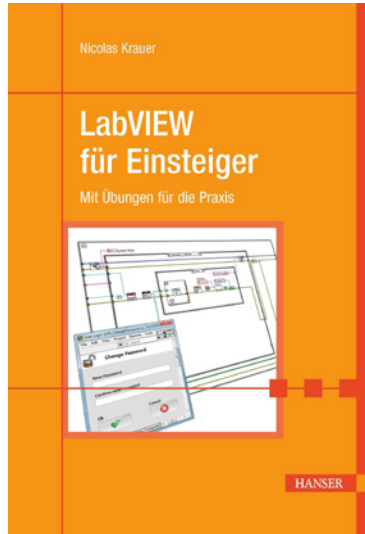


# HANSER



## Leseprobe

zu

## LabVIEW für Einsteiger

von Nicolas Krauer

ISBN (Buch): 978-3-446-45906-9

ISBN (E-Book): 978-3-446-45946-5

Weitere Informationen und Bestellungen unter  
<http://www.hanser-fachbuch.de/978-3-446-45906-9>  
sowie im Buchhandel

© Carl Hanser Verlag, München

# Vorwort

Dieses Lehrbuch entstammt einem Skript, welches speziell für den LabVIEW-Unterricht an Berufs- und höheren Fachschulen konzipiert wurde. Dabei stand ein starker Bezug zur Praxis und der täglichen Arbeit mit LabVIEW im Vordergrund.

Meine Tätigkeit als Dozent an der höheren Fachschule Uster sowie die Erfahrung als Dienstleister für LabVIEW-Applikationen erlauben einen Blick auf die Thematik von unterschiedlichen Seiten. Das gibt mir die Chance, die Grundlagen so vermitteln, wie sie auch in der Praxis später zur Anwendung kommen.

Dies ist auch der Hauptunterschied zu anderen Lehrmitteln, welche teils sehr vertieft die theoretischen Grundlagen von LabVIEW thematisieren. Dies hat durchaus seine Berechtigung, wenn man sich über einen längeren Zeitraum Wissen aneignen will, ist allerdings für einen raschen Einstieg mit baldigen Programmiererfolgen eher ungeeignet.

Der Inhalt besteht sowohl aus der Beschreibung grundsätzlicher Konzepte grafischer Programmierung als auch weiterführenden Anwendungen und Best-Practice-Anwendungen aus der Industrie. Dabei erhebt das Buch keinerlei Anspruch auf Vollständigkeit, was aufgrund des großen Anwendungsbereichs und der vielfältig nutzbaren Funktionen auch nicht realistisch ist. Behandelt werden die aus meiner Sicht grundlegenden Thematiken, die es dem LabVIEW-Einsteiger ermöglichen, möglichst schnell und mit Rücksicht auf aktuelle Designrichtlinien lauffähige Mess- und Testumgebungen zu realisieren. Sehr spezifische Themen werden dabei nicht berücksichtigt. Bedingte Grundkenntnisse über programmiertechnische Begriffe werden allerdings vorausgesetzt. Es wird speziell darauf geachtet, einen konsequenten Programmierstil, wie ihn National Instruments vorschlägt, umzusetzen.

Jedes Kapitel enthält eine Sammlung von Übungen, welche dem Benutzer die jeweils praktische Umsetzung der in diesem Buch behandelten Themen in LabVIEW ermöglicht. Es wird sehr empfohlen, diese Übungen durchzuführen, um sich nach und nach mit LabVIEW vertraut zu machen. Die Grundlagen der ersten Kapitel werden für die Durchführung von Übungen späterer Kapitel benötigt.

Die Musterlösungen zu sämtlichen Übungen sowie zusätzliche Demos zu einzelnen Kapiteln lassen sich als Softwarepaket unter <https://www.krauer-engineering.ch/labview-fuer-einsteiger/> herunterladen. Zur Bearbeitung benötigen Sie lediglich ein Tool, um die ZIP-Datei zu entpacken, sowie die Entwicklungsumgebung LabVIEW, welche Ihnen mit diesem Buch zur Verfügung gestellt wird. Die Lösungen der Übungen stellen immer nur eine mögliche Variante dar. Es gibt fast immer mehrere Lösungswege und Möglichkeiten, eine Aufgabe zu bearbeiten.

Sämtliche Illustrationen wurden mit der englischen Version LabVIEW 2015.0f2 generiert. Da die englische Version in der Industrie weit verbreitet ist, sind die meisten spezifischen Begriffe ebenfalls in Englisch gehalten. Eine Vermischung kann allerdings nicht vollkommen vermieden werden, weshalb sich durchaus auch deutsche Begriffe für Elemente und Funktionen in LabVIEW in diesem Buch wiederfinden.

Danksagungen möchte ich vor allem Manuel Leppert vom Hanser Verlag aussprechen. Er hat sich für dieses Buchprojekt viel Zeit genommen und war mir stets eine verlässliche Anlaufstelle bei Fragen. Ein ebenso großer Dank gebührt all den Studierenden, die mich auf Fehler oder Ungereimtheiten in meinem Unterrichtsskript hingewiesen und damit zur wesentlichen Verbesserung in Form dieses Lehrmittels beigetragen haben.

Abschließend möchte ich Dr. Fabian Wehnekamp von National Instruments für die Unterstützung dieses Buchprojekts danken.

Ich wünsche allen Leserinnen und Lesern viel Erfolg mit der Arbeit dieses Buches.

Rapperswil-Jona, im November 2018

Nicolas Krauer

## ■ Hinweis des Verlags

Sehr geehrte/r Leser/in,

die für das Selbststudium mit diesem Buch notwendige Software LabVIEW können Sie über die folgenden Optionen beziehen:

### 1. Bezug der Student Install Option:

Als Student einer Hochschule, welche eine Academic Site Licence vorhält, haben Sie die Möglichkeit über die sog. Student Install Option eine jährliche Lizenz über ihre Hochschule zu beziehen, so lange Sie bei dieser eingeschrieben sind. Die Student Install Option umfasst das gleiche Softwarepaket wie die Academic Site Licence (<http://www.ni.com/white-paper/8115/en/>). Informationen hierzu finden Sie unter dem folgenden Link (<https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019MbaSAE>). Der Bezug dieser Lizenz erfolgt direkt über Ihre Hochschule und nicht über National Instruments. Bitte kontaktieren Sie hierfür Ihre Lokale IT oder Ihren LabVIEW Academy Instructor.

### 2. Bezug einer Evaluierungslizenz:

Sie haben die Möglichkeit über [ni.com/download](http://ni.com/download) die aktuellste LabVIEW Version zur Evaluierung herunterzuladen. Nach der Installation können Sie die Software 7 Tage lang kostenfrei nutzen. Durch die Verknüpfung der Installation mit einem ni.com-Profil haben Sie die Möglichkeit, die Evaluierung auf 45 Tage zu verlängern.

### 3. Bezug der LabVIEW Student Edition/LabVIEW Home Edition:

Sie haben die Möglichkeit, eine Lizenz der Software über den lokalen Reseller, <https://www.conrad.de> zu erwerben. Hierbei haben Sie die Wahl zwischen der LabVIEW Student Edition (erfordert einen Immatrikulationsnachweis) oder der LabVIEW Home Edition. Diese Softwarepakete umfassen das LabVIEW Full Development System, das Control Design and Simulation sowie das MathScript RT Module.

Wir wünschen Ihnen viel Erfolg beim Einstieg in die Welt der grafischen Programmierung.

Carl Hanser Verlag

# Inhalt

<b>1</b>	<b>Entwicklungsumgebung</b> .....	<b>11</b>
1.1	Was ist LabVIEW? .....	11
1.2	Grundeinstellungen .....	12
1.3	Bestandteile eines VIs .....	13
1.3.1	Frontpanel .....	14
1.3.2	Blockdiagramm .....	17
1.3.3	Symbol und Anschlussblock .....	19
1.4	Kontexthilfe .....	21
1.5	Paletten .....	22
1.6	Werkzeuge .....	27
1.7	Projekt Explorer .....	28
1.7.1	Organisation mit virtuellen Ordnern .....	29
1.7.2	Arbeiten mit dem Projekt Explorer .....	31
1.7.3	Erstellen von Exe-Files .....	34
1.7.4	Erstellen von Installern .....	37
1.8	Tipps und Tricks beim Arbeiten mit LabVIEW .....	42
1.9	Übungsaufgaben .....	44
<b>2</b>	<b>Grundlegende Konzepte</b> .....	<b>49</b>
2.1	Datenfluss .....	49
2.2	Datentypen .....	51
2.2.1	Numerische Datentypen .....	51
2.2.2	Boolean .....	57
2.2.3	String .....	58
2.2.4	Path .....	60
2.2.5	Enum & Ring .....	62
2.2.6	Weitere Datentypen .....	64
2.3	Übungsaufgaben .....	65

<b>3</b>	<b>Debugging</b> .....	<b>69</b>
3.1	Fehlerhafte VIs .....	69
3.2	Debugging-Methoden .....	70
3.2.1	Highlight Execution .....	71
3.2.2	Retain Wire Values .....	71
3.2.3	Stepping .....	71
3.2.4	Breakpoints .....	72
3.2.5	Probes .....	72
3.3	Übungsaufgabe .....	74
<b>4</b>	<b>Modularität und SubVIs</b> .....	<b>75</b>
4.1	Prinzip der Modularität .....	75
4.2	Erstellung von SubVIs .....	77
4.3	Einbettung von SubVIs .....	80
4.4	Dokumentation .....	82
4.4.1	Dokumentation von SubVIs .....	82
4.4.2	Dokumentation von Elementen .....	83
4.4.3	In-Code-Dokumentation .....	84
4.4.4	Best Practice Dokumentation .....	85
4.5	Übungsaufgaben .....	86
<b>5</b>	<b>Loops</b> .....	<b>89</b>
5.1	While-Loops .....	89
5.2	For-Loops .....	90
5.3	Timing von Loops .....	91
5.4	Tunnel .....	92
5.5	Schieberegister .....	95
5.6	Übungsaufgaben .....	96
<b>6</b>	<b>Entscheidungsstrukturen</b> .....	<b>101</b>
6.1	Case-Struktur .....	101
6.2	Eventstruktur .....	103
6.2.1	Verwendung der Eventstruktur .....	103
6.2.2	Melder- und Filter-Events .....	106
6.3	Übungsaufgaben .....	107

<b>7</b>	<b>Strukturierte Daten</b>	<b>111</b>
7.1	Arrays	111
7.1.1	Darstellung	111
7.1.2	Statische Erstellung von Arrays	112
7.1.3	Programmatische Erstellung von Arrays	113
7.1.4	Bearbeitung und Manipulation von Arrays	114
7.1.5	Polymorphie bei Arrays	115
7.2	Cluster	117
7.2.1	Erstellung von Clustern	117
7.2.2	Sortierung von Clustern	118
7.2.3	Verwendung von Clustern	119
7.3	Error-Cluster	121
7.4	Typdefinitionen	124
7.4.1	Erstellung einer Typdefinition	125
7.4.2	Unterscheidung zwischen Control, Type Def und Strict Type Def	126
7.5	Variant	128
7.5.1	Verwendung von Variant	128
7.5.2	Attribute von Variant	129
7.6	Übungsaufgaben	130
<b>8</b>	<b>Visualisierung von Daten</b>	<b>135</b>
8.1	Eigenschaften von grafischen Anzeigen	136
8.2	Arten von grafischen Anzeigen	138
8.2.1	Waveform Chart	138
8.2.2	Waveform Graph	140
8.2.3	XY-Graph	142
8.3	Übungsaufgaben	143
<b>9</b>	<b>Dateibearbeitung</b>	<b>147</b>
9.1	Schreiben und Lesen von Dateien	147
9.2	Spreadsheets	150
9.3	Pfade	152
9.4	Technical Data Management Streaming (TDMS)	153
9.4.1	Aufbau der TDMS-Struktur	153
9.4.2	Anwendung von TDMS	154
9.4.3	Eigenschaften	156
9.5	Konfigurationsfiles	157
9.6	Best Practice File-I/O	159
9.7	Übungsaufgaben	160

<b>10</b>	<b>Steuerung der Benutzeroberfläche</b>	<b>167</b>
10.1	VI-Server-Architektur	167
10.2	Property Nodes	169
10.2.1	Implizite Property Nodes	169
10.2.2	Explizite Property Nodes	170
10.2.3	Anwendung von Property Nodes	170
10.3	Invoke Nodes	172
10.4	Übungsaufgaben	173
<b>11</b>	<b>Datenerfassung mit NI-Hardware</b>	<b>185</b>
11.1	Übersicht über die Hardware	185
11.2	Measurement & Automation Explorer	186
11.3	Die DAQmx-Palette	189
11.3.1	Task erstellen	190
11.3.2	Task lesen (Analog Input)	192
11.3.3	Task schreiben (Analog Output)	193
11.3.4	Task löschen	193
11.4	Übungsaufgaben	194
<b>12</b>	<b>Synchronisation von Prozessen</b>	<b>199</b>
12.1	Synchronisation ohne Datenaustausch	200
12.1.1	Occurrence	200
12.1.2	Semaphore	201
12.1.3	Rendezvous	202
12.2	Synchronisation mit Datenaustausch	203
12.2.1	Notifier	204
12.2.2	Queues	206
12.3	Übungsaufgaben	209
<b>13</b>	<b>Entwurfsmuster (Design Patterns)</b>	<b>211</b>
13.1	State Machine	211
13.1.1	State Machine mit Enum	213
13.1.2	State Machine mit String	215
13.2	Queued Message Handler	216
13.3	Übungsaufgaben	217
	<b>Index</b>	<b>231</b>

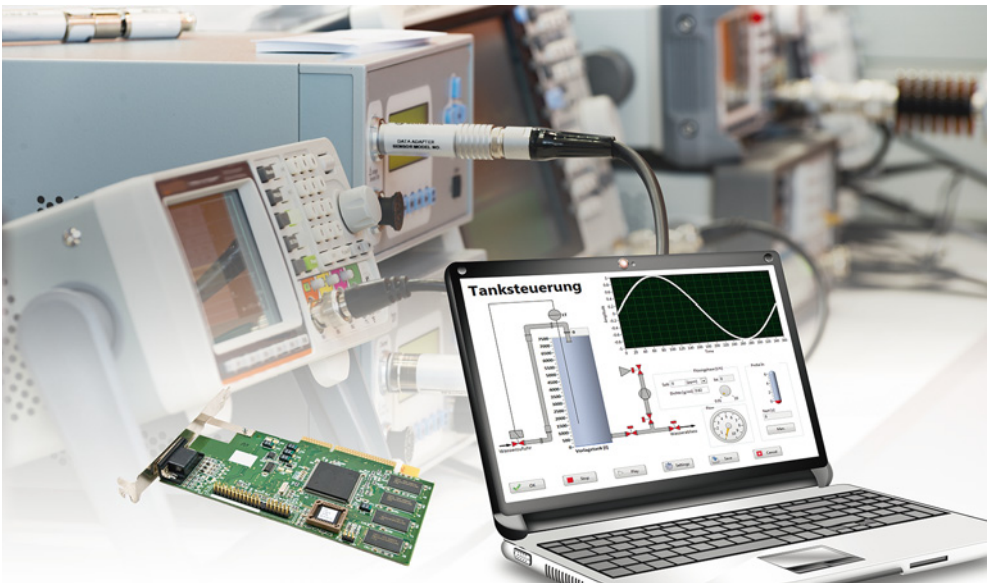
# 1

## Entwicklungsumgebung

Dieses Kapitel dient als Einstieg in die Entwicklungsumgebung. Es werden die grundlegenden Funktionen erklärt, der Umgang mit den Tools von LabVIEW erläutert und erste Projekte erstellt. Mit den Übungsaufgaben am Ende des Kapitels kann das LabVIEW-Handling geübt und das angeeignete Wissen vertieft werden.

### ■ 1.1 Was ist LabVIEW?

LabVIEW ist eine grafische Programmierumgebung zum Entwickeln von anspruchsvollen Mess-, Prüf-, Steuer- und Regelaufgaben. Dabei wird auf die sogenannte *Virtual Instrumentation* (virtuelle Instrumentierung) von Messgeräten gesetzt, also reale Messgeräte durch Softwaretools abgebildet und miteinander kombiniert. So können nicht wie bei einem konventionellen Multimeter nur Strom und Spannung gemessen, sondern auch gleich noch ein Oszilloskop, ein Generator oder andere beliebige Messgeräte in einer einzigen Applikation verwendet werden (Bild 1.1).



**Bild 1.1** Eine einzelne Applikation zur Erfassung, Auswertung und Darstellung

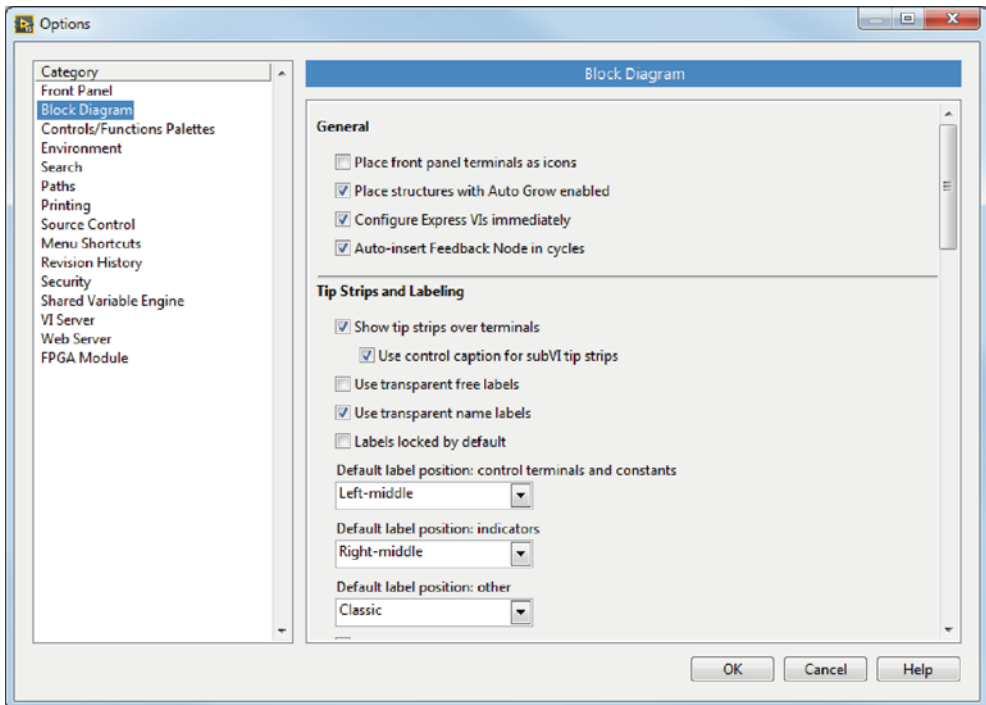
Ein LabVIEW-Programm oder eine einzelne Datei oder Funktion wird deshalb auch als VI (*Virtual Instrument*) bezeichnet. Die Dateiendung solcher Dateien lautet ebenfalls *.vi*.



Es findet also mit LabVIEW eine Vernetzung von klassischen Messgeräten mit der IT-Welt statt. Dabei können, sofern es die entsprechende Hardware zulässt, eine sehr große Anzahl von Datenkanälen parallel erfasst und verarbeitet werden. LabVIEW arbeitet mit einer Vielzahl an Hardwareprodukten und beinhaltet eine große Anzahl von Analysefunktionen.

## ■ 1.2 Grundeinstellungen

Wer die Entwicklungsumgebung nach eigenen Vorlieben konfigurieren möchte, hat unter *Tools* → *Options* die Möglichkeit, verschiedenste Grundeinstellungen vorzunehmen. Auf der linken Seite wählt man die Kategorie, während rechts die Optionen zur entsprechenden Kategorie zu sehen sind (Bild 1.2).



**Bild 1.2** Einstellungen des Blockdiagramms

Die meisten Optionen können auf dem Standard belassen werden. Lediglich für das Blockdiagramm bietet es sich an, darauf zu achten, dass die Checkbox *Place front panel terminals as icons* nicht selektiert ist. Dies spart Platz und erhöht die Übersicht im Blockdiagramm.

## ■ 1.3 Bestandteile eines VIs

Ein LabVIEW-Programm kann unter Umständen aus nur einem einzigen VI, also einer einzelnen Datei bestehen. Das VI unterscheidet drei wesentliche Hauptbestandteile (Bild 1.3):

- **Frontpanel:** ist die Bedienoberfläche, enthält Controls (Bedienelemente, zur Eingabe von Daten) und Indicators (Anzeigeelemente, zur Ausgabe von Daten)
- **Blockdiagramm:** enthält den grafischen Quellcode und damit die Funktion des VIs, enthält Anschlüsse für Controls und Indicators auf dem Frontpanel
- **Symbol/Anschlussblock:** repräsentiert das VI und ermöglicht die Verwendung als SubVI, zeigt die Belegung der Ein- und Ausgänge des VIs an

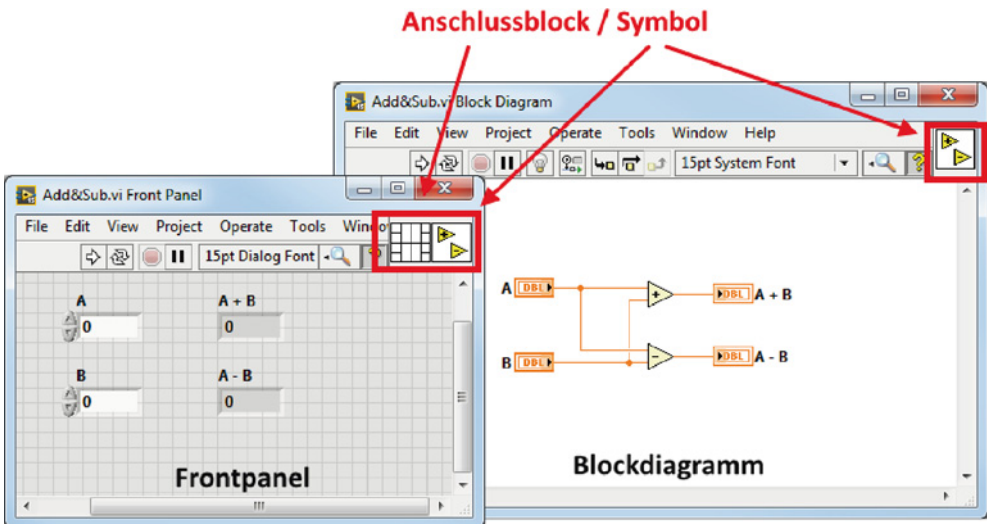
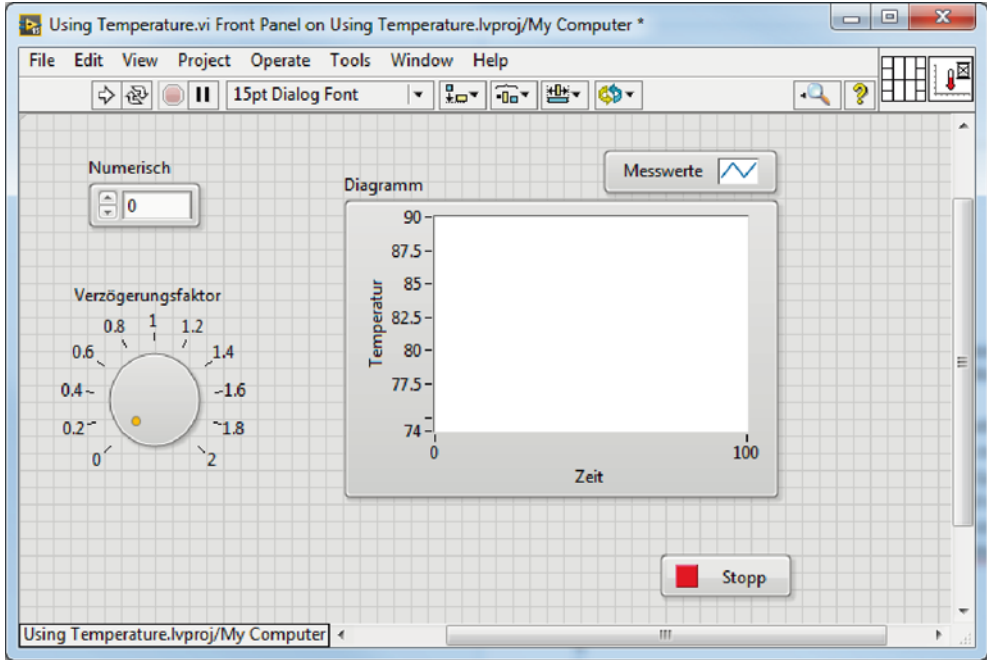


Bild 1.3 Hauptbestandteile eines VIs

### 1.3.1 Frontpanel

Das Frontpanel stellt in LabVIEW die Schnittstelle zum Benutzer dar oder zu anderen VIs. Es beinhaltet sämtliche Eingänge und Ausgänge, die für eine Interaktion mit der Welt außerhalb des VIs benötigt werden (Bild 1.4).



**Bild 1.4** Frontpanel eines VIs

Die Elemente werden in zwei Gruppen unterteilt: in Bedien- und Anzeigeelemente (Tabelle 1.1).

**Tabelle 1.1** Controls (Bedienelemente) und Indicators (Anzeigeelemente)

Controls	Indicators
Buttons, Drehknöpfe, Schieberegler, Texteingabefelder etc.	LEDs, Graphen, Diagramme, Textboxen etc.





Die Symbolleiste des Frontpanels enthält verschiedene Funktionen zur Ausführung des VIs sowie zur Gestaltung und Anordnung der vorhandenen Frontpanel-Elemente (Bild 1.5).



**Bild 1.5** Symbolleiste des Frontpanels

Die in [Tabelle 1.2](#) beschriebenen Buttons dienen der Kontrolle der Ausführung des entsprechenden VIs.

**Tabelle 1.2** Funktionelle Buttons der Symbolleiste

	Ausführen des VIs
	Wiederholtes Ausführen des VIs
	Sofortiger Abbruch der Ausführung des VIs
	Pausieren der Ausführung des VIs

Mit dem Dropdown-Menü, gleich daneben, lassen sich Schriftarten der Frontpanel-Elemente und deren Aussehen verändern ([Bild 1.6](#)).

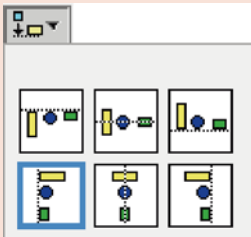
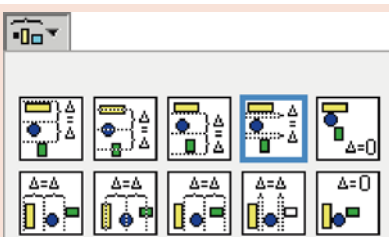


**Bild 1.6** Schrifteditor

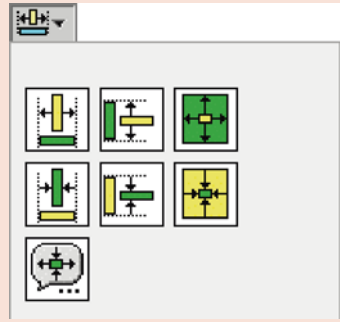
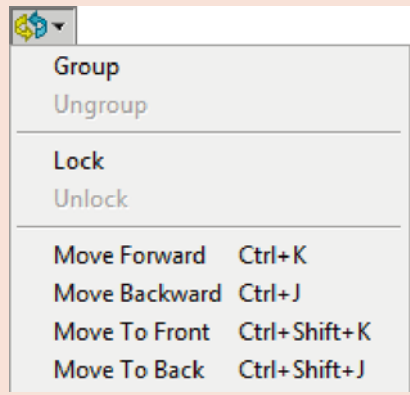
Die weiteren Funktionen dienen der optimierten Darstellung und der Ausrichtung der Elemente auf dem Frontpanel und haben keinen Einfluss auf das Programmverhalten des VIs. Trotzdem sollte ein Frontpanel ansprechend sein und alle Controls und Indicators nach einem übersichtlichen Muster angeordnet sein. Die folgenden Tools in [Tabelle 1.3](#) helfen dem Programmierer bei der Gestaltung.

Die verschiedenen Datentypen werden auf dem Frontpanel als Controls und Indicators dargestellt ([Tabelle 1.4](#)).


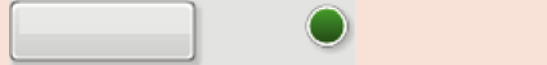

**Tabelle 1.3** Funktionen zur Darstellung und Ausrichtung von Frontpanel-Elementen

	<p><i>Align</i> – Ausrichtung der Elemente: Mit diesem Tool können selektierte Elemente sowohl horizontal (linker Rand, Mitte, rechter Rand) als auch vertikal (oberer Rand, Mitte, unterer Rand) zueinander ausgerichtet werden.</p>
	<p><i>Distribute</i> – Verteilen der Elemente: Mit diesem Tool können selektierte Elemente gleichmäßig zueinander verteilt werden, z. B. in gleiche Abstände zwischen mehreren Elementen (Selektion).</p>

**Tabelle 1.3** Funktionen zur Darstellung und Ausrichtung von Frontpanel-Elementen (Fortsetzung)

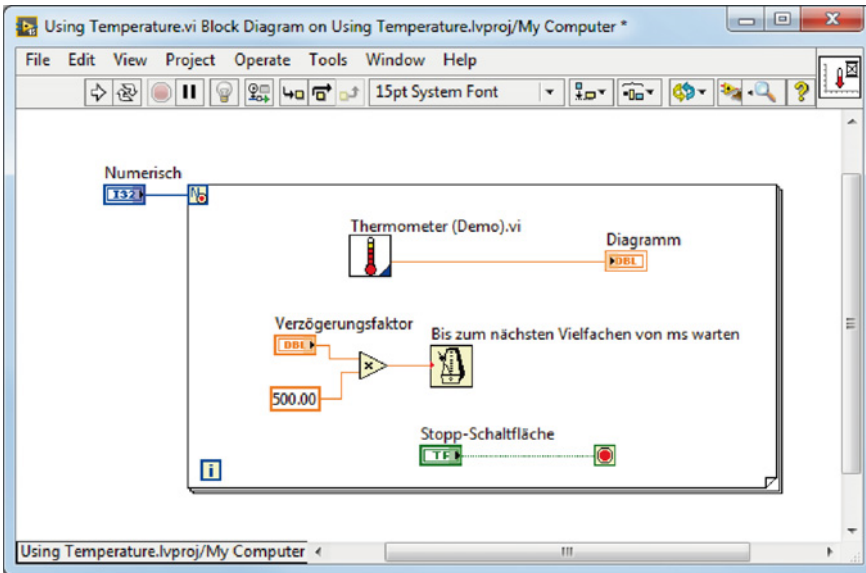
	<p><b>Resize</b> – Größenanpassung von Elementen: Mit diesem Tool können selektierte Elemente auf die gleichen Dimensionen getrimmt werden.</p>
	<p><b>Reorder</b> – Gruppieren und Ebenen festlegen: Mit diesem Tool können selektierte Elemente gruppiert und gelockt werden. Ebenfalls kann bei übereinanderliegenden Elementen die Reihenfolge bestimmt und Elemente in den Vorder- bzw. Hintergrund verschoben werden.</p>

**Tabelle 1.4** Controls und Indicators der gebräuchlichsten Datentypen in LabVIEW

<p><b>Numeric Control</b>      <b>Numeric Indicator</b></p> 	<p>Mit einem <i>numerischen Datentyp</i> können Zahlen verschiedenen Typs dargestellt werden, z. B. Gleitkommazahlen, ganzzahlige Typen etc. Numerische Controls sowie Indicators gehören zu den am häufigsten eingesetzten Elementen.</p>
<p><b>Blank Button</b>      <b>LED</b></p> 	<p>Mit dem <i>booleschen Datentyp</i> lassen sich die Zustände TRUE und FALSE darstellen. Andere Elemente außer Buttons sind z. B. Slide Switches oder Radio Buttons.</p>
<p><b>String Control</b>      <b>String Indicator</b></p> 	<p><i>Strings</i> sind zusammengesetzte Ketten aus ASCII-Zeichen. Häufig werden Controls als Eingabeelemente für Benutzertext verwendet, Indicators z. B. zur Anzeige einer Statusmeldung oder anderer Messages.</p>

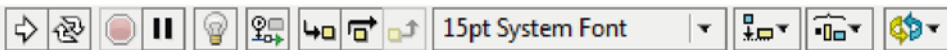
## 1.3.2 Blockdiagramm

Hinter dem Blockdiagramm versteckt sich die eigentliche Funktionalität des VIs (Bild 1.7). Im Gegensatz zum Frontpanel ist das Blockdiagramm für den Benutzer zur Laufzeit nicht sichtbar. Zu typischen Objekten des Blockdiagramms zählen Anschlüsse, SubVIs, Funktionen, Konstanten, Strukturen und Wires (Verbindungen), über welche die Daten zwischen den Objekten des Blockdiagramms übertragen werden.






**Bild 1.7** Blockdiagramm eines VIs

Die Symbolleiste des Blockdiagramms (Bild 1.8) enthält zum Teil dieselben Funktionen wie diejenige des Frontpanels, ist aber zusätzlich mit Debug- und Analysetools ausgerüstet (Tabelle 1.5).



**Bild 1.8** Symbolleiste des Blockdiagramms

**Tabelle 1.5** Debugging-Funktionen des Blockdiagramms

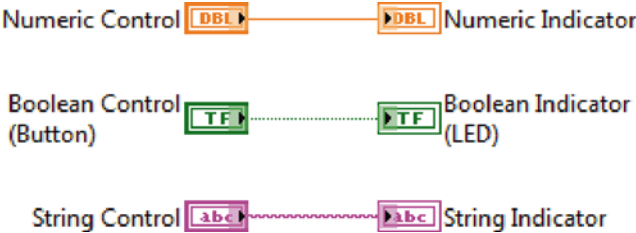
	<i>Highlight-Modus:</i> Ausführung des Programms wird extrem verlangsamt, sodass alle Werte auf den Wires sichtbar werden
	<i>Verbindungswerte speichern:</i> Wenn eingeschaltet, behalten die Wires ihre Werte, auch wenn das Signal das Wire schon passiert hat
	<i>Step into, step over, step out:</i> Programmausführung im Einzelschritt-Modus zur Kontrolle aller Signale



Das *Blockdiagramm* enthält den Programmcode und die Funktionalität eines VIs.

Die Controls (Bedienelemente) und Indicators (Anzeigeelemente), welche bereits im Frontpanel eingesetzt wurden, müssen im Blockdiagramm natürlich funktional eingebunden werden.

In [Bild 1.9](#) sind die gleichen Controls und Indicators abgebildet, welche im Frontpanel schon in [Tabelle 1.4](#) aufgeführt wurden. Grundsätzlich unterscheiden sich Controls und Indicators im Blockdiagramm aufgrund der Position des schwarzen Anschlusspfeils (wo eine Verbindung erfolgen muss) sowie der Breite des Rahmens. Controls sind immer Datenquellen, weshalb der Ausgangsanschluss rechts liegt, Indicators sind die Verbraucher der Daten, weshalb der Eingang stets links liegt. Die Farbe des Elements, egal ob Control oder Indicator, entspricht der Farbe des enthaltenen Datentyps.



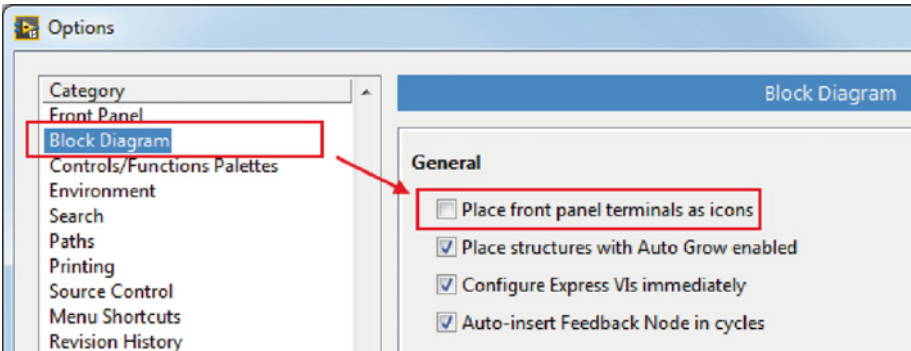
**Bild 1.9** Controls und Indicators im Blockdiagramm

Die Elemente können im Blockdiagramm auch als Icons dargestellt werden ([Bild 1.10](#)). Eventuell ist dies bei der Installation auch so voreingestellt. Da diese Ansicht jedoch keinen Vorteil bringt und erst noch mehr Platz benötigt als die normalen Symbole, sollten die Anschlüsse nie als Icons angezeigt werden.



**Bild 1.10** Darstellung der Elemente als Icons

Um einzelne Controls oder Indicators zu ändern, muss mit Rechtsklick auf das Element und unter *View As Icon* der Haken entfernt werden. Um die Einstellung für alle zukünftig platzierten Elemente zu setzen, muss unter *Tools* → *Options* in der Kategorie *Block Diagram* der entsprechende Haken entfernt werden ([Bild 1.11](#)).

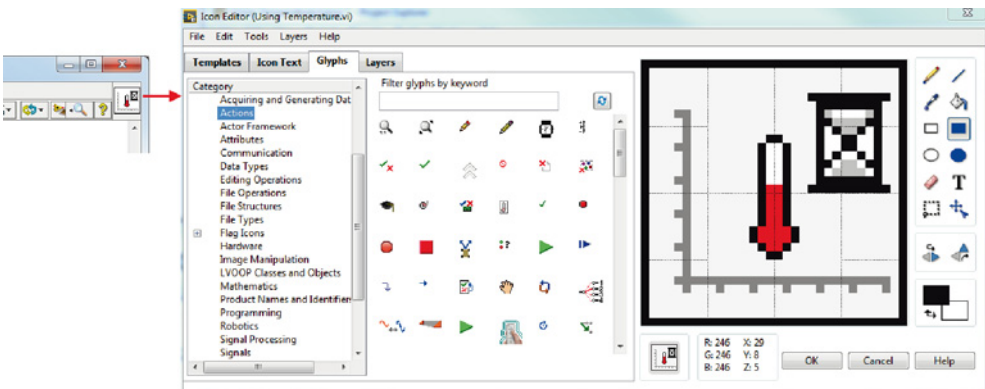


**Bild 1.11** Options Dialog

### 1.3.3 Symbol und Anschlussblock

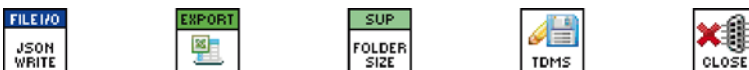
Jedes VI verfügt über ein Symbol und einen Anschlussblock. Wird ein VI im Programmcode (Blockdiagramm) eines anderen VIs verwendet, ist es wichtig, dass es auf den ersten Blick identifizierbar ist und der Programmierer anhand des Symbols erkennt, welche Funktion das VI übernimmt. In [Bild 1.7](#) befindet sich ein VI mit dem Namen Thermometer (Demo).vi im Programmcode des Haupt-VIs. Anhand des Symbols ist klar, dass sich hinter dem VI eine Temperaturapplikation verbergen muss.

Das Symbol kann einfach über einen Doppelklick auf das VI-Symbol in der rechten oberen Ecke des Frontpanels oder des Blockdiagramms editiert werden. Darauf öffnet sich der Icon-Editor ([Bild 1.12](#)), wo Text und vordefinierte Icons im  $32 \times 32$  Pixel großen Symbolfeld platziert werden können.



**Bild 1.12** VI Icon-Editor

Das Symbol dient der einfachen Erkennung des VIs, wenn es innerhalb von anderen VIs verwendet wird. Dabei muss nicht zwingend eine Grafik als Symbol verwendet werden, es reicht auch eine kryptische Beschreibung oder eine Kombination von Symbol und Text. In [Bild 1.13](#) sind Beispiele aussagekräftiger VI-Symbole dargestellt.



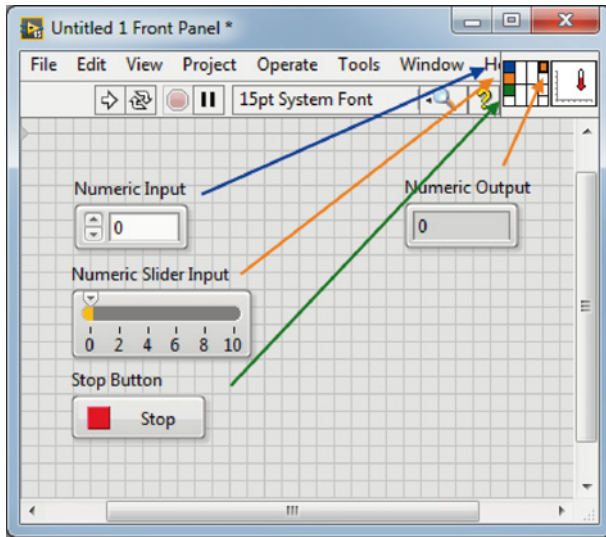
**Bild 1.13** Beispiele von aussagekräftigen VI-Symbolen



Jedes VI muss mit einem individuellen Symbol ausgestattet sein.

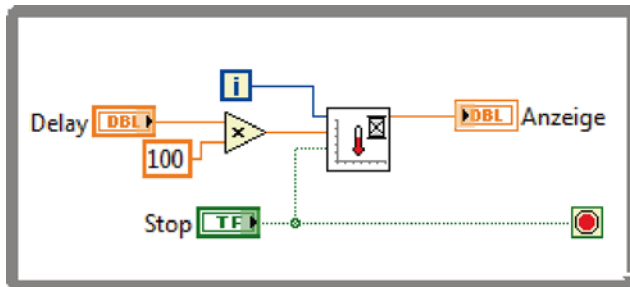
Der Anschlussblock eines VIs ist die Schnittstelle zu einem übergeordneten VI. Er definiert, an welchen Ein-/Ausgängen welche Daten dem VI übergeben bzw. vom VI zurückgegeben werden. Werden die einzelnen Felder des Anschlussblocks mit Frontpanel-Elementen verbunden, nehmen sie automatisch die Farbe des entsprechenden Datentyps an. Hierzu kann mit der Drahtrolle (automatisches Tool) auf ein leeres Feld des Anschlussblocks geklickt werden und danach auf das Frontpanel-Element, welches mit dem Feld verbunden werden soll ([Bild 1.14](#)).





**Bild 1.14** Anschlussblock eines VIs

Für die Verwendung in einem übergeordneten VI muss der Anschlussblock definiert sein, um Werte übergeben und übernehmen zu können.



**Bild 1.15** Verwendung eines VIs mit Ein- und Ausgängen

Für den Anschlussblock stehen verschiedene Muster mit einer kleineren bis größeren Zahl Anschlüsse zur Verfügung. Im Anschlussblock mit einem Rechtsklick auf *Patterns* lässt sich das Muster verändern (Bild 1.16).

In LabVIEW wird nach dem Datenflussprinzip programmiert (Kapitel 3). Das heißt, die Verarbeitung verläuft von links nach rechts. Deshalb ist es wichtig, diesem Prinzip auch bei der Belegung von Anschlussblöcken Rechnung zu tragen. Eingänge werden grundsätzlich auf der linken Seite verbunden, Ausgänge rechts, wie in Bild 1.17 dargestellt. Etwaige zusätzliche Anschlussfelder oben und unten werden erst verwendet, wenn die seitlichen Anschlüsse nicht mehr ausreichen.

# Index

1D-Array [111](#), [113](#)  
2D-Array [112](#), [113](#), [141](#)

## A

Abbruchbedingung [89](#)  
Abort Execution [71](#)  
Add Case for every value [213](#)  
Anschlussblock [13](#), [19](#), [21](#), [79](#)  
Anschlussblock eines VIs [20](#)  
Anzeigeelemente (→ Indicators) [14](#), [16](#), [18](#),  
[60](#), [63](#)  
Array [64](#), [93](#), [111–114](#), [143](#)  
Array-Funktionen [114](#)  
Attribute [129](#)  
Aufschlüsseln (→ Unbundle) [119](#)  
Autopopulating [30](#)

## B

Bedienelemente (→ Controls) [14](#), [16](#), [18](#), [126](#)  
Benutzeroberfläche [167](#)  
Blockdiagramm [12](#), [13](#), [17](#), [18](#), [25](#), [58](#), [63](#), [73](#)  
Boolean [57](#), [101](#)  
boolesche Controls [58](#)  
boolescher Datentyp [16](#)  
Breakpoints [72](#)  
Build Path [152](#)  
Build-Spezifikation [34](#)  
Bundle (Bündeln) [119](#)  
Bundle by Name [120](#)

## C

Calling VIs [75](#)  
Case-Struktur [101](#), [216](#)  
cDAQ [186](#)  
Channels [153](#)  
Chart History [139](#)  
Chart Properties [136](#), [137](#)

Cluster [64](#), [117](#)  
Conditional Terminal [90](#)  
Config File [158](#)  
Container [111](#)  
Controls (→ Bedienelemente) [14](#), [16](#), [18](#), [126](#)  
cRIO [186](#)

## D

DAQmx [187](#)  
Datalogging [147](#)  
Dateipfad [60](#)  
Datenaustausch [200](#)  
Datenerfassung mit NI-Hardware [185](#)  
Datenfluss [20](#), [49–51](#)  
Datentypen [51](#)  
Debugging-Funktionen [17](#), [69](#)  
Debugging-Methoden [70](#)  
dequeue [207](#)  
Design Patterns (→ Entwurfsmuster) [211](#)  
Disabled [171](#)  
Disabled and Grayed Out [171](#)  
Dokumentation [82](#), [83](#)

## E

Eigenschaften (→ Properties) [136](#), [156](#)  
Einbettung von SubVIs [80](#)  
enqueue [207](#)  
Entscheidungsstrukturen [101](#)  
Entwurfsmuster (→ Design Patterns) [211](#)  
Enum [62](#), [102](#), [125](#), [216](#)  
Ereignisgesteuerte Programmierung [103](#)  
Error-Cluster [102](#), [121](#)  
Error-Code [122](#)  
Error-Handler [123](#)  
Error-Leitung [89](#)  
Event [104](#)  
Eventstruktur [103](#), [104](#), [106](#), [199](#)  
Explizite Property Nodes [170](#)

**F**

Fehlerliste 70  
File I/O 147  
Filter-Event 106  
Fixed-Point-Zahlen 53  
Fixkommazahlen 51  
Fließkommazahlen 51, 53  
For-Loop 90, 91, 94  
Free Text 84  
Frontpanel 13–16, 23, 24, 54, 57, 58, 60, 62, 63  
Funktionen des Error-Handlings 123

**G**

Ganzzahlige Datentypen 53  
Generic 168  
GObject 168  
Groups 153  
Guided User Interface (GUI) 167

**H**

Highlight Execution 71

**I**

Indexing Tunnel 93, 94  
Indicators (→ Anzeigeelemente) 14, 16, 18, 60, 63  
Integer 51, 52, 101  
Invoke Nodes (→ Methodenknoten) 172  
Iterationszähler 89

**K**

Keys 158  
Klasse 167  
Komplexe Zahlen 51, 53  
Konfigurationsfiles 157  
Kontexthilfe 21, 82, 115, 129, 151, 152  
Kontexthilfe im Blockdiagramm 22  
Kontextmenü 118

**L**

Logging 124, 149  
Loops (→ Schleifen) 89

**M**

Measurement & Automation Explorer (MAX) 186, 187  
Melder-Event 106  
Messdaten 185  
Methodenknoten (→ Invoke Nodes) 172  
Modularität 75

**N**

NI-DAQmx 186  
NI-USB 185  
Notifier 203, 204  
numerische Datentypen 16, 51–55

**O**

Occurrence 200, 201  
Open Type Def 125

**P**

Paletten 22, 23  
Path (→ Pfad) 60, 152  
Pause 71  
PCI/PCI-E 185  
Pfad (→ Path) 60, 152  
physical channels 190  
Plot 137  
Polymorphie 115, 116  
Probes 72, 73  
Programmierstil 51  
Projekt Explorer 28, 29, 31, 126  
Properties (→ Eigenschaften) 136, 156  
Property Nodes 169  
PXI 186

**Q**

Queue 203, 204, 206  
Queued Message Handler (QMH) 211, 216  
Queue-Funktionen 208

**R**

Read Delimited Spreadsheet 150, 151  
Referenz 147, 170

Rendezvous 202, 203  
 Reorder Controls 118  
 Retain Wire Values 71  
 Ring 62  
 Run 69

## S

Schieberegister (→ Shift Register) 95  
 Schleifen (→ Loops) 89  
 Sections 158  
 Semaphore 201  
 Sequenzierung 124  
 Shift Register (→ Schieberegister) 95  
 Signed Integer 52  
 Skalar 115  
 Slider 66  
 Sonde 72  
 Spreadsheets 150  
 State Machine (→ Zustandsmaschine) 211  
 State Machine mit Enum 213  
 State Machine mit String 215  
 Step-Funktionen 71, 72  
 Stepping 71  
 Strict Type Def 126  
 String 16, 58, 102, 216  
 String-Funktionen 59, 60  
 Strip Path 152  
 Strukturen 101  
 strukturierte Daten 111  
 Subdiagram Label 84  
 SubVIs 69, 75  
 Symbol 13, 19  
 Symbolleiste 14, 15, 17  
 Synchronisation 199  
 Synchronisation mit Datenaustausch 203  
 Synchronisation ohne Datenaustausch 200

## T

Task Read 192  
 TDMS Viewer 155  
 TDMS-Funktionen 157  
 Technical Data Management Streaming  
 (TDMS) 153  
 Testpanels 187  
 Timeout 103  
 Timeout-Event 103

Timing 91  
 Tip Strip 83  
 To Variant 128  
 Transitions 211  
 Tunnel 69, 92  
 Typdefinitionen 124  
 Typkonvertierung 54

## U

Überlauf 53  
 Unbundle (→ Aufschlüsseln) 119  
 Unbundle by name 119  
 Unsigned Integer 52  
 Use Default If Unwired 103

## V

Variant 128  
 Virtual Folder 29  
 VIs 11, 13, 14, 17, 19, 50  
 VI-Server 167  
 vorzeichenlose Datentypen 52

## W

Wait 92  
 Wait until next ms multiple 92  
 Wait-Funktionen 92  
 Waveform 138  
 Waveform Chart 135, 138  
 Waveform Graph 135, 140  
 Werkzeuge der Tools Palette 28  
 While-Loop 89, 92, 106, 201  
 Wire 73  
 Wiring 50  
 Write Delimited Spreadsheet 150

## X

XY-Graph 142

## Z

Zählanschluss 90  
 Zustandsdiagramm 212  
 Zustandsmaschine (→ State Machine) 211