

Wolfgang Georgi
Philipp Hohl

Einführung in LabVIEW



6., erweiterte Auflage



HANSER



bleiben Sie auf dem Laufenden!

Hanser Newsletter informieren Sie regelmäßig über neue Bücher und Termine aus den verschiedenen Bereichen der Technik. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter

www.hanser-fachbuch.de/newsletter

Wolfgang Georgi • Philipp Hohl

Einführung in LabVIEW

6., erweiterte Auflage

Mit 1018 Bildern und 163 Aufgaben



Fachbuchverlag Leipzig
im Carl Hanser Verlag

Prof. Dipl.-Math. Wolfgang Georgi

Hochschule Ravensburg-Weingarten für Technik, Wirtschaft und Sozialwesen

M.Eng. Philipp Hohl

Hochschule Ravensburg-Weingarten für Technik, Wirtschaft und Sozialwesen



Alle in diesem Buch enthaltenen Programme, Verfahren und elektronischen Schaltungen wurden nach bestem Wissen erstellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund ist das im vorliegenden Buch enthaltene Programm-Material mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieses Programm-Materials oder Teilen davon entsteht.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN: 978-3-446-44272-6

E-Book-ISBN: 978-3-446-44407-2

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung – mit Ausnahme der in den §§ 53, 54 URG genannten Sonderfälle –, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Fachbuchverlag Leipzig im Carl Hanser Verlag

© 2015 Carl Hanser Verlag München

Internet: <http://www.hanser-fachbuch.de>

Lektorat: Franziska Jacob, M. A.

Herstellung: Dipl.-Ing. (FH) Franziska Kaufmann

Satz: Kösel Media GmbH, Krugzell

Covergestaltung: Marc Müller-Bremer, www.rebranding.de, München

Coverrealisierung: Stephan Rönigk

Druck und Bindung: Pustet, Regensburg

Printed in Germany

Vorwort zur sechsten Auflage

Dieses Lehrbuch führt wie in der ersten Auflage in das Programmieren mit LabVIEW ein. Damals mussten wir noch erklären, dass sich LabVIEW für messtechnische Anwendungen eignet und in der Industrie mehr und mehr geschätzt wird. Heute ist das allgemein bekannt, auch, dass es sich bei dieser von der Firma National Instruments entwickelten Software um ein Werkzeug handelt, das sich weit über die Messtechnik hinaus vorteilhaft anwenden lässt.

Um einen guten Lernerfolg zu erzielen, sollte der Leser möglichst viele Beispiele und Übungen am PC durcharbeiten. Alle Beispielprogramme wurden für die LabVIEW-Version 2014 geschrieben. Wir setzen also voraus, dass der Leser die Version 2014 von LabVIEW installiert hat. Frühere Versionen wie LabVIEW 2009 oder LabVIEW 8.0 sind im Kern recht ähnlich. Programme, die mit diesen Versionen erstellt werden, laufen auch unter der Version 2014. Doch trifft das Umgekehrte naturgemäß nicht zu, weil jede neue Version auch neue Möglichkeiten bietet. Weiter wird vorausgesetzt, dass der PC unter einem der Betriebssysteme Windows 7 oder Linux arbeitet.

Das Buch wendet sich an Studierende, aber auch an Ingenieure, die unter dem Stichwort "Lebenslanges Lernen" versuchen, neueren Trends in der Industrie zu folgen.

Das Lehrbuch gliedert sich in vier Teile:

Teil I: Grundlagen des Programmierens in LabVIEW

Teil II: Technische Anwendungen

Teil III: Kommunikation

Teil IV: Fortgeschrittene Techniken

In Teil I werden Installation und Aufruf von LabVIEW, grundlegende Arbeitsmittel wie (Front-)Panel, Diagramm, Paletten für Eingabe/Ausgabe, Funktionen und Werkzeuge behandelt, ferner Konzepte von LabVIEW, Datentypen, Grundlagen der Programmierung und Visualisierungstechniken.

Teil II befasst sich mit Anwendungen wie Fouriertransformation, Filterung, Lösen von Differenzialgleichungen und Differenzialgleichungssystemen in der Technik.

Teil III geht auf die Kommunikation ein. Hier sind zwei Aspekte von Bedeutung:

- Externe Kommunikation mit anderen Geräten und Rechnern, z.B. über USB, Datenerfassungskarten, TCP/IP (Internetanbindung),
- Kommunikation mit anderen Softwarepaketen, z.B. mit der Erstellung und Anbindung selbst geschriebener C-Module.

Teil IV befasst sich mit Zustandsautomaten, mit der objektorientierten Programmierung (OOP), mit Tabellenkalkulation (Excel) und Datenbankanwendungen (Access), mit dem

Datenaustausch über Intra- und Internet, mit dem Compact RIO-System von National Instruments samt FPGA-Programmierung und Verwendung von XControls und XNodes.

Neu hinzugekommen sind in der sechsten Auflage Kapitel 23 und 24, in denen wir Scripting und das bisher recht unzugängliche Konstrukt der XNodes behandeln.

Weitgehend neu haben wir das Kapitel 20 gestaltet. Abschnitt 20.5 wurde ins Internet (<http://www.geho-labview.de>) gestellt und ersetzt durch das seit LabVIEW 2013 neu eingeführte Konzept des Webdienstes.

Mehr als in der fünften Auflage haben wir verschiedene Kapitel ergänzt mit dem Ziel, dem Leser anhand von Beispielen 'guten Programmierstil' zu vermitteln.

Das Lehrbuch ist trotz seines dadurch stark gewachsenen Umfangs immer noch knapp gehalten. Es kann also nur eine Einführung sein, die allerdings versucht, die wichtigsten Aspekte von LabVIEW zu berücksichtigen. Bei einem so umfangreichen Softwaresystem wie LabVIEW sind jedoch Lücken unvermeidlich. Hier verweisen wir auf weiterführende Literatur, auf die Veröffentlichungen von National Instruments, auf User Groups und auf das Internet ganz allgemein. Diese Hinweise werden wir in den verschiedenen Kapiteln des Lehrbuchs noch vertiefen.

Wir bedanken uns ganz herzlich bei allen, die uns geholfen haben:

Besonders bei Herrn Thakur Adhikari, der sich in seiner Masterthese mit XNodes befasst und damit regelungstechnische Anwendungen entwickelt hat. Ohne seine gründlichen Recherchen zu den in der Literatur nur unzulänglich dokumentierten XNodes wäre es nicht möglich gewesen, Kapitel 24 noch in dieser Auflage herauszugeben.

Schließlich danken wir allen Lesern, die mit ihren Fragen Verständnisprobleme deutlich gemacht und uns damit zur Verbesserung mancher Erklärung angeregt haben. Wir sind auch weiterhin für Anregungen und Kritik dankbar. Diese können Sie uns jetzt auch direkt über die Internetseite <http://www.geho-labview.de> übermitteln.

Weitere Informationen zu LabVIEW sowie die Downloads der Test- bzw. Studentenversionen finden Sie unter: www.ni.com/download-labview/d/

Dem Fachbuchverlag Leipzig, und hier besonders Frau Werner, Frau Jacob und Frau Kaufmann, danken wir für die gründliche Korrektur und ihre Ratschläge zur Gestaltung des Layouts.

Weingarten, Juni 2015
Wolfegg

W. Georgi
P. Hohl

Inhalt

Teil I: Grundlagen des Programmierens in LabVIEW	17
1 Was ist LabVIEW?	19
1.1 Entwicklungsstufen	19
1.2 Was will dieses Lehrbuch?.....	21
1.3 Installation.....	21
1.4 Einführendes Beispiel.....	21
1.4.1 Programmierung von $c = a + b$	25
1.4.2 Speicherung als Programm Add.vi	28
1.4.3 Starten und Stoppen von Add.vi.....	28
1.4.4 Fehlersuche in Add.vi (Debugging).....	29
1.5 Beispiel für eine Grafik in LabVIEW	30
1.6 Grundlegende Konzepte von LabVIEW	31
1.6.1 Frontpanel.....	31
1.6.2 Blockdiagramm.....	31
1.7 Rezepte.....	32
1.8 Shortcuts.....	33
2 Einstellungen, Paletten	35
2.1 Einstellungen	35
2.1.1 Einstellungen von LabVIEW.....	35
2.1.2 Frontpanel.....	36
2.1.3 Blockdiagramm.....	37
2.1.4 Ausrichtungsgitter	39
2.1.5 Wiederherstellungen	39
2.2 Paletten	39
2.2.1 Werkzeugpalette (Tools Palette).....	40
2.2.2 Eingabe-/Ausgabe-Elemente.....	41
2.2.3 Funktionenpalette.....	44
2.2.4 Palette konfigurieren	47
3 Programmstrukturen	48
3.1 Strukturiertes Programmieren.....	48
3.2 Sequenz	50
3.3 Case-Struktur	53

3.4	Schleifen	57
3.5	Guter Programmierstil.....	61
4	Datentypen	63
4.1	Numerische Datentypen	63
4.1.1	Kontextmenü: 'Darstellung'	63
4.1.2	Kontextmenü: 'Anzeigeformat...'	64
4.2	Boolesche Datentypen	66
4.3	String und Pfad	68
4.4	Arrays	71
4.4.1	Definition und Initialisierung eines 1-dimensionalen Arrays	71
4.4.2	Definition und Initialisierung eines 2-dimensionalen Arrays	73
4.4.3	Array erstellen.....	74
4.4.4	Rechnen mit Arrays: Addition	75
4.4.5	Rechnen mit Arrays: Multiplikation.....	76
4.4.6	Steuerung von For-Schleifen mit Arrays	77
4.4.7	Behandlung einzelner Arrayelemente	79
4.5	Cluster	81
4.5.1	Erzeugung eines Clusters	82
4.5.2	Clusterwerte ändern	83
4.5.3	Aufschlüsseln eines Clusters	85
4.5.4	Umordnen der Elemente eines Clusters	86
4.5.5	Cluster-Arrays	87
4.6	Ring & Enum.....	88
4.7	Datentyp FXP	90
4.8	Datentyp Variant	92
4.9	Guter Programmierstil.....	94
5	Unterprogramme und Typdefinitionen	96
5.1	Wozu Unterprogramme (SubVIs)?	96
5.2	Erstellen von Unterprogrammen	97
5.2.1	Einführendes Beispiel	97
5.2.2	Weitere Hinweise für die Erstellung eines Unterprogramms	100
5.2.3	Einstellungen für Programme und Unterprogramme	102
5.2.4	Erstellen von Unterprogrammen mit internem Zustand	104
5.2.5	Erstellen von polymorphen Unterprogrammen.....	105
5.3	Aufruf von Unterprogrammen	108
5.3.1	Statische Bindung.....	108
5.3.2	Dynamische Bindung	109
5.3.2.1	VI-Referenz öffnen und schließen.....	109
5.3.2.2	Aufruf eines VI über seine Referenz	110
5.3.2.3	Beispiel für den SubVI-Austausch während der Laufzeit.....	112
5.3.2.4	Rekursiver Aufruf von Unterprogrammen	112
5.3.2.5	Testen (Debugging) von ablaufinvarianten SubVIs	113

5.4	Typdefinitionen	115
5.4.1	Beispiel einer Typdefinition für Enum-Variablen.....	115
5.4.2	Beispiel einer Typdefinition für Registerkarten	117
5.5	Guter Programmierstil	118
5.5.1	Vereinfachung durch Unterprogramme und Typdefinitionen	118
5.5.2	Aussagekräftige Symbole (Icons).....	120
5.5.3	Anordnung häufig verwendeter Elemente.....	120
5.5.4	Kommentierung der Elemente und Funktionen eines VI	120
5.5.5	Detaillierte Hilfe.....	122
6	Prozessvisualisierung	123
6.1	OOP-Konzepte	123
6.2	Eigenschafts- und Methodenknotten	123
6.3	Grafische Ausgabe	127
6.3.1	Chart (Signalverlaufdiagramm).....	127
6.3.1.1	Darstellung einer Sinuskurve	127
6.3.1.2	Darstellung von zwei oder mehr Kurven in einem Chart	129
6.3.1.3	Legende zu einem Chart oder Graphen	130
6.3.1.4	Skalierung der Ordinate in einem Chart.....	131
6.3.2	Graph (Signalverlaufgraph).....	133
6.3.2.1	Darstellung einer Sinuskurve	133
6.3.2.2	Darstellung von zwei oder mehr Kurven in einem Graphen	134
6.3.2.3	Skalierung der Abszisse in einem Graphen	136
6.3.3	XY-Graph.....	138
6.3.3.1	Darstellung einer Relation im XY-Graphen	139
6.3.3.2	Darstellung mehrerer Relationen in einem XY-Graphen	140
6.3.4	Signalverlauf	141
6.4	Express-VIs, Programmierstil	146
6.4.1	Express-VI zur Erzeugung von Kurven	146
6.4.2	Express-VI zur Erstellung von Berichten.....	147
7	Referenzen, Fehlerfunktionen	149
7.1	Einführendes Beispiel.....	149
7.1.1	Vertauschung von zwei Variablenwerten	149
7.1.2	Referenzen auf Bedien- und Anzeigeelemente	150
7.1.3	Lösung des Vertauschungsproblems	151
7.2	Vererbung.....	152
7.2.1	Eigenschaften der Basisklasse	154
7.2.2	Eigenschaften von abgeleiteten Klassen.....	154
7.3	Fehlerfunktionen	156
7.3.1	Fehlermeldungen mit oder ohne Dialog.....	156
7.3.2	Wo findet man wichtige Fehlerelemente und Fehlerfunktionen?	157
7.3.3	Verschiedene Fehlerarten.....	158
7.3.3.1	Standardfehlerleitung.....	158
7.3.3.2	Funktionen ohne oder mit vereinfachter Fehlerleitung	158

7.3.4	Ausgang aus While-Schleifen	160
7.3.5	Erzwingung von sequenziellem Ablauf	161
8	Datentransfer von und zur Festplatte	162
8.1	Dateifunktionen	162
8.1.1	Allgemeines zur Speicherung von Dateien.....	162
8.1.2	Palette Dateifunktionen	164
8.1.3	Einführendes Beispiel	165
8.1.4	Modifiziertes Beispiel	166
8.1.5	Beispiel: Anlegen einer Protokolldatei	167
8.1.6	Überschreiben ohne Warnung	167
8.2	Pfade	168
8.2.1	Pfadkonstanten	168
8.2.2	Pfadkonstante 'Standardverzeichnis'.....	169
8.2.3	'Standardverzeichnis' ändern	170
8.2.4	'Standarddatenverzeichnis' ändern	171
8.2.5	Lesen und Schreiben anderer Datentypen	171
8.2.6	Verketteten von Schreib- und Lesefunktionen	172
8.2.7	Tabellenkalkulation.....	173
8.3	Pfade in einer EXE-Datei	173
8.4	Fortgeschrittene Dateitypen	175
8.4.1	LVM- , TDMS- und TDM-Dateien	176
8.4.2	Diadem.....	179
8.4.3	ZIP-Dateien	179
8.4.4	Konfigurationsdateien	181
9	LabVIEW-Kurzüberblick	185
9.1	Aufbau des LabVIEW-Systems	185
9.1.1	Programmierung in G	185
9.1.1.1	Interpretieren oder kompilieren?	185
9.1.1.2	Datenflussprogrammierung.....	187
9.1.2	Hardware-Unterstützung	187
9.1.3	Bibliotheken mathematischer und technischer Funktionen	188
9.1.4	Benutzerschnittstelle	189
9.1.5	Technologische Abstraktion	190
9.1.6	Rechenmodelle	190
9.2	Projekte	190
9.3	Erstellung von EXE-Dateien.....	192
9.3.1	Erstellung einer EXE-Datei	192
9.3.2	EXE-Datei auf einem Rechner ohne LabVIEW-System.....	194
9.4	Strukturen zur Programmentwicklung.....	197
9.4.1	Deaktivierungsstrukturen.....	197
9.4.2	Debug-Einstellung in der Projektverwaltung	199
9.5	LabVIEW-Bibliotheken.....	200
9.6	Umwandeln von LLB-Bibliotheken	202

9.7	Einbindung von C-Funktionen unter Windows	204
9.7.1	Reihe in C#	205
9.7.2	Reihe in C++	209
9.7.3	Reihe mit MathScript	213
9.8	Hilfen zu LabVIEW	213
9.9	Schnelleinfügeleiste (Quickdrop)	215
9.10	Der VI Package Manager	217
9.10.1	Verwalten der LabVIEW-Entwicklungsumgebung	218
9.10.2	Eigenes Paket erstellen	220

Teil II: Technische Anwendungen **221**

10 Fouriertransformation **222**

10.1	Zeit- und Frequenzbereich	222
10.1.1	Die reelle Fouriertransformation	223
10.1.2	Darstellung der Fourierkoeffizienten \underline{c}_k in LabVIEW	226
10.2	Diskrete Fouriertransformation	229
10.2.1	Satz von Shannon	229
10.2.2	Aliasing	231
10.2.3	Frequenzauflösung	232

11 Filterung **234**

11.1	Filtertypen	234
11.1.1	Ideale und reale Filter	234
11.1.2	Beispiel eines digitalen Filters	235
11.2	LabVIEW-Filterfunktionen	238
11.3	Filterung im Frequenzbereich	240
11.3.1	Idee der Filterung im Frequenzbereich	240
11.3.2	Die inverse Fouriertransformation in LabVIEW	240
11.3.3	Beispiel eines Tiefpasses	241

12 Differenzialgleichungen **243**

12.1	Lösen mit LabVIEW-ODE-Funktionen	243
12.2	Lösen nach dem Analogrechnerprinzip	245
12.2.1	Blockdiagramm-Darstellung	245
12.2.2	Vereinfachungen	248
12.3	Genauigkeit numerischer Verfahren	250

13 Systeme von Differenzialgleichungen **253**

13.1	Systeme gewöhnlicher Differenzialgleichungen	253
13.2	Gekoppeltes Feder-Masse-System	253
13.2.1	Lösung mit eingebauter ODE-Funktion	254
13.2.2	Lösung mit Blockdiagramm wie in MATLAB®	255
13.3	Umwelt und Tourismus	257

14	Parallelverarbeitung, Laufzeiten, Ereignisse	260
14.1	Einführendes Beispiel	260
14.2	Grundbegriffe der Parallelverarbeitung	262
14.2.1	Multiprocessing, Multitasking, Multithreading	262
14.2.2	Synchronisierung von Prozessen	263
14.3	Parallelverarbeitung unter LabVIEW	264
14.3.1	Erzeugen von Ressourcen für die Prozesskommunikation	265
14.3.2	Freigabe von Ressourcen der Prozesskommunikation	267
14.3.3	Zeitbegrenzung Ressource schont Prozessor	268
14.4	Prozess-Synchronisierung ohne Datenaustausch	268
14.4.1	Occurrences	268
14.4.2	Semaphor	269
14.4.3	Rendezvous	271
14.5	Prozess-Synchronisierung mit Datenaustausch	272
14.5.1	Melder-Operationen	272
14.5.2	Queue-Operationen	273
14.6	Globale Variablen	274
14.7	Laufzeitprobleme und ihre Behandlung	275
14.7.1	Laufzeitprobleme bei lokalen Variablen	275
14.7.2	Laufzeitprobleme bei globalen Variablen	278
14.8	Ereignisgesteuerte Programmierung	279
14.8.1	Frontpanel-Ereignisse	279
14.8.2	Wertänderungs-Ereignisse	284
14.8.3	Gefilterte Ereignisse	285
14.9	Zeitschleifen	287
Teil III: Kommunikation		289
15	Serielle Eingabe/Ausgabe	290
15.1	RS-232	290
15.2	Programmierung der RS-232 in LabVIEW	292
15.3	Die USB-Schnittstelle	295
15.4	Feld-Bus, CAN-Bus	299
15.4.1	CAN-Protokoll	299
15.4.2	CAN-Interface	301
15.4.3	CANopen-Protokoll, ZILA-Sensor	302
15.4.4	CAN-Bus mit Laptop und zwei Sensoren	304
15.4.5	XNET-System von National Instruments	304
15.5	Der byte-serielle GPIB-Bus	315
16	Datenerfassungsgeräte	317
16.1	Datenerfassungskarten/Datenerfassungsgeräte	317
16.2	Allgemeines	318
16.2.1	Treiber, MAX (Measurement and Automation Explorer)	318
16.2.2	Physikalische und virtuelle Kanäle, Task	324
16.2.3	Programmierung von Datenerfassungs-VIs, simulierte Geräte	325

16.2.4	Programmierung von VIs zur Analogausgabe	330
16.2.5	Programmierung von VIs zum Digital-I/O	331
16.2.6	Programmierung mit Hilfe des DAQ-Assistenten	331
16.2.7	Programmatische Task-Erstellung	333
16.3	USB-Gerät NI USB-6251	334
16.3.1	Begriffe 'differenziell', 'RSE' und 'NRSE'	334
16.3.2	Zwei Analogsignale mit der NI USB-6521 lesen	336
16.3.3	Triggern mit NI USB-6521	337
16.3.4	Streaming mit NI USB-6521	338
16.4	Ältere Datenerfassungskarten/-geräte	345
16.5	TEDS	345
16.6	IVI-Gerät NI USB-513	349

Teil IV: Fortgeschrittene Techniken **355**

17 Professionelle Programmentwicklung **356**

17.1	Sequenzstruktur	356
17.2	Zustandsautomaten.....	357
17.2.1	Notation für Zustandsautomaten	358
17.2.2	Umsetzung Zustandsdiagramm → LabVIEW-Programm	359
17.2.2.1	Strings für die Zustandsauswahl	360
17.2.2.2	Enum für die Zustandsauswahl.....	362
17.3	Münzautomat	363
17.4	Münzautomat mit Queues und Ereignisstrukturen.....	372
17.5	Programmierhilfen	376
17.5.1	Arbeiten mit vorgefertigten Strukturen (Templates).....	376
17.5.2	Beurteilung Programmeffizienz und geeignete Werkzeuge dazu	376

18 Objektorientierte Programmierung **380**

18.1	Warum objektorientiert?	380
18.2	Erstes Beispiel zur objektorientierten Programmierung	383
18.2.1	Bildung einer Klasse.....	383
18.2.2	Private Eigenschaften der Klasse	384
18.2.3	Methoden der Klasse	385
18.3	Weitere Beispiele zur OOP.....	389
18.3.1	Vererbung	389
18.3.2	Polymorphie.....	393
18.3.3	Modulaustausch	397
18.4	Schutz einer Klassenbibliothek	405

19 LabVIEW: Tabellenkalkulation, Datenbanken **408**

19.1	Schreib-/Lesebefehle zur Tabellenkalkulation	408
19.2	Allgemeines über ActiveX.....	410
19.2.1	ActiveX-Container in LabVIEW	411
19.2.2	ActiveX in LabVIEW zur Steuerung von Anwendungen.....	413

19.3	Beispiele zur Anwendung auf Excel	414
19.3.1	Öffnen und Schließen von Excel.....	415
19.3.2	Sichtbarmachen einer Excel-Tabelle	416
19.3.3	Eintragen von Daten in eine Excel-Tabelle	418
19.3.4	Geschwindigkeit der Datenspeicherung	420
19.3.5	Erstellen von Makros zum Umwandeln einer Tabelle in eine Grafik	421
19.3.6	Aufruf von Makros in LabVIEW mit Hilfe von ActiveX.....	424
19.3.7	Erhöhung der Geschwindigkeit	425
19.3.8	Schreiben mehrerer Dateien	428
19.4	Microsoft-Datenbank Access	432
19.4.1	Einführung.....	432
19.4.2	Verbindung mit der Datenbank	433
19.4.3	SQL	435
19.4.4	Verwendung von SubVIs	436
20	Internet, Server und Client	437
20.1	Allgemeine Bemerkungen zum Internet	437
20.1.1	Ethernet.....	437
20.1.2	Ethernet-Karten, MAC- und IP-Adresse	438
20.1.3	TCP/IP-Protokoll	438
20.2	Einfaches LabVIEW-Beispiel: Ping.....	439
20.3	Programmieren mit DataSocket.....	441
20.4	Programmieren mit TCP/IP	443
20.4.1	Server und Client	443
20.4.2	Beispiel für die Übertragung von Sinusdaten über TCP/IP.....	444
20.5	Webdienste	447
20.5.1	Grundbegriffe	447
20.5.2	Struktur der Webdienstkommunikation	448
20.5.3	Erstes einfaches Beispiel.....	448
20.5.4	Zweites einfaches Beispiel	453
20.5.5	Drittes Beispiel	456
20.5.6	Dreiecksberechnung	457
20.5.7	Webserver im Internet	462
20.5.7.1	Firmeninternes Netz.....	462
20.5.7.2	Aufruf im Internet	463
21	Compact RIO-System und FPGA	464
21.1	Definition	464
21.2	Installation.....	466
	Schritt 1: Software-Installation auf dem PC.....	466
	Schritt 2: Zusammenstellen der cRIO-Hardware	467
	Schritt 3: Zuweisung einer IP-Adresse zum cRIO-System.....	467
	Schritt 4: Installation weiterer Software auf dem cRIO-System	470
	Schritt 5: Verbindung eines PC mit einem cRIO-System im Netz	470

21.3	Programmierbeispiele für FPGA	471
21.3.1	Beispiel zur Digitalausgabe	472
21.3.2	Beispiel eines Zählers	477
21.3.3	FPGA-Anwendung: Ermittlung eines Frequenzganges	479
21.3.4	Umgebungsvariablen	489
21.3.4.1	Projekt 'Shared_Einzeln'.....	490
21.3.4.2	Projekt 'Shared_Netzwerk'	492
21.3.4.3	Projekt 'Shared_cRIO'	495
21.3.5	FPGA-Anwendungen auf dem cRIO-9014 ohne PC-Unterstützung.....	497
21.3.5.1	Projekt 'RIO_MOD1_Switch'	497
21.3.5.2	Projekt 'RIO_User1_Switch'	499
21.3.5.3	Umstellung des cRIO-Systems von einem Standalone-Projekt zum nächsten	502
22	XControls	504
22.1	Unterschied zu einfachen Ctls	504
22.2	Anzeige der Flugbahn eines Steines.....	504
22.3	Erstellen eines XControls	506
22.3.1	Allgemeines Rezept	506
22.3.2	Beispiel XControl_Pfeil.xctl	508
22.3.3	Eigenschaften in einem XControl	514
22.3.4	Bedeutung der Rahmen [1] bis [4] im Fassaden-VI	517
22.3.5	Weitere Verbesserungen	523
22.4	XControl zur Erstellung von Symbolleisten.....	525
22.4.1	Zustand der Symbolleiste	526
22.4.2	Funktionen der Symbolleiste	527
22.4.2.1	Symbole hinzufügen	527
22.4.2.2	Alle Symbole löschen	528
22.4.2.3	Rückmeldung des Symbols, das unter dem Mauszeiger liegt	528
22.4.2.4	Anpassung des Erscheinungsbilds an eigene Bedürfnisse	532
22.4.3	Leistungsmerkmal 'Status für Speichern umwandeln'	532
23	LabVIEW VI-Skripte	534
23.1	Was sind VI-Skripte	534
23.2	Die VI-Skripte Funktionen in der Palette anzeigen	534
23.3	Die VI-Skripte Funktionen	536
23.3.1	Neues VI.....	536
23.3.2	Neues VI-Objekt	537
23.3.3	VI-Objektreferenz öffnen	539
23.3.4	Abstand des neuen VI-Objekts von Referenzobjekt	539
23.3.5	GObjects suchen	541
23.3.6	GObject-Beschriftung abfragen	542
23.3.7	Klassenhierarchie mittels Klassennamen ermitteln	542
23.3.8	Weiterführende Informationen	544
23.4	Wo werden VI-Skripte eingesetzt?	545

23.5	Modifizierung der Projektvorlage "Leeres VI"	546
23.6	Erstellen eines Quickdrop Plugins mit VI-Skripting.....	550
24	XNodes	553
24.1	Einführung	553
24.2	Regelungstechnische Anwendung	554
24.3	Aufbau eines XNodes	556
24.4	Wie bildet man einen XNode?.....	556
24.4.1	Vorbereitende Überlegungen	556
24.4.2	Programmierung von NeueKuh.xnode.....	561
24.4.2.1	Template-VI	561
24.4.2.2	Ability-VIs	562
24.5	Wie ändert man einen XNode?	571
24.6	XNodes in der Funktionspalette speichern.....	571
	Literatur	575
	Index	577

Teil I: Grundlagen des Programmierens in LabVIEW

Systematische Einführung in die wichtigsten Konzepte von LabVIEW. Das umfasst:

Installation und Aufruf von LabVIEW, grundlegende Arbeitsmittel wie Frontpanel, Diagramm, Paletten für Eingabe/Ausgabe und ihre Anpassung an Benutzerwünsche, ferner Bedienelemente und Funktionen, Datentypen, das Erstellen von Unterprogrammen, Visualisierungstechniken, Umgang mit Referenzen sowie das Schreiben und Lesen von Daten auf bzw. von der Festplatte. Kapitel 9 gibt eine Kurzübersicht über den Aufbau von LabVIEW und zusätzliche Hilfen zum Erlernen dieser grafischen Programmiersprache.

1 Was ist LabVIEW?

Lernziel

Der Leser soll anhand eines sehr einfachen Beispiels einen ersten Eindruck von LabVIEW, von der Idee der Datenflussprogrammierung und den wichtigsten Programmierkonzepten gewinnen. Er kann einfache VIs von Beginn an entwickeln.

1.1 Entwicklungsstufen

Software wurde und wird unter verschiedenen Aspekten geschaffen. Bekannt sind Begriffe wie 'strukturierte Programmierung', 'objektorientierte Programmierung' usw. In jüngerer Zeit spielt auch die **Prozessvisualisierung** eine zunehmende Rolle.

Ursache ist die ständig komplizierter werdende Technik. Sie verlangt bessere Darstellungsmöglichkeiten, damit der Anwender den Überblick nicht verliert. Man beschränkt sich heute bei der Abbildung technischer Prozesse nicht mehr allein auf konventionelle Anzeigeeinstrumente, sondern stellt auch den Prozessablauf selbst auf dem Bildschirm eines Rechners grafisch dar. Es geht um Anschaulichkeit. Die Füllstandsanzeige eines Behälters wird z.B. nicht mehr nur durch ein analoges Manometer auf dem Bildschirm verkörpert, sondern durch die Zeichnung des Kessels selbst, in dem bunt gefärbt die Flüssigkeit auf- und absteigt. So kann auch der Laie erahnen, in welchem Zustand sich ein technischer Prozess gerade befindet. Um die Programmierung solcher grafischen Oberflächen mit anschaulichen, teilweise auch bewegten Bildern zu unterstützen, sind verschiedene Programmierwerkzeuge entwickelt worden.

Eines davon ist das Softwarepaket **LabVIEW** von National Instruments. LabVIEW ist die Abkürzung von Laboratory Virtual Instrument Engineering Workbench. Es ist **Entwicklungsumgebung** und **grafische Programmiersprache** zugleich.

Grafische Hilfsmittel in Papierform wie Programmablaufpläne oder Flussdiagramme gibt es schon seit langem, doch 'Zeichnen am Computer' wurde erst möglich, als die Rechner hinreichend leistungsfähig und Bildschirme als Ausgabe- und Eingabegeräte mit hoher Auflösung verfügbar wurden. Das geschah gegen Ende der 70er-Jahre. Zu der Zeit hatten zwei der Gründer der Messtechnikfirma National Instruments, Jim Truchard und Jeff Kodosky, die Idee, die Software zum Testen ihrer Messgeräte ähnlich wie diese selbst zu strukturieren. Sie nannten einzelne Bausteine deshalb virtuelle Instrumente oder VIs, eine Bezeichnung, die sich bis heute als Datei-Erweiterung eines jeden LabVIEW-Programms erhalten hat. Eine andere Idee bestand darin, die Programmierung nicht, wie bisher üblich, zeilenweise in Form von Anweisungen, genannt 'Statements', niederzuschreiben, sondern Funktionsblöcke in einem Blockdiagramm auf dem Bildschirm darzustellen. Dies gestaltet sich ganz so, wie man das auch früher schon mit Bleistift und Papier gemacht hatte.

Die Entwicklung der ersten lauffähigen LabVIEW-Version war eng geknüpft an das Aufkommen leistungsfähiger Personalcomputer, speziell des Macintosh von Apple. Anfang der 80er-Jahre bot nur dieser PC die grafischen Voraussetzungen zur Realisierung der mit LabVIEW verfolgten Ideen. Die erste LabVIEW-Version erschien 1986. Seitdem gab es folgende Entwicklungsstufen:

1986: LabVIEW 1.0 für den Macintosh II

1988: LabVIEW 1.2

1990: LabVIEW 2.0

1992: LabVIEW 2.5

1993: LabVIEW 3.0

1996: LabVIEW 4.0 (erst ab 1995 war Microsoft Windows so weit verbessert, dass LabVIEW auch unter diesem Betriebssystem lauffähig wurde)

1998: LabVIEW 5.0

2000: LabVIEW 6.0 (auch 'LabVIEW 6i' genannt)

2001: LabVIEW 6.1

2003: LabVIEW 7.0 (auch 'LabVIEW 7 Express' genannt)

2004: LabVIEW 7.1

2005: LabVIEW 8.0

2006: LabVIEW 8.2 (eigentlich LabVIEW 8.20 wegen damals 20 Jahren LabVIEW)

2007: LabVIEW 8.5

2008: LabVIEW 8.6

2009: LabVIEW 2009 (von da an wurde das Jahr zur Versionsnummer)

.

.

2013: LabVIEW 2013

2014: LabVIEW 2014

LabVIEW hat sich in den letzten Jahren stark verbreitet. Gleichzeitig hat es sich von einer anfangs messtechnisch orientierten zu einer universellen Programmiersprache entwickelt. National Instruments bietet LabVIEW inzwischen längst nicht mehr nur für das Betriebssystem MacOS von Apple-Rechnern an, sondern für eine Fülle anderer Systeme, von denen hier nur genannt seien:

Microsoft Windows 7 (32 Bit, 64 Bit)

Betriebssystem von Sun-Workstations (Solaris)

Linux und andere UNIX-Varianten

Die Aufzählung der verschiedenen Vorteile von LabVIEW sprengt den Rahmen dieses Abschnitts. Nur so viel sei einleitend erwähnt:

LabVIEW erlaubt die Anbindung an gängige Programmiersprachen. Damit wird unter LabVIEW z.B. die Nutzung von C-Code möglich. Von LabVIEW 8.2 an wurde die objektorientierte Programmierung erleichtert.

Heutzutage ist das Internet aus unserem Leben nicht mehr wegzudenken. Konsequenterweise verfügen die heutigen Versionen von LabVIEW über Module, welche die Anbindung an das Internet erleichtern und so z.B. die **Fernüberwachung** von Maschinen und Anlagen erlauben.

1.2 Was will dieses Lehrbuch?

Das vorliegende Lehrbuch führt in die Programmierung mit LabVIEW ein. Es setzt voraus, dass der Leser die Beispiele und Übungen am PC durcharbeitet.

Die neueste Ausgabe von LabVIEW ist die Version 2014 (zum Zeitpunkt der Drucklegung dieses Buches). Die Bilder in diesem Buch und die Beispiele sind durchgängig auf LabVIEW 2014 unter Windows 7 abgestellt.

Man kann davon ausgehen, dass derzeit noch viele Leser Zugang zu einem PC mit den älteren Versionen haben, z.B. wenn sie Mitarbeiter einer Firma sind und sich in die Behandlung messtechnischer Probleme mit LabVIEW einarbeiten müssen. Auch Besitzer von älteren LabVIEW-Versionen können dieses Buch zu Rate ziehen, denn die Unterschiede bei den einführenden Beispielen sind nicht sehr groß. Programme, die mit den Versionen 8.2, 8.6, 2009 erstellt wurden, laufen auch unter der Version 2014.

Doch trifft das Umgekehrte naturgemäß nicht zu, weil jede neue Version auch neue Möglichkeiten bietet. Braucht man diese allerdings nicht, kann man zu einer früheren Version zurückgehen, indem man z.B. bei einem unter Version 2014 entwickeltes Programm 'Für vorige Version speichern...' anklickt und anschließend eine der Versionen 2013 bis 8.2 aussucht. Das Programm lässt sich dann mit der gewählten Version aufrufen.

1.3 Installation

Die Installation der LabVIEW-Version 2014 von DVD ist selbsterklärend. Als Betriebssysteme sind z.B. Microsoft Windows XP, Microsoft Windows 7, ein Apple-Betriebssystem oder Linux geeignet. Die Hardware des PC muss den Anforderungen des jeweiligen Betriebssystems entsprechen. Alle Beispiele in diesem Buch wurden unter den Betriebssystemen Windows 7 (32 Bit) und, soweit möglich, unter Linux getestet.

1.4 Einführendes Beispiel

Angenommen, Sie wollen die Summe

$$c = a + b$$

berechnen. Programmiersprachen wie C, C++, C# und ihre Vorläufer sind so konzipiert, dass sie genau diese Art von Aufgaben perfekt lösen können. Man schreibt einfach:

$$c = a + b;$$

und muss also anscheinend nur das Semikolon hinzufügen.

Doch übersieht man dabei Eingabe und Ausgabe. Der Anwender will Werte für a und b eingeben und am Bildschirm das Ergebnis ablesen. Fügt man die entsprechenden Programmteile hinzu, ist ein C-Programm längst nicht mehr so übersichtlich.

LabVIEW verringert diese Schwierigkeiten mit zwei Methoden:

- **grafische** Programmierung nach dem **Datenflussprinzip**,
- Verwendung umfangreicher **Funktionsbibliotheken** für Ein- und Ausgabe.

Bild 1.1 macht deutlich, dass Eingabe, Ausgabe und mathematische Operationen nach dem Datenflussprinzip organisiert sind. Das ist hier am Beispiel $c = a + b$ erklärt. Die folgende Skizze in Bild 1.2 reduziert dieses Prinzip auf seinen Kern.

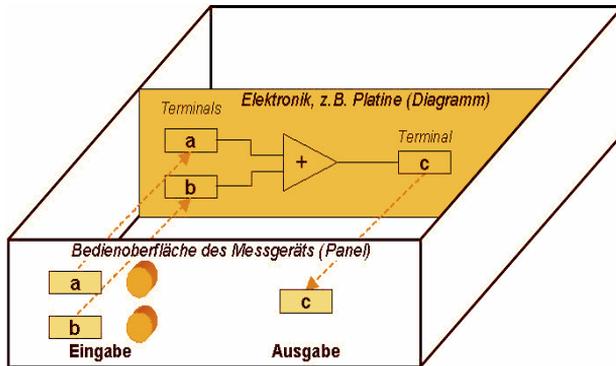


Bild 1.1 Idee der Simulation eines realen Messgeräts mit Hilfe von (Front-)Panel und (Block-)Diagramm in LabVIEW

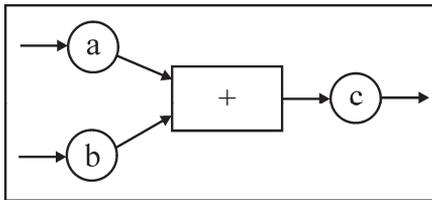


Bild 1.2 Idee der Datenflussprogrammierung am Beispiel von $c = a + b$. Links die Eingabe, rechts die Ausgabe

Aufruf von LabVIEW

Ein linker Mausklick auf das LabVIEW-Icon auf dem Desktop öffnet eine Startseite, im Beispiel dargestellt für die Version 2014. Man kann das aber auch mit 'Programme' – 'National Instruments' – 'LabVIEW 2014' im Windows-Startmenü erreichen. Dann öffnet sich nach einigen Sekunden das in Bild 1.3 dargestellte Fenster.

Dort erlaubt das Feld 'Neu' – 'Leeres VI' die Anfertigung eines LabVIEW-Programms. Mit 'Öffnen' kann man ein bereits existierendes VI von der Festplatte laden, z.B. um es zu modifizieren. 'Beispiele' – 'Beispiele suchen...' ermöglicht das Erlernen von LabVIEW anhand von Beispielen. Hilfreich sind auch die Rubriken 'Hilfe' – 'Erste Schritte mit LabVIEW'. Für Fortgeschrittene, die sich bereits mit einer älteren Version auskennen, sind 'Hilfe' – 'Liste aller neuen Funktionen' und 'Online-Unterstützung' nützlich.

Wählt man nun 'Neu...' – 'Leeres VI', so erscheinen zwei Fenster: das eine mit dem Titel 'Unbenannt 1 Frontpanel', das andere mit dem Titel 'Unbenannt 1 Blockdiagramm'. Sie sind in Bild 1.4 und Bild 1.5 dargestellt.

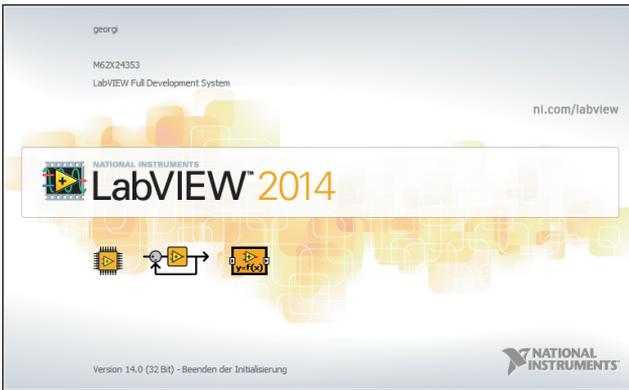


Bild 1.3 Startfenster, das sich beim Aufruf von LabVIEW 2014 öffnet

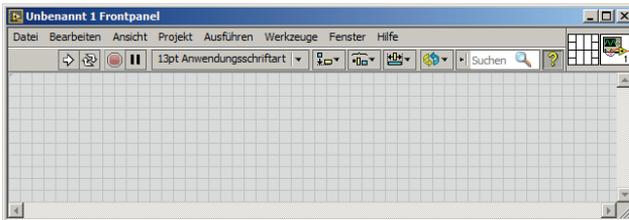


Bild 1.4 Frontpanel oder kürzer 'Panel', für die Benutzeroberfläche des späteren Programms

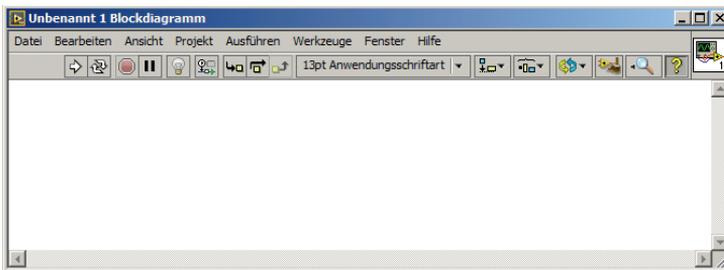


Bild 1.5 Blockdiagramm oder kürzer 'Diagramm', zur Programmierung und grafischen Darstellung des Programms

Wichtig für die Programmierung ist die 'Werkzeugpalette' nach Bild 1.6. Sie erscheint bei entsprechender Voreinstellung automatisch. Wenn nicht, kann man sie vom Panel oder vom Diagramm aus mit 'Ansicht' – 'Werkzeugpalette' holen. Zur Programmerstellung brauchen Sie ferner die Elementpalette gemäß Bild 1.7 und die Funktionenpalette nach Bild 1.8.



Bild 1.6 Werkzeugpalette

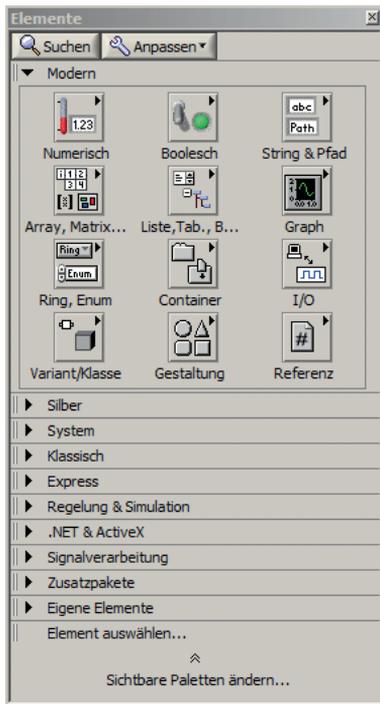


Bild 1.7 Palette mit 'Elementen', die zur Gestaltung des Panels genutzt werden. Hier wird die Palette aus der Kategorie 'Modern' gezeigt. Voreingestellt ist allerdings die Kategorie 'Express'. Doch lässt sich leicht von einer Einstellung zur anderen wechseln.

Man kann auch statt der voreingestellten Palettenkategorie oder zusätzlich zu ihr weitere Kategorien wählen und öffnen wie z.B. 'Silber', 'System' oder 'Klassisch'.

Man erhält die Paletten aus Bild 1.7, indem man im **Frontpanel** (Bild 1.4) mit der **rechten** Maustaste auf die freie Fläche klickt. Mit der **linken** Maustaste kann man diese Palette dauerhaft sichtbar machen, indem man sie 'anpinnt', d.h. die kleine Reißzwecke links oben im zugehörigen Loch versenkt. Danach kann man mit dem Doppelpfeil unten weitere Kategorien anzeigen und gegebenenfalls zu einer von ihnen durch Linksklick wechseln. Erneuter Linksklick lässt die Elemente einer geöffneten Kategorie verschwinden. Auf diese Weise kann man die Ansicht von Bild 1.7 gewinnen.

Die Palette der Funktionen in Bild 1.8 erhält man, indem man im **Diagramm** (Bild 1.5) mit der **rechten** Maustaste auf die freie Fläche klickt.

Man kann diese Paletten auch unter 'Ansicht' - 'Elementpalette' im Panel bzw. unter 'Ansicht' - 'Funktionspalette' im Diagramm finden. Manche Fenster lassen sich ebenso mit **Shortcuts** öffnen, etwa das Blockdiagramm vom Panel aus mit <Strg>+<E> oder das Panel vom Blockdiagramm aus mit denselben beiden Tasten, die gleichzeitig zu drücken sind.

In den folgenden Abschnitten werden wir uns meist auf die Teilpalette 'Programmierung' beziehen. Man kann sie **dauerhaft** zur voreingestellten Palette machen, indem man aus einem VI heraus die Elementpalette (bzw. Funktionspalette) aufruft und nach Anpinnen mit 'Anpassen' - 'Sichtbare Paletten ändern...' ein Fenster öffnet, in dem man eine **einzig**e Kategorie markiert und alle anderen nicht. Beim Rechtsklick auf das Frontpanel erscheint dann später **nur diese** Palette ganz oben und bereits geöffnet. Entsprechendes gilt für die Funktionspalette.



Bild 1.8 Palette mit Funktionen, mit deren Hilfe das Programm im Diagramm erstellt wird. Hier ist die Teilpalette 'Funktionen' – 'Programmierung' geöffnet. Die Voreinstellung bei Version 2014 ist wieder 'Express'. Die letzte Zeile wird nur angezeigt, wenn man vorher das Zusatzmodul 'FPGA Interface' mit installiert hat.

Der Anwender kann leicht von einer Palette auf die andere umschalten. Auch lassen sich zusätzlich andere Unterpaletten wie 'Mess-I/O', 'Mathematik', 'Express' usw. öffnen

1.4.1 Programmierung von $c = a + b$

Folgende Schritte sind auszuführen:

1. Kontrollieren, ob 'Bearbeitungsmodus' eingestellt ist. Das sieht man an den Symbolen (Icons) unter der Menüzeile im Frontpanel. Im Bearbeitungsmodus ist dort das Suchfenster zu sehen. Fehlt es, befindet man sich im 'Ausführungsmodus', der keine Programmierung erlaubt, sondern anzeigt, wie das Frontpanel während der Ausführung aussehen wird. Zur Umstellung 'Ausführen' – 'In Bearbeitungsmodus wechseln' anklicken oder <Strg>+<M> drücken.
2. Eingabefelder für die Variablen a und b anlegen. Zu diesem Zweck 'Elemente' gemäß Bild 1.7 holen, indem man dort Cursor zum Icon links oben bewegt. Es erscheint die Überschrift 'Numerisch'. Ein Mausklick führt zur nächsten Unterpalette. Wiederum das Icon links oben ('Numerisches Bedienelement') anklicken und aufs Panel ziehen. Das Ergebnis dieser Operation sieht man in Bild 1.9. Gleichzeitig verändert sich automatisch das Diagramm durch ein zugeordnetes 'Terminal', siehe Bild 1.10.

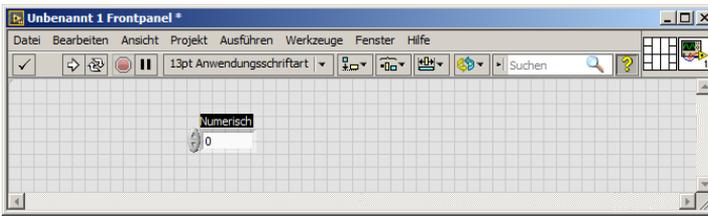


Bild 1.9 Panel mit numerischem (Bedien-)Element

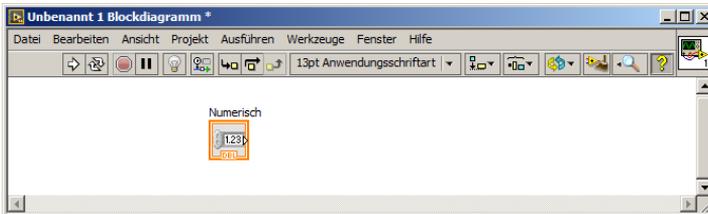


Bild 1.10 Diagramm mit 'Terminal', das dem (Bedien-)Element auf dem Panel entspricht

Das 'DBL' im Symbol von Bild 1.10 bedeutet, dass die Eingabedaten vom Typ 'Double Precision' sind (Gleitkommazahlen doppelter Genauigkeit). Voreingestellt ist auch die Darstellung des Symbols als Quadrat. Wir können jedoch auch auf eine platzsparende Darstellung umschalten, indem wir mit Rechtsklick auf dieses Symbol das Kontextmenü öffnen und dort die Markierung vor 'Als Symbol anzeigen' entfernen. Dann erhalten wir ein kleineres Rechteck. Diese Darstellung können wir auch dauerhaft einstellen, siehe dazu Kapitel 2, Bild 2.4. Wir verwenden für Terminals stets Rechteckssymbole.

3. Das Eingabefeld in Bild 1.9 dient zur Zahleneingabe über die Tastatur. Man kann die Zahlenwerte aber auch mit den Aufwärts-Abwärts-Pfeilen am linken Rand des Bedienelements ändern.
4. Wie schon erwähnt, wird das zugehörige Terminal im Diagramm automatisch gebildet. Man kann es als Darstellung der Durchführung auffassen, die in einem realen Messinstrument vom Gehäuse zur Platine führt und den vom Benutzer eingestellten Wert an die elektronische Schaltung weitergibt. Siehe dazu nochmals Bild 1.1.

Merke: Die Wahl des Datentyps DBL ist voreingestellt. Wir werden später sehen, dass es viele andere Datentypen gibt.

Wir können die Beschriftung 'Numerisches Element' für die Eingabevariable in Bild 1.9 durch einen eigenen sinnvollen Namen ersetzen, z.B. durch 'a'. Das kann entweder unmittelbar nach der Platzierung des Elements auf dem Panel erfolgen, wenn die Schrift noch schwarz unterlegt ist, siehe Bild 1.9. Oder man muss, wenn man die Beschriftung später ändern will, in der Werkzeugpalette das große 'A' wählen, die Beschriftung mit der Maus schwarz einfärben und dann mit der gewünschten Bezeichnung überschreiben. Eine zweite Möglichkeit besteht im Links-Mausdoppelklick auf die Beschriftung, was der Wahl von 'A' entspricht. Das Ergebnis ist in Bild 1.11 dargestellt.

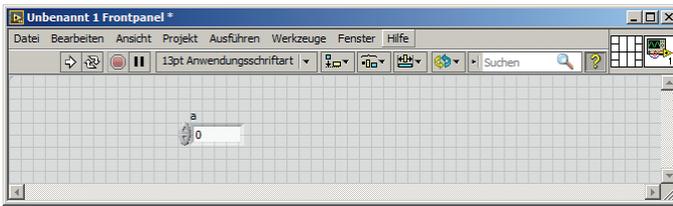


Bild 1.11 Panel mit Bedienelement für die Variable a

5. In gleicher Weise geht man mit der Variablen b um.
6. Dagegen muss man c als Ausgabevariable (Anzeigeelement) etwas anders behandeln, weil ihr Wert nicht vom Anwender gewählt, sondern vom LabVIEW-Programm errechnet wird. Man erhält sie in der Unterpalette der Palette von Bild 1.7 unter 'Numerisch' – 'Numerisches Anzeigeelement'. Bild 1.12 zeigt das Ergebnis.

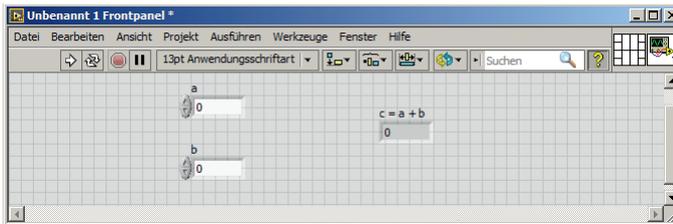


Bild 1.12 Panel mit zwei Eingabevariablen a und b sowie der Ausgabevariablen $c = a + b$

7. Haben Sie versehentlich ein falsches Element platziert, müssen Sie es löschen. Dazu das rechteckige Feld in der ersten Zeile der Werkzeugpalette anklicken und auf die Farbe Grün stellen, was Automatikbetrieb bedeutet. Je nach Anwendung wählt nun LabVIEW automatisch das der jeweiligen Aufgabe entsprechende Werkzeug. In diesem Fall das falsch gesetzte Symbol mit ständig gedrückter linker Maustaste umfahren. Es bildet sich ein gestricheltes Rechteck, dessen Ränder blinken. Nun mit der <Entf>-Taste löschen. Allgemein gilt: Fehler lassen sich mit 'Bearbeiten' – 'Rückgängig...' oder mit <Shift>+<Z> korrigieren. Zurück zum alten Zustand mit 'Wiederherstellen...' oder mit <Strg>+<Shift>+<Z>.
8. Verknüpfen der Eingabe- mit der Ausgabevariablen über die gewünschte Funktion 'Addieren'. Das geschieht im Diagramm. Zunächst wählen Sie in der Funktionenpalette von Bild 1.8 das Symbol 'Numerisch' (Zahlen 123 und Pluszeichen). Die Unterpalette zeigt Symbole, wie man sie von elektrischen Schaltplänen her kennt. Hier ist der Plusoperator links oben anzuklicken und ins Diagramm zu ziehen, siehe Bild 1.13.

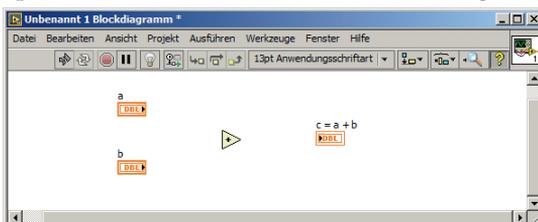


Bild 1.13 Diagramm mit Additionsfunktion in der Mitte

9. Nun sind die Symbole noch mit Drähten zu verbinden, wie schon in Bild 1.2 angedeutet wurde. Dazu dient die Drahtrolle in der Werkzeugpalette. Entweder wählt man diese

Rolle durch Mausklick oder man verlässt sich auf den Automatik-Modus der Werkzeugpalette. Die Drahtrolle wird wirksam, wenn man sich entweder einem Terminal oder einer Funktion nähert. Jedes Icon streckt dann kleine Fühler aus, die Anschlüsse der Funktion. Berührt man mit dem Mauszeiger einen dieser Anschlüsse, verändert er seine Form und wird zu einer kleinen Drahtrolle. Drückt man nun dort die linke Maustaste und bewegt die Maus, so zieht man eine gestrichelte Linie hinter dem Mauszeiger her, mit der sich die Icons verbinden lassen. Die Verbindung ist hergestellt, sobald man an einem der Fühler des zweiten Icons die Maustaste loslässt. Die Wegführung des Drahtes ist ebenfalls zu beeinflussen, indem man an beliebigen Zwischenpunkten die Maus kurz loslässt. Dieser Punkt ist dann fixiert, und der Programmierer kann die Drahtrichtung ändern. Das Endergebnis ist in Bild 1.14 zu sehen.

10. Falsche Verbindungsleitungen kann man löschen, indem man sie hinreichend weit von den Anschlüssen entfernt anklickt und dann auf <Entf> drückt. Dabei wird meist nur ein Teil der Verbindungslinie gelöscht. Alle restlichen Teile beseitigt man mit dem Shortcut <Strg>+. Das ist einfacher, als alle Teillinien einzeln mit <Entf> zu löschen.

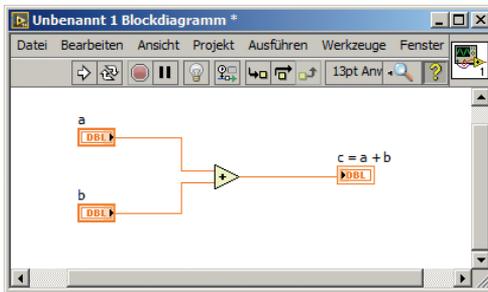


Bild 1.14 Diagramm des fertigen Programms zur Berechnung von $c = a + b$

1.4.2 Speicherung als Programm Add.vi

Das fertige Programm sollte jetzt unter einem einprägsamen Namen auf der Festplatte gespeichert werden. Zum Beispiel könnte man unser Programm mit 'Datei' – 'Speichern unter...' mit dem Namen 'Add.vi' ablegen. Die Datei-Erweiterung 'vi' ist erforderlich, wenn man das Programm später aus einem Ordner heraus mit Mausedoppelklick aufrufen möchte. Sie wird automatisch angehängt und muss nicht mit eingegeben werden. Hier speichern wir das Programm der besseren Auffindbarkeit wegen als '0115-Add.vi'.

1.4.3 Starten und Stoppen von Add.vi

Das Programm kann in den Hauptspeicher geladen werden:

- durch Doppelklick auf seinen Namen in dem Ordner, in dem es gespeichert ist,
- vom LabVIEW-Startfenster aus mit 'Öffnen...' und Pfadwahl,
- von einem anderen, bereits geöffneten VI aus mit 'Datei' – 'Öffnen...' und Pfadwahl.

Sobald Add.vi geladen ist, kann man es vom Panel aus auf dreierlei Wegen starten:

- Durch Anklicken des in der Symbolleiste ganz links stehenden Pfeils. Dann läuft das Programm genau einmal und stoppt dann.
- Durch Anklicken von 'Ausführen' – 'Starten'. Gleiche Wirkung wie oben.

- Durch Anklicken des zweiten Icons links oben mit der Bezeichnung 'Wiederholt ausführen', das zwei verschlungene Pfeile zeigt. Jetzt wird das Programm ständig ausgeführt, und zwar so lange, bis es der Anwender mit Hilfe des dritten Icons mit dem roten Stoppzeichen anhält.

Bild 1.15 zeigt das Programm Add.vi im Modus 'Wiederholt ausführen', wobei der Anwender als Eingabedaten die Werte $a = 7$ und $b = -3$ eingestellt hatte. Er kann diese Werte während des Programmablaufs beliebig verändern. Dazu klickt man das gewünschte Eingabefeld an und ändert den Variablenwert entweder mit den Aufwärts-/Abwärts-Pfeilen oder man klickt direkt ins Eingabefeld und gibt den Wert über die Tastatur ein.

Hinweis: Der Modus 'Wiederholt ausführen' sollte möglichst vermieden werden. Besser lässt man das VI in einer Schleife ablaufen. Näheres siehe Abschnitt 3.4, While-Schleife

Solange im letzteren Fall die Eingabe noch nicht abgeschlossen ist, erscheint in der Symbolleiste links vom Startsymbol ein weiteres Icon mit einem kleinen Haken. Sobald man diesen Haken mit der Maus anklickt, betrachtet LabVIEW die Eingabe als beendet und rechnet von dem Moment an mit dem neuen Variablenwert. Statt den Haken anzuklicken, kann man aber einfacher mit der Maus unmittelbar neben das Eingabefeld klicken oder die Return-Taste betätigen. Auch in diesem Fall verschwindet das Icon mit dem Haken.

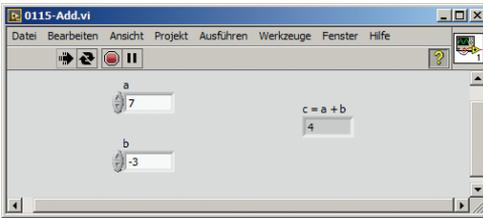


Bild 1.15 Panel im Modus 'Wiederholt ausführen'

1.4.4 Fehlersuche in Add.vi (Debugging)

Die Fehlersuche in diesem VI erscheint unnötig. Das Programm ist zu einfach. Doch lässt sich das Prinzip gut erklären: Zum Debuggen, d.h. Fehler suchen, geht man ins Diagramm und klickt auf das Icon mit der Glühlampe (fünftes Symbol von links, Bezeichnung 'Highlight-Funktion'). Die Lampe färbt sich dann gelb. Ein erneuter Mausklick macht sie wieder weiß. Im gelben Zustand verzögert die Lampe den Programmablauf, so dass man die Datenströme mit bloßem Auge verfolgen kann. Dazu zeigt Bild 1.16 einen während des Debuggens aufgenommenen Schnappschuss. Um den Ablauf noch genauer verfolgen zu können, klickt man auf das Pause-Icon in der Symbolleiste. Damit wird eine Pause erzwungen.

Danach kann man das Programm mit der Pfeiltaste zwei Kästchen rechts von der Glühlampe in Einzelschritten ausführen. Die zwei zusätzlichen Pfeiltasten dienen zum Überspringen von Unterprogrammen bzw. zum Verlassen des VI. Im Beispiel von Bild 1.16 sieht man zwei kleine Kugeln, die an den Terminals für a und b starten und nach rechts laufen. Sie repräsentieren die Datenströme. Im Moment befinden sie sich innerhalb des Additionssymbols. Die Momentanwerte von a und b werden ebenfalls angezeigt. Das Ergebnis $c = 4,00$ erscheint einen Moment später rechts vom Additionssymbol. **Ausprobieren!**