ausgabegrenzen (V)	Update-Takt	Übertragungsmethode
AT-MIO-16/16D	0, 1	12-Bit Doppelpuffer	—	0 bis 10, ±10, ±Vref, 0 bis Vref	Update-Takt 1 ist ctr2 oder externer Update. Zeitbasis-Signalbereich ist 1 000 000, 100 000, 10 000, 1 000 und 100.	Interrupts
NB-MIO-16/16X	0, 1	MIO-16:12-Bit; MIO-16X: 12-Bit Doppelpuffer	—	0 bis 10, ±10, ±Vref, 0 bis Vref	Update-Takt 1, externer Update (nur MIO-16X). Zeitbasis-Signalbereich ist 1 000 000, 100 000, 10 000, 1 000 und 100.	MIO-16:DMA; MIO-16X: DMA, Interrupts

Tabelle B-6. Analogausgangs-Eigenschaften—Geräte E-Serie

Gerät	Reglitching möglich	Bezugsmasse möglich	Kann Bedienelement FIFO-Modian fordern	AO Gating, anhalten/ wieder- aufnehmen unterstützt	Mehrfachpuffer unterstützt
AT-MIO-16E-1 AT-MIO-16E-2 AT-MIO-64E-3 NEC-MIO-16E VXI-MIO-64E-1	Ja	Nein	Nein	Nein	Ja
AT-MIO-16E-10 AT-MIO-16DE-10	Nein	Nein	Nein	Ja	Ja
AT-MIO-XE-50 NEC-MIO-16XE-50	Nein	Nein	Nein	Nein	Ja
DAQPad-MIO-16XE-50	Nein	Nein	Nein	Nein	Ja
AT-MIO-16XE-10	Nein	Nein	Nein	Ja	Ja
VXI-MIO-64XE-10	Nein	Nein	Nein	Nein	Ja
PCI-MIO-16E-1 CPCI-6070E PXI-6070E PCI-6071E	Ja	Ja	Ja	Ja	Nein
PCI-MIO-16E-4 CPCI-6040E PXI-6011E	Nein	Ja	Ja	Ja	Nein
PCI-MIO-16XE-10 CPCI-6030E PXI-6030E PCI-6031E	Nein	Nein	Ja	Ja	Nein
DAQPad-6020E	Nein	Ja	Nein	Ja	Nein
PCI-6110E PCI-6111E	Nein	Nein	Ja	Ja	Nein

Tabelle B-7. Digitale I/O-Hardware-Fähigkeiten—MIO- und AI-Geräte

Gerät	Anschlußtyp	Anzahl der Anschlüsse	Handshake-Modus	Richtung	DIO-Takte	Übertragungs-methode
All MIO-16-Geräte AT-MIO-16D[1] AT-MIO-64F-5	4-Bit-Anschlüsse	0, 1	Kein Handshake-Betrieb	Lesen oder schreiben	Keine	Software-Abfrage
All MIO-16E-Geräte All NEC-E-Serie-Geräte AT-MIO-64E-12 AT-MIO-16DE-10[1] AT-MIO-16XE-50 DAQPad-MIO-16XE-50 PCI-MIO-16XE-50 PXI-6040E (MIO-16E-4) PXI-6070E (MIO-16E-1) PXI-6071E (MIO-64E-1) PCI-6031E (MIO-64XE-10) PCI-6032E (AI-16XE-10) PCI-6033E (AI-64XE-10) PCI-6110EPCI-6111E	8-Bit-Anschlüsse	0	Kein Handshake-Betrieb	Bitwiese Richtungs-steuerung	Keine	Software-Abfrage
AT-MIO-16D[1] AT-MIO-16DE-10[1]	8-Bit-Anschlüsse	2, 3	Handshake-Betrieb an oder aus	Lesen oder schreiben, Anschluß 2 kann bidirektional sein	Keine	Interrupts
	8-Bit-Anschlüsse	4	Kein Handshake-Betrieb; kann nicht benutzt werden, wenn Anschluß 2 oder 3 Handshake-Betrieb benutzt	Lesen oder schreiben	Keine	Software-Abfrage

[1] Diese Geräte sind in dieser Tabelle mehrmals aufgeführt, da sie über eine verbesserte digitale Funktionalität verfügen.

Tabelle B-8. Counter-Eigenschaften—MIO- und AI-Geräte

Gerät	Verwendeter Counter-Chip	Anzahl der verfügbaren Allzweck-Counter	Verfügbare Zeitbasen	Anzahl der Bits	Verfügbare Gate-Modi	Verfürbare Ausgänge	Verfügbare Ausgabemodi	Taktrichtung1
Geräte der E-Serie	DAQ-STC	2	2 intern: 20 MHz oder 100 kHz; extern	24	ansteigende Flanke, absteigende Flanke, High-Level, Low-Level	2		Auf- oder abwärts, kann SW- oder HW-gesteuert sein
AT-MIO-16F-5 AT-MIO-64F-5 AT-MIO-16/16D NB-MIO-16/16X	Am-9513	3	5 oder 6 intern: 5 MHz (nur auf CTR2 von16F-5, 64F-5 und AT-MIO-16X), 1 MHz, 100 kHz, 10 kHz, 1 kHz, 100 Hz; extern	16	ansteigende Flanke, absteigende Flanke, High-Level, Low-Level	2	TC takten oder TC um-schal-ten	Auf

[1]SW = Software; HW = Hardware.

Tabelle B-9. Counter-Benutzung für Analogeingang und -ausgang—MIO- und AI-Geräte

Gerätename	Counter-Chip verwendet	AI Kanaltakt	AI Abtastwerte-Counter	AI Scan -Takt	AO Update-Takt
Geräte der E-SErie	DAQ-STC	Der DAQ-STC-Chip verwendet für diese Zwecke dedizierte Takte.			
AT-MIO-16F-5 AT-MIO-64F-5 AT-MIO-16X	Am9513	Ctr 3	Ctr 4 (& 5)[1]	Ctr 2 (oder 1)[2]	Ctr 5, 2 oder 1
AT-MIO-16/16D NB-MIO-16X	Am9513	Ctr 3	Ctr 4 (& 5)[1]	Ctr 2 (oder 1)[2]	Ctr 2 (und via DMA für NB-MIO-16X)
NB-MIO-16	Am9513	Ctr 3	Ctr 4 (& 5)[1]	None (oder 1)[2]	(via DMA)

[1] Wenn die Gesamtzahl der Abtastwerte weniger als 65535 ist, wird nur der erste Counter benutzt. Wenn die Anzahl der Abtastwerte 65536 übersteigt, wird der erste Counter zusammen mit dem zweiten Counter als 32-Bit-Abtastwert-Counter benutzt.

[2] Ctr 2 (oder kein Counter für NB-MIO-16) wird für normale Scanning-Operationen benutzt, und Counter 1 wird für AMUX-64T und SCXI-Hardware-Scanning benutzt.

Hardware-Leistungsfähigkeit der Lab- und Serie-1200-Geräte und tragbarer Geräte

Tabelle B-10. Konfigurationsprogrammierbarkeit der Analogeingabe—Lab- und Serie-1200-Geräte und tragbare Geräte

Gerät	Verstärkung	Bereich	Polarität	SE/DIFF	Kopplung
Lab-LC Lab-NB	Nach Gruppe	Nach Gerät	Nach Gerät	SE	DC
Lab-PC+	Nach Gruppe	Nach Gruppe	Nach Gerät	Nach Gerät	DC
SCXI-1200 DAQPad-1200 DAQCard-1200 PCI-1200	Nach Gruppe	Nach Gruppe	Nach Gruppe	Nach Gruppe	DC
DAQCard-500	1	Nur 1 Bereich verfügbar	Bipolar	SE	DC
DAQCard-PC-516	1	Nur 1 Bereich verfügbar	Bipolar	Nach Gruppe	DC
DAQCard-700	1	Nach Gruppe	Bipolar	Nach Gruppe	DC
PC-LPM-16	1	Nach Gerät	Bipolar	SE	DC

☞ **Hinweis** ***Nach Gerät bedeutet, daß Sie den Wert eines Parameters mit Hardware-Jumpern auswählen, wobei sich die Auswahl auf jede Kanalgruppe auf dem Gerät auswirkt. Nach Gruppe bedeutet, daß Sie die Auswahl mittels Software programmieren, wobei sich die Auswahl auf alle gleichzeitig benutzten Kanäle auswirkt. Nach Kanal bedeutet, daß Sie die Auswahl mit Hardware-Jumpern oder mittels Software auf für je einen Kanal programmieren. Wenn für einen Parameter ein spezifischer Wert angezeigt wird, ist dieser Wert fixiert.***

Tabelle B-11. Analog-Eingangseigenschaften—Lab- und Serie-1200-Geräte und tragbare Geräte (Teil 1)

Gerät	Anzahl der Kanäle	Auflösung (Bits)	Verstärkung[1]	Bereich(V)[1]	Eingabe FIFO (Abtastwerte)
Lab-LC Lab-NB	8SE	12	1; 2; 5; 10; 20; 50; 100	±5, 0 bis10	16
Lab-PC+ SCXI-1200 DAQPad-1200 DAQCard-1200 PCI-1200	8SE, 4DI	12	1; 2; 5; 10; 20; 50; 100	±5, 0 bis10	2048; Lab-PC: 512
DAQCard-500	8SE	12	1	±5	16
DAQCard PC516	8SE,4DI	16	1	±5	512
DAQCard-700	16SE, 8DI	12	1	±10; ±5, ±2,5	512
PC-LPM-16	16SE	12	1	±5, ±2,5;0 bis10, 0 bis 5	16

1 Sie können die Grenzeinstellungen für Ihr Gerät bestimmen, indem Sie die Bereichs- und die Spannungswerte miteinander multiplizieren. Weitere Informationen zu Grenzeinstellungen in LabVIEW finden Sie in Kapitel 3, *Grundlagende Konzepte zur Datenerfassung*, im *Grundlagen der Datenerfassung mit LabVIEW.*

Tabelle B-12. Analogeingabe-Eigenschaften—Lab- und Serie-1200-Geräte und tragbare Geräte (Teil 2)

Gerät	Scanning	Trigger	Max Abtastwert-rate(S/s)	Übertragungs-methode
Lab-LC Lab-NB	Jeder einzelne Kanal; für mehrere Kanäle, *N* bis 0, wobei $N \leq 7$ ist.	Software-Trigger, Pretrigger, und Posttrigger mit digitalem Trigger	62.5 k	Interrupts
Lab-PC+ SCXI-1200 DAQPad-1200 DAQCard-1200 PCI-1200	Jeder einzelne Kanal; für mehrere Kanäle, *N* bis 0, wobei $N \leq 7$ ist.	Software-Trigger, Pretrigger, und Posttrigger mit digitalem Trigger	100 k; Lab-PC+: 83 k	Interrupts; Lab-PC+: Interrupts, DMA
DAQCard-500 DAQCard 516 PC-516	Jeder einzelne Kanal; für mehrere Kanäle, *N* bis 0, wobei $N \leq 7$ ist.	Nur Software-Trigger	50 k	Interrupts
DAQCard-700	Jeder einzelne Kanal; für mehrere Kanäle, *N* bis 0, wobei *N Nr.15* ist.	Nur Software-Trigger	100 k	Interrupts
PC-LPM-16	Jeder einzelne Kanal; für mehrere Kanäle, *N* bis 0, wobei *N Nr.15* ist.	Nur Software-Trigger	50 k	Interrupts

Tabelle B-13. Analogausgabe-Eigenschaften—Lab- und Serie-1200-Geräte und tragbare Geräte

Gerät	Kanal-nummern	DAC-Typ	Ausgabebe-grenzungen (V)	Update-Takte	Signalverlauf-Gruppierung	Übertra-gungs-methoden
Lab-NB Lab-LC	0, 1	12-Bit Doppelpuf-fer	0 bis 10, ±5	Update-Takt 1 ist ctrA2 oder externes Update; Zeitbasis ist 1 MHz oder ctrB0	0, 1 oder 0 und 1	Interrupts
Lab-PC+ SCXI-1200 DAQPad-1200 DAQCard-1200 PCI-1200	0, 1	12-Bit Doppelpuf-fer	0 bis 10, ±5	Update-Takt 1 ist ctrA2 oder externes Update; Zeitbasis-Signalbereich ist 1 000 000, 100 000, 10 000, 1 000 und 100	0, 1 oder 0 und 1	Interrupts

☞ **Hinweis** ***Die DAQCard-516- und PC 516-Geräte haben keine Analogausgabe.***

Tabelle B-14. Counter-Benutzung für Analogeingabe und -ausgabe—Lab- und Serie-1200-Geräte und tragbare Geräte

Gerätename	Verwendeter Counter-Chip	AI Kanaltakt	AI Abtastwert-Counter	AI Scan-Takt	AO Update-Takt
Lab-NB, Lab-LC	82C53	Ctr A0 (& B0)[1]	Ctr A1	Keine	Ctr A2
Lab-PC+, DAQPad-1200, SCXI-1200, DAQCard-1200, PCI-1200	82C53	Ctr A0 (& B0)[1]	Ctr A1	Ctr B1	Ctr A2
DAQCard-500, DAQCard-700,	8254	Ctr 0	(Software)	Keine	Keine
DAQCard 516 PC-516	82C54	Ctr 0	SW	Keine	Keine
PC-LPM-16	82C53	Ctr 0	(Software)	Keine	Keine

[1] Der zweite Counter wird als erweiterte Zeitbasis für getaktete Analogeingaben oder -ausgaben benutzt, wenn das Abtastwert-Intervall 65.535 ms übersteigt.

Tabelle B-15. Digitale I/O-Hardware-Leistungsfähigkeiten—Lab- und Serie-1200-Geräte und tragbare Geräte

Gerät	Anschluß-typ	Anzahl der Anschlüsse	Handshake-Modi	Richtung	DIO-Takte	Übertragungs methode
Lab-NB Lab-LC Lab-PC+ SCXI-1200 DAQCard-1200 DAQPad-1200 PCI-1200	8-Bit-Anschluß	0, 1	Handshake-Betrieb an oder aus	Lesen oder schreiben, Anschluß 0 kann u.U. bidirektional sein	Keine	Interrupts
	8-Bit-Anschluß	2	Kein Handshake -Betrieb; nicht benutzbar, wenn Anschluß 0 oder 1 Handshake-Betrieb benutzt	Lesen oder schreiben	Keine	Software-Ab-frage
PC-LPM-16	8-Bit-Anschlüsse	0, 1	Kein Handshake-Betrieb	0: lesen oder schreiben	Keine	Software-Ab-frage
DAQCard-500 DAQCard-516	4-Bit-Anschlüsse	0, 1	Kein Handshake-Betrieb	0: schreiben, 1: lesen	Keine	Software-Abfrage
DAQCard-700	8-Bit-Anschlüsse	0, 1	Kein Handshake-Betrieb	0: schreiben, 1: lesen	Keine	Software-Abfrage

54xx-Geräte

Tabelle B-16. Analogausgabe- und Digitalausgabe-Eigenschaften—Serie-54XX-Geräte

Eigenschaften	AT-5411, PCI-5411
Kanalnummern	0
Maximale Update-Rate	40 MHz.
Update-Intervall	1 bis 65535.
DAC-Typ	12-Bit, Doppelpuffer
Ausgabegrenzen (V) (Nur interne Referenz)	±5 in 50 Ω Ladung ±10 ohne Endwiderstand (hohe Eingangsimpedanz) Belastung.
Update-Takte	Update-Takt 1.
Trigger	Bei ansteigender TTL-Flanke, bei Trigger-Eingabeanschluß oder RTSI-Pin. Kann auch intern durch Software erzeugt werden.

Tabelle B-16. Analogausgabe- und Digitalausgabe-Eigenschaften—Serie-54XX-Geräte (Fortsetzung)

Eigenschaften	AT-5411, PCI-5411
RTSI Trigger-Bus	Ja
Digitale Ausgabe	16-Bits mit Taktsignal
Signalverlaufgruppierung	0
Signalverlauf-Speichertiefe -ARB-Modus -Direkte digitale Synthese (DDS) Modus	 2 000 000 16-Bit Abtastwerte (standard) 16 384 16-Bit Abtastwerte maximal
Maximale Signalverlaufstufen	290
Puffernummern	1 bis 1.000.
Pufferiterationen	1 bis 65.535
Puffer-Abtastwertzählung -ARB-Modus - DDS-Modus	256 Abtastwerte minimal Speichertiefe maximal Hinweis: Puffergröße sollte ein Vielfaches von 8 Abtastwerten sein. Muß gleich 16.384 Abtastwerten sein. Wenn Sie eine geringere Anzahl von Abtastwerten laden, sehen Sie den Inhalt der ungefüllten Speicherabschnitte auch in der Signalverlaufserzeugung.
Markerausgabe	TTL-Level, einer für jede Stufe verfügbar
DDS-Akkumulatorgröße	32-Bit
Maximale Ausgabefrequenz	16 MHz
Ausgabefrequenz-Auflösung (nur DDS-Modus)	9.31 mHz
Ausgabedämpfung (nach der DAC)	0 bis 74,000 dB (Dezibel) in 0,001 dB Schritten
SYNC-Tastverhältnis (% Hoch)	TTL-Level, 20% bis 80%.
PLL-Referenz-Takt	1 MHz, 10 MHz oder 20 MHz
Ausgabe aktiviert	Software umschaltbar auf EIN oder AUS
Ausgangsimpedanz	50Ω oder 75Ω (Video), Software auswählbar
Tiefpaß-Filter	16 MHz, Software umschaltbar auf EIN oder AUS
Digitaler Halbband-Interpolations-Filter	80 MSPS, Software umschaltbar auf EIN oder AUS
Trigger-Operationsmodus	Einfach, fortlaufend, gestuft und getrennt

☞ **Hinweis** ***Standardeinstellungen für Ihr Gerät sind in Ihrem Hardware-Benutzerreferenz-Handbuch vermerkt.***

Tabelle B-17. Counter-/Timer-Eigenschaften—Lab- und Serie-1200-Geräte und tragbare Geräte

Gerät	Verwendeter Counter-Chip	Verfügbare Anzahl von Allzweck-Countern	Verfügbare Zeitbasen	Anzahl von Bits	Verfügbarer Gate-Modus	Verfügbare Ausgabe	Verfügbare Ausgabemodi	Zählrichtung
Lab-NB Lab-LC Lab-PC+ SCXI-1200 DAQCard-1200 DAQPad-1200 PC-LPM-16 PCI-1200	8253	3 (2 mit SOURCE-Eingabe bei I/O-Anschluß)	Intern: 1 MHz; (PC-LPM-16:nur auf CTRB0) extern	16	High Level oder ansteigende Flanke, abhängig vom Eingabemodus	3	Sehen Sie in der ICTRControl-VI-Beschreibung zu Modi in Kapital 28 *Fortgeschrittene Counter-VIs* nach.	Abwärts
DAQCard-500 DAQCard 516 DAQCard-700 PC-516	8254	3 (2 mit SOURCE-Eingabe bei I/O-Anschluß)	Intern: 1 MHz nur auf CTRB0; extern	16	High Level oder ansteigende Flanke, abhängig vom Ausgabemodus	3(2 für DAQCard-500)	Sehen Sie in der ICTRControl-VI-Beschreibung zu Modi in Kapital 28 *Fortgeschrittene Counter-VIs* nach.	Abwärts

Hardware-Leistungsfähigkeit des SCXI-Moduls

Tabelle B-18. Analogeingangs-Eigenschaften—SCXI-Module (Teil 1)

Modul	Anzahl der Kanäle	Eingangs-Spannungs-bereich(V)	Verstärkung[1]	Filter[1]	Erregungs-kanäle[1]	Modusunter-stützung
SCXI-1100	32 DI	±10	1, 2, 5, 10, 20, 50, 100, 200, 500, 1,000, 2,000 (SW/M)[1]	Tiefpaß-Filter (oder kein Filter) mit 10 kHz oder 4 Hz Grenz-frequenz (JS/M)[1]	—	Multiplexbetrieb
SCXI-1102	32 DI	±10	1, 100 (SW/C)[1]	1 Hz Tiefpaß auf jedem Kanal	—	Multiplexbetrieb
SCXI-1120 SCXI-1121	8 DI (SCXI-1120) 4 DI (SCXI-1121)	±5	1, 2, 5, 10, 20, 50, 100, 200, 500, 1,000, 2,000 (JS/C)[1]	Tiefpaß-Filter mit 10 kHz oder 4 Hz Grenzfrequenz (JS/C)[1]	Nur SCXI-1121: 4 Spannungs- oder Stromer-regungen JS/C[1] (Kanäle)	Multiplex- oder Parallelbetrieb
SCXI-1120D	8 DI (SCXI-1120) 4 DI (SCXI-1121)	±5	0.5, 1, 2.5, 5, 10, 25, 50, 100, 250, 500, 1,000	4.500, 24.500 Hz	Nur SCXI-1121: 4 Spannungs- oder Stromer-regungen JS/C[1] (Kanäle)	Multiplex- oder Parallelbetrieb
SCXI-1122	16 DI oder 8 DI und 8 Erregungs-SW/M1-Kanäle	±10	0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1,000, 2,000 (SW/M)[1]	Tiefpaß-Filter mit 4kHz oder 4 Hz Grenz-frequenz	8 Spannungs- oder Stromer-regungskanäl e in 4-Verbin-dungs-Scan-ning-Modus	Multiplexbetrieb

Tabelle B-18. Analogeingangs-Eigenschaften—SCXI-Module (Teil 1) (Fortsetzung)

Modul	Anzahl der Kanäle	Eingangs-Spannungs-bereich(V)	Verstärkung[1]	Filter[1]	Erregungs-kanäle[1]	Modusunter-stützung
SCXI-1140	8 DI, abtasten und halten	±10	1, 10, 100, 200, 500 (DS/C)[1]	Keine	—	Multiplex- oder Parallelbetrieb
SCXI-1141	8 DI	±5	1, 2, 5, 10, 20, 50, 100 (SW/C)[1]	Elliptischer Tiefpaß-Filter mit 10 Hz bis 25 KHz Grenz-frequenz[2] (SW/M)[1] (deaktiviert auf Kanalbasis)	—	Multiplex- oder Parallelbetrieb

[1]DS/C = DIP-Schalter auswählbar pro Kanal, JS/C = Jumper auswählbar pro Kanal, JS/M = Jumper auswählbar pro Modul, SW/C = Software auswählbar pro Kanal, SW/M = Software auswählbar pro Modul

[2] SCXI-1141 hat eine automatische Filtereinstellung. LabVIEW stellt die Filterfrequenz aufgrund der mit dem Modul benutzten Scan-Raten ein.

Tabelle B-19. Analogausgangs-Eigenschaften—SCXI-Modules

Modul	Anzahl der Kanäle	Ausgangsspannungsbereich (V oder mA)	Modusunter-stützung
SCXI-1124	6 Spannung oder Strom	0 bis 1, 0 bis 5, 0 bis 10, ±1, ±5, ±10 (Software auswählbar) oder 0 bis 20 mA	Multiplex-betrieb

Tabelle B-20. Relais-Eigenschaften—SCXI-Module

Modul	Anzahl der Kanäle[1]	Geschaltet oder nichtgeschaltet	Start-Relais-Position[1]	Modusunterstüt-zung
SCXI-1160	16	Geschaltet	Relais in der Abschalt-Position lassen.	Multiplexbetrieb
SCXI-1161	8	Nichtgeschal-tet	Zur Position Normally Closed (NC, normalerweise geschlossen) umschalten, wenn die Hardware-Rückstellung auf dem Modul eingestellt ist.	Multiplex-betrieb

[1] Sie können jedes SCXI-Relais einzeln zurücksetzen, ohne die anderen Relais zu beeinflussen, oder Sie können alle Relais auf einmal ändern.

Tabelle B-21. Digital Input and Output Characteristics—SCXI Modules

Modul	Modultyp	Anzahl der Kanäle1	Eingangsspannungs-bereich	Modusunterstützung
SCXI-1162	Eingang	32 (optisch isoliert)	0 bis 5 V	Parallelunterstützung, wenn mit einem DIO-24, DIO-96- oder DIO-32F-Gerät verbunden. Unterstützung des Multiplexbetriebs mit jedem DAQ-Gerät, das SCXI unterstützt.
SCXI-1162HV	Eingang	32 (optisch isoliert)	Gleich- oder Wechselstromsignale bis zu ±240 V	Parallelunterstützung, wenn mit einem DIO-24, DIO-96- oder DIO-32F-Gerät verbunden. Unterstützung des Multiplexbetriebs mit jedem DAQ-Gerät, das SCXI unterstützt.
SCXI-1163	Ausgang	32 (optisch isoliert)	0 bis 5 V	Parallelunterstützung, wenn mit einem DIO-24, DIO-96- oder DIO-32F-Gerät verbunden. Unterstützung des Multiplexbetriebs mit jedem DAQ-Gerät, das SCXI unterstützt.
SCXI-1163R*	Ausgang	32 (optisch isoliert)	±240 V	Parallelunterstützung, wenn mit einem DIO-24, DIO-96- oder DIO-32F-Gerät verbunden. Unterstützung des Multiplexbetriebs mit jedem DAQ-Gerät, das SCXI unterstützt.

[1]Funktional dem SCXI-1163 gleich, benutzt aber anstelle von Digitalausgaben statische Relais.

Tabelle B-22. Terminalblock-Auswahlanleitung—SCXI-Module

SCXI-Modul	Terminalblöcke	Cold-Junction Compensation Sensor (CJC; Kaltverbindungs-Kompensationssensor)
SCXI-1100 SCXI-1102	SCXI-1303 SCXI-1300	Thermistor IC-Sensor
SCXI-1120 SCXI-1121	SCXI-1320 SCXI-1321[1] SCXI-1327 SCXI-1328	IC-Sensor IC-Sensor Thermistor Thermistor
SCXI-1122	SCXI-1322	Thermistor
SCXI-1124	SCXI-1325	—
SCXI-1140	SCXI-1301 SCXI-1304	— —
SCXI-1141	SCXI-1304	—
SCXI-1160	SCXI-1324	—
SCXI-1161	Terminale ohne Verschraubung im Modul.	—
SCXI-1162 SCXI-1162HV SCXI-1163 SCXI-1163R	SCXI-1326	—
SCXI-1180	SCXI-1302	—
SCXI-1181	SCXI-1300 SCXI-1301	IC-Sensor —
SCXI-1200	SCXI-1302 CB-50	— —

[1] Nur SCXI-1121

Tabelle B-23. Konfigurationsprogrammierbarkeit der Analogeingabe

Gerät	Verstärkung	Kopplung
5102-Geräte	Nach Kanal	Nach Kanal

Tabelle B-24. Konfigurationsprogrammierbarkeit der Analogeingabe

Gerät	Anzahl der Kanäle	Auflösung	Verstärkung	Bereich(V)	Eingangs-FIFO (Worte)	Scanning
5102-Geräte	2	8 Bits	1, 5, 20, 100	± 5	663546	1 oder 2 zwei Kanäle in beliebiger Reihenfolge ohne Wiederholungen.

☞ **Hinweis** *Nach Gerät* ***bedeutet, daß Sie den Wert eines Parameters nach den Hardware-Jumpern auswählen, wobei sich die Auswahl auf jede Kanalgruppe auf dem Gerät auswirkt****. Nach Gruppe* ***bedeutet, daß Sie die Auswahl mittels Software programmieren, wobei sich die Auswahl auf alle gleichzeitig benutzten Kanäle auswirkt.*** *Nach Kanal* ***bedeutet, daß Sie die Auswahl mit Hardware-Jumpern oder mittels Software auf für je einen Kanal programmieren. Wenn für einen Parameter ein spezifischer Wert angezeigt wird, ist dieser Wert fixiert.***

Hardware-Leistungsfähigkeiten von Nur-Analoggeräten

Tabelle B-25. Analogausgabe-Eigenschaften—Nur-Analogeräte

Gerät	Anzahl der Kanäle	DAC-Typ	Ausgangsbegrenzungen	Update-akt	Signalverlauf-Gruppierung	Übertragungsmethode
AT-AO-6 AT-AO-10 NB-AO-6	0 bis 5, 6 bis 9*	12-Bit Doppelpuffer mit1 K FIFO für Update-Takt 1 Kanäle	±10V, ±Vref1, 0 bis 10 V, 0 bis Vref1, 4 bis 20 mA	Update-Takt 1 ist ctr0 oder externes Update. Update-Takt 1 Kanäle sind 0, 1, 2, 3, 4, 5, 6*, 7*, 8*, 9*, 0 bis 1, 0 bis 3, 0 bis 5, 0 bis7*, 0 to 9*. Update-Takt 2 ist ctr1. Update-Takt 2 Kanäle sind 2, 3, 4, 5, 6*, 7*, 8*, 9*, 2 bis 3, 2 to 5, 2 bis 7*, 2 to 9*; Zeitbasis-Signalbereich ist 1 000 000, 100 000, 10 000, 1 000, 100	Für Update-Takt 1 Kanäle sind jeder beliebige Kanal*N* oder Kanalpaare: 0–*N;* für Update-Takt 2 Kanäle sind 2–*N*, gleiche Regeln wie oben: *N*r.6, *N*r.10*	Update-Takt 1 Kanäle: DMA, Interrupts; Update-Takt 2 Kanäle: Interrupts
PC-AO-2DC (Plug and Play)	0, 1	—	0 bis 10V, ±5V, 0–20mA Senden durch Software auswählbar	—	—	—

Tabelle B-25. Analogausgabe-Eigenschaften—Nur-Analogeräte (Fortsetzung)

Gerät	Anzahl der Kanäle	DAC-Typ	Ausgangsbegrenzungen	Update-akt	Signalverlauf-Gruppierung	Übertragungsmethode
DAQCard-AO-2DC	0, 1	—	0 to 10V, ±5V, 0–20mA Senke durch Software auswählbar	—	—	—

*Nur AT-AO-10

Hardware-Leistungsfähigkeit der dynamischen Signalerfassungsgeräte

Tabelle B-26. Konfigurationsprogrammierbarkeit der Analogeingabe—Dynamische Signalerfassungsgeräte

Gerät	Verstärkung	Bereich (V)	Polarität	SE/DIFF	Kopplung
EISA-A2000 NB-A2000	1	±5	Bipolar	SE	Nach Kanal
NB-A2100 AT-DSP2200	1	±2.828	Bipolar	SE	Nach Gruppe
NB-A2150 AT-A2150	1	±2.828	Bipolar	SE	Nach Kanalpaar 0 und 1, 2 und 3

☞ **Hinweis** *Nach Gerät* ***bedeutet, daß Sie den Wert eines Parameters nach den Hardware-Jumpern auswählen, wobei sich die Auswahl auf jede Kanalgruppe auf dem Gerät auswirkt****. Nach Gruppe* ***bedeutet, daß Sie die Auswahl mittels Software programmieren, wobei sich die Auswahl auf alle gleichzeitig benutzten Kanäle auswirkt.*** *Nach Kanal* ***bedeutet, daß Sie die Auswahl mit Hardware-Jumpern oder mittels Software auf für je einen Kanal programmieren. Wenn für einen Parameter ein spezifischer Wert angezeigt wird, ist dieser Wert fixiert.***

Tabelle B-27. Analogausgangs-Eigenschaften—Dynamische Signalerfassungsgeräte

Gerät	Anzahl der Kanäle	DAC-Typ	Ausgabebegrenzungen	Update -Takte	Übertragungsmethode
PCI-4451 PCI-4552	0, 1	18-Bit	±10V, ±1V, ±100	Update-Takt1	DMA

Tabelle B-28. Analogeingangs-Eigenschaften—Dynamische Signalerfassungsgeräte

Gerät	Anzahl der Kanäle	Auflösung	Bereich (V)	Eingabe FIFO (Worte)	Trigger	Scanning	Max. Abtastwert-Rate (S/s)	Übertragungs-methode
EISA-A2000 NB-A2000	4 SE	12 Bits	±5	EISA:51; NB:1,024	Software-Trigger, Retrigger und Posttrigger mit digitaler oder analoger Trigger- und Posttrigger-Verzögerung	0, 1, 2, 3, 0 und 1, 2 und 3, 0 bis 3.	1M	DMA, Inter-rupts
NB-A2100 NB-A2150	2 SE	16B Bits	±2,828	32	Software-Trigger, Retrigger und Posttrigger mit digitaler oder analoger Triggerung	A2150: 0, 1, 2, 3, 0 und 1, 2 und 3, 0 bis 3; A2100: 0, 1, 0 und 1	2100:48 k, 2150:24 k, 2150C: 48 k, 2150S: 51,2 k	DMA, Inter-rupts
AT-A2150	4 SE	16 Bits	±2.828	—	Software-Trigger, Retrigger und Posttrigger mit digitaler oder analoger Triggerung	0, 1, 2, 3, 0 and 1, 2 and 3, 0 and 3	2150:24 k 2150:51,2 k	DMA, Inter-rupts

Hardware-Leistungsfähigkeit für Nur-Digitalgeräte

Tabelle B-29. Digitale Hardware-Leistungsfähigkeiten—Digital-I/O-Geräte

Gerät	Anschluß-nummern	Anzahl der An-schlüsse	Handshake-Modus	Richtung	DIO-Takte	Übertragungs-methode
AT-DIO-32F NB-DIO-32F	8-Bit-Anschlüsse	0, 1, 2, 3	8-Bit-Anschluß Handshake-Betrieb an oder aus; extensiver Handshake-Betrieb	Lesen oder schreiben	Zwei Takte verfügbar 16-Bit mit variabler Zeitbasis	DMA für jede Gruppe; Zweifach-Kanal DMA für Gruppen mit Anschluß 0
	2-Bit-Anschlüsse	4	Kein Handshake-Betrieb	Lesen oder schreiben	Keine	Software-Abfrage

Tabelle B-29. Digitale Hardware-Leistungsfähigkeiten—Digital-I/O-Geräte (Fortsetzung)

Gerät	Anschluß-nummern	Anzahl der An-schlüsse	Handshake-Modus	Richtung	DIO-Takte	Übertragungs-methode
PC-DIO-24 NB-DIO-24 DAQCard-DIO-24	8-Bit-Anschluß	0, 1	Handshake--Betrieb an oder aus	Lesen oder schreiben, Anschluß 0 kann bidirektional sein	Keine	Interrupts
	8-Bit-Anschluß	2	Kein Handshake-Betrieb; kann nicht benutzt werden, wenn Anschluß 0 oder 1 Handshake-Betrieb benutzt	Lesen oder schreiben	Keine	Software-Abfrage
PC-DIO-96 PCI-DIO-96 NB-DIO-96 DAQPad-6507	8-Bit-Anschluß	0, 1, 3, 4, 6, 7, 9, 10	Handshake-Betrieb an oder aus	Lesen oder schreiben, Anschlüsse 0, 3, 6 und 9 kann bidirektional erfolgen	Keine	Interrupts
	8-Bit-Anschluß	2, 5, 8, 11	Kein Handshake-Betrieb; kann nicht benutzt werden, wenn Anschluß A und B des 8255-Chips den Handshake-Betrieb benutzen	Lesen oder schreiben	Keine	Software-Abfrage
PC-OPDIO-16 (Plug and Play)	Optisch isolierter 8-Bit-Anschluß	0, 1	—	Anschluß 0 ist Ausgabe (schreiben); Anschluß 1 ist Eingabe (lesen)	Keine	Programmiertes I/O

Hardware-Leistungsfähigkeiten von Nur-Timing-Geräten

Tabelle B-30. Digitale Hardware-Leistungsfähigkeiten—Nur-Timing-Geräte

Gerät	Anschluß-typ	Anzahl der Anschlüsse	Handshake-Modi	Richtung	DIO-Takte	Übertragungs methode
PC-TIO-10 NB-TIO-10	8-Bit-Anschlüsse	0, 1	Kein Handshake -Betrieb	Bitweise Richtungs-steuerung	Keine	Software-Abfrage

Tabelle B-31. Counter-/Timer-Eigenschaften—Nur-Timing-Geräte

Gerät	Verwendete Counter-Chip	Anzahl verfügbarer Allzweck-Counter	Verfügbare Zeitbasen	Anzahl der Bits	Verfügbare Gate-Modi	Verfügbare Ausgänge	Verfügbare Ausgangsmodi	Zählrichtung
PC-TIO-10 NB-TIO-10	Am-9513	10 (8 haben SOURCE-Eingabe am I/O-Anschluß)	Intern: 5 MHz (nur auf CTR5 und CTR10), 1 MHz, 100 kHz, 10 kHz, 1 kHz, 100 Hz; extern	16	High Level, Low Level, ansteigende Flanke, absteigende Flanke	10	TC-Impuls, TC umschalten	Aufwärts oder abwärts

Hardware-Leistungsfähigkeiten der 5102-Geräte

Tabelle B-32. Konfigurationsprogrammierbarkeit der Analogeingabe

Gerät	Verstärkung	Kopplung
5102-Geräte	Nach Kanal	Nach Kanälen

Tabelle B-33. Analogeingabe-Eigenschaften

Gerät	Anzahl der Kanäle	Auflösung	Verstärkung	Bereich(V)	Eingangs- FIFO (Worte)	Scanning
5102-Geräte	2	8 Bits	1, 5, 20, 100	±5	663 000	1 oder 2 Kanäle, in beliebiger Reihenfolge ohne Wiederholungen

Tabelle B-34. Analogeingabe-Eigenschaften, Teil 2

Gerät	Trigger	Max. Abtastrate (S/s)
5102-Geräte	SW, Pre, Post, Analog	20.000.000 Echtzeit

C

GPIB mehrzeilige Schnittstellen-Nachrichten

In diesem Anhang werden mehrzeilige Schnittstellen-Nachrichten aufgeführt; hierbei handelt es sich um von IEEE 488 definierte Befehle. GPIB wird durch mehrzeilige Schnittstellen-Nachrichten gesteuert; sie führen Aufgaben durch, wie z.B. die Businitialisierung sowie die Adressierung von Geräten und zur Aufhebung dieser Adressierung und dem Einstellen von Gerätemodi für Lokal- und Remote-Programmierung. Diese mehrzeiligen Schnittstellen-Nachrichten werden mit ATN TRUE gesendet und empfangen. In folgender Liste sind die Gedächtnishilfen und Nachrichten enthalten, die den Schnittstellenfunktionen entsprechen.

Weitere Informationen zu diesen Nachrichten finden Sie in ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation.*

Mehrzeilige Schnittstellen-Nachrichten

Hex	Oct	Dec	ASCII	Nach-richt
00	000	0	NUL	—
01	001	1	SOH	GTL
02	002	2	STX	—
03	003	3	ETX	—
04	004	4	EOT	SDC
05	005	5	ENQ	PPC
06	006	6	ACK	—
07	007	7	BEL	—

Hex	Oct	Dec	ASCII	Nach-richt
08	010	8	BS	GET
09	011	9	HT	TCT
0A	012	10	LF	—
0B	013	11	VT	—
0C	014	12	FF	—
0D	015	13	CR	—
0E	016	14	SO	—
0F	017	15	SI	—
10	020	16	DLE	—
11	021	17	DC1	LLO
12	022	18	DC2	—
13	023	19	DC3	—
14	024	20	DC4	DCL
15	025	21	NAK	PPU
16	026	22	SYN	—
17	027	23	ETB	—
18	030	24	CAN	SPE
19	031	25	EM	SPD
1A	032	26	SUB	—
1B	033	27	ESC	—
1C	034	28	FS	—
1D	035	29	GS	—
1E	036	30	RS	—
1F	037	31	US	—
20	040	32	SP	MLA0

Hex	Oct	Dec	ASCII	Nach-richt
21	041	33	!	MLA1
22	042	34	“	MLA2
23	043	35	#	MLA3
24	044	36	$	MLA4
25	045	37	%	MLA5
26	046	38	&	MLA6
27	047	39	’	MLA7
28	050	40	(	MLA8
29	051	41	)	MLA9
2A	052	42	*	MLA10
2B	053	43	+	MLA11
2C	054	44	,	MLA12
2D	055	45	-	MLA13
2E	056	46	.	MLA14
2F	057	47	/	MLA15
30	060	48	0	MLA16
31	061	49	1	MLA17
32	062	50	2	MLA18
33	063	51	3	MLA19
34	064	52	4	MLA20
35	065	53	5	MLA21
36	066	54	6	MLA22
37	067	55	7	MLA23
38	070	56	8	MLA24
39	071	57	9	MLA25

Hex	Oct	Dec	ASCII	Nach-richt
3A	072	58	:	MLA26
3B	073	59	;	MLA27
3C	074	60	<	MLA28
3D	075	61	=	MLA29
3E	076	62	>	MLA30
3F	077	63	?	UNL
40	100	64	@	MTA0
41	101	65	A	MTA1
42	102	66	B	MTA2
43	103	67	C	MTA3
44	104	68	D	MTA4
45	105	69	E	MTA5
46	106	70	F	MTA6
47	107	71	G	MTA7
48	110	72	H	MTA8
49	111	73	I	MTA9
4A	112	74	J	MTA10
4B	113	75	K	MTA11
4C	114	76	L	MTA12
4D	115	77	M	MTA13
4E	116	78	N	MTA14
4F	117	79	O	MTA15
50	120	80	P	MTA16
51	121	81	Q	MTA17

Hex	Oct	Dec	ASCII	Nach-richt
52	122	82	R	MTA18
53	123	83	S	MTA19
54	124	84	T	MTA20
55	125	85	U	MTA21
56	126	86	V	MTA22
57	127	87	W	MTA23
58	130	88	X	MTA24
59	131	89	Y	MTA25
5A	132	90	Z	MTA26
5B	133	91	[	MTA27
5C	134	92	\	MTA28
5D	135	93	]	MTA29
5E	136	94	^	MTA30
5F	137	95	_	UNT
60	140	96	`	MSA0,PPE
61	141	97	a	MSA1,PPE
62	142	98	b	MSA2,PPE
63	143	99	c	MSA3,PPE
64	144	100	d	MSA4,PPE
65	145	101	e	MSA5,PPE
66	146	102	f	MSA6,PPE
67	147	103	g	MSA7,PPE
68	150	104	h	MSA8,PPE
69	151	105	i	MSA9,PPE

Hex	Oct	Dec	ASCII	Nach-richt
6A	152	106	j	MSA10,PPE
6B	153	107	k	MSA11,PPE
6C	154	108	l	MSA12,PPE
6D	155	109	m	MSA13,PPE
6E	156	110	n	MSA14,PPE
6F	157	111	o	MSA15,PPE
70	160	112	p	MSA16,PPD
71	161	113	q	MSA17,PPD
72	162	114	r	MSA18,PPD
73	163	115	s	MSA19,PPD
74	164	116	t	MSA20,PPD
75	165	117	u	MSA21,PPD
76	166	118	v	MSA22,PPD
77	167	119	w	MSA23,PPD
78	170	120	x	MSA24,PPD
79	171	121	y	MSA25,PPD
7A	172	122	z	MSA26,PPD
7B	173	123	{	MSA27,PPD
7C	174	124	|	MSA28,PPD
7D	175	125	}	MSA29,PPD
7E	176	126	~	MSA30,PPD
7F	177	127	DEL	—

Nachrichtendefinitionen

Gedächtnisstütze	Definition
DCL	Gerät reinitialisieren
GET	Gruppenausführung Trigger
GTL	Geh zu lokal
LLO	Lokaler Lockout
MLA	Meine Höradresse
MSA	Meine Zweitadresse
MTA	Meine Sprechadresse
PPC	Parallelabfrage konfigurieren
PPD	Parallelabfrage deaktivieren
PPE	Parallelabfrage aktivieren
PPU	Parallelabfrage dekonfigurieren
SDC	Ausgewähltes Gerät re-initialisieren
SPD	Serielle Abfrage deaktivieren
SPE	Serielle Abfrage aktivieren
TCT	Steuerung übernehmen
UNL	Hören deaktivieren
UNT	Sprechen deaktivieren

D

Kundenbetreuung

Um Ihnen im Falle technischer Probleme die Zusammenarbeit mit uns zu erleichtern, sind in diesem Anhang Formulare enthalten, die Ihnen dabei helfen, die technischen Informationen zusammenzustellen, die wir zur Lösung Ihrer technischen Probleme benötigen. Außerdem ist ein Formular enthalten, auf dem Sie Kommentare zur Produktdokumentation abgeben können. Wenn Sie sich mit uns in Verbindung setzen, benötigen wir die Informationen auf dem Formular für Technische Unterstützung sowie auf dem Konfigurationsformular, soweit dies in Ihrem Handbuch enthalten ist, mit dessen Hilfe wir Ihre Fragen zur Systemkonfiguration so schnell wie möglich beantworten.

National Instruments bietet technische Unterstützung auf elektronischem Wege und per Fax und Telefon und läßt Ihnen somit die von Ihnen benötigten Informationen schnellstmöglich zukommen. Zu unserem elektronischen Service gehört ein Bulletin-Board, eine FTP-Site, ein Faxabrufsystem sowie E-Mail-Unterstützung. Versuchen Sie bei Hardware- oder Softwareproblemen erst unser elektronisches Unterstützungssystem. Sollten Ihre Fragen durch die hier zur Verfügung stehenden Informationen nicht beantwortet werden, bieten wir Ihnen durch unsere mit Anwendungsingenieuren besetzten technischen Support-Zentren per Fax und Telefon Hilfe.

Elektronischer Service

Bulletin Board Support

National Instruments hat BBS- und FTP-Sites, die Ihnen rund um die Uhr zur Verfügung stehen und auf denen sich Dateien und Dokumente mit Antworten zu den am häufigsten vorkommenden Kundenfragen befinden. Von diesen Sites können Sie auch die neuesten Instrumententreiber sowie Updates und Beispielprogramme herunterladen. Aufgezeichnete Anweisungen zur Benutzung des BBS- und FTP-Service sowie automatisierte BBS-Informationen erhalten Sie unter der Tel.Nr. USA 1-512 795 6990. Sie können auf diesen Service folgendermaßen zugreifen:

Vereinigte Staaten: 1-512 794 5422
Bis zu 14.400 Baud, 8 Datenbits, 1 Stoppbit, keine Parität

Großbritannien: 01635 551422
Bis zu 9.600 Baud, 8 Datenbits, 1 Stoppbit, keine Parität

Frankreich: 01 48 65 15 59
Bis zu 9.600 Baud, 8 Daten bits, 1 Stoppbit, keine Parität

FTP-Unterstützung

Melden Sie sich zum Zugriff auf unsere FTP-Site bei unserem Internet-Host `ftp.natinst.com`, `anonymous` an und benutzen Sie Ihre E-Mail-Adresse, wie z.B. `joesmith@anywhere.com` als Paßwort. Die Unterstützungsdateien und -dokumente befinden sich in den Verzeichnissen `/support`.

Faxabruf-Unterstützung

Bei der Faxabfrage handelt es sich um ein Informationssystem, das rund um die Uhr zugänglich ist und eine Bibliothek an Dokumenten zu einem umfassenden Bereich technischer Informationen enthält. Sie können auf die Faxabfrage von einem Telefon mit Tonwahl unter der folgenden Nummer zugreifen: 1-512 418 1111.

Unterstützung per E-Mail

Sie können dem Anwendungstechnikerteam Fragen zur technischen Unterstützung per E-Mail an unsere unten aufgeführte Internet-Adresse zukommen lassen. Bitte vergessen Sie nicht, Ihren Namen, Anschrift und Telefonnummer anzugeben, damit wir Ihnen unsere Lösungen und Vorschläge mitteilen können.

`ni.germany@natinst.com`

Unterstützung per Telefon und Fax

National Instruments unterhält auf der ganzen Welt Niederlassungen. In nachstehender Liste finden Sie die Nummer für technische Unterstützung für Ihr Land. Sollte in Ihrem Land kein Büro von National Instruments bestehen, setzen Sie sich bitte mit der Stelle in Verbindung, von der Sie die Software gekauft haben, für die Sie Unterstützung benötigen.

Land	Telefon	Fax
Australien	03 9879 5166	03 9879 6277
Belgien	02 757 00 20	02 757 03 11
Brasilien	011 288 3336	011 288 8528
Dänemark	45 76 26 00	45 76 26 02
Deutschland	089 741 31 30	089 714 60 35
Finnland	09 725 725 11	09 725 725 55
Frankreich	01 48 14 24 24	01 48 14 24 14
Großbritannien	01635 523545	01635 523154
Hongkong	2645 3186	2686 8505
Israel	03 6120092	03 6120095
Italien	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Kanada (Ontario)	905 785 0085	905 785 0086
Kanada (Québec)	514 694 8521	514 694 4399
Korea	02 596 7456	02 596 7455
Mexiko	5 520 2635	5 520 3282
Niederlande	0348 433466	0348 430673
Norwegen	32 84 84 00	32 84 86 00
Österreich	0662 45 79 90 0	0662 45 79 90 19
Singapur	2265886	2265887
Spanien	91 640 0085	91 640 0533
Schweden	08 730 49 70	08 730 43 70
Schweiz	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
Vereinigte Staaten	512 795 8248	512 794 5678

Formular für technische Unterstützung

Fotokopieren Sie dieses Formular und bringen Sie es jedes Mal, wenn Sie Änderungen an Ihrer Software oder Hardware vornehmen, auf den neuesten Stand. Verwenden Sie die ausgefüllte Kopie dieses Formulars als Nachschlagequelle für Ihre derzeitige Konfiguration. Wenn Sie dieses Formular sorgfältig ausfüllen, bevor Sie sich wegen technischer Unterstützung mit National Instruments in Verbindung setzen, helfen Sie damit unseren Anwendungsingenieuren, Ihre Fragen effizienter zu beantworten.

Sollten Sie Hardware- oder Softwareprodukte von National Instruments verwenden, die mit diesem Problem zusammenhängen, legen Sie bitte Konfigurationsformulare aus den jeweiligen Handbüchern bei. Verwenden Sie, wenn nötig, zusätzliche Seiten.

Name _____

Firma _____

Adresse _____

_____

Fax (___) _____Tel. (___) _____

Computermarke_____ Modell _____Prozessor_____

Betriebssystem (inkl. Versionsnr.) _____

Taktgeschw. _____MHz RAM _____MB Anzeigeadapter_____

Maus ___ja ___nein Andere installierte Adapter_____

Festplatten-Kapazität _____MB Marke_____

Verwendete Instrumente _____

_____

National Instruments Hardware-Produktmodell _____ Revision _____

Konfiguration _____

National Instruments Softwareprodukt _____ Version _____

Konfiguration _____

Beschreibung des Problems: _____

_____

_____

_____

_____

Bitte alle Fehlermeldungen aufführen: _____

_____

_____

Durch folgende Schritte wird das Problem wiederholt: _____

_____

_____

_____

_____

LabVIEW-Hardware- und Software-Konfigurationsformular

Notieren Sie die Einstellungen und Revisionen Ihrer Hardware und Software, die sich in der Zeile rechts von jedem Artikel befindet. Füllen Sie bei jeder Änderung an Ihrer Software- oder Hardware-Konfiguration jedes Mal eine neue Kopie dieses Formulars aus, und benutzen Sie dieses Formular als Nachschlagequelle für Ihre derzeitige Konfiguration. Wenn Sie dieses Formular sorgfältig ausfüllen, bevor Sie sich wegen technischer Unterstützung mit National Instruments in Verbindung setzen, helfen Sie damit unseren Anwendungsingenieuren, Ihre Fragen effizienter zu beantworten.

National Instruments Produkte

DAQ-Hardware _____

Interrupt-Level der Hardware _____

DMA-Kanäle der Hardware _____

Basis-I/O-Adresse der Hardware _____

Programmierungswahl _____

HiQ-, NI-DAQ-, LabVIEW- oder LabWindows-Version _____

Andere Karten in Ihrem System _____

Basis-I/O-Adresse anderer Karten _____

DMA-Kanäle anderer Karten _____

Interrupt-Level anderer Karten _____

Andere Produkte

Computer-Hersteller- und Modell _____

Mikroprozessor _____

Taktfrequenz oder -geschwindigkeit_____

Typ der installierten Videokarte _____

Betriebssystemversion _____

Betriebssystemmodus _____

Programmiersprache _____

Programmiersprachenversion _____

Andere Karten in Ihrem System _____

Basis-I/O-Adresse anderer Karten _____

DMA-Kanäle anderer Karten _____

Interrupt-Level anderer Karten _____

Formular für Kommentare zur Dokumentation

National Instruments bittet Sie um Ihre Meinung zu der mit unseren Produkten mitgelieferten Dokumentation. Mit Hilfe dieser Informationen sind wir besser in der Lage, die von Ihnen benötigten Qualitätsprodukte zu liefern.

Titel: *LabVIEW™ Funktionen- und VI-Referenzhandbuch*

Ausgabe: Juli 1998

Artikelnummer: 321988A-01

Ihre Meinung zu Vollständigkeit, Klarheit und Aufbau des Handbuchs.

_____
_____
_____
_____
_____
_____
_____

Falls Sie im Handbuch Fehler entdecken, geben Sie bitte die Seite an und beschreiben Sie den Fehler.

_____
_____
_____
_____
_____
_____
_____

Wir bedanken uns für Ihre Hilfe.

Name _____

Titel _____

Firma _____

Adresse _____

_____

Tel. (___) _____ Fax (___) _____

E-Mail-Adresse _____

Senden an: Technical Publications
National Instruments Germany GmbH
Konrad-Celtis-Str. 79
81369 München

Fax an: Technical Publications
National Instruments Germany GmbH
(089) 714 60 35

Stichwörterverzeichnis

Nummern

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z