Programmieren mit LabVIEW

M.Goldau

31. Oktober 2002

Inhaltsverzeichnis

1 Einleitung

LabVIEW ist ein grafisches Programmiersystem. Mit ihm ist es möglich mit sogenannten virtuellen Instrumenten, ein Programm zusammenklicken. Ebenfalls kann man mittels LabVIEW Datenerfassungshardware (DAQ-Hardware) ansteuern und somit komplexe Messinstrumente simulieren, so dass hier Kosten erheblich reduziert werden können, und hohe Flexibilität entsteht. Die Haupteinsatzbereiche sind damit die Meß- und Regeltechnik. Wir benutzen LabVIEW hier am Beispiel der Analog-Digital- und Diagital-Analog-Wandlung.

2 Konzepte

Das leere Programm in LabVIEW ist einfach ein weisses „Blatt“ (das sogn. Flussdiagramm).
In dieses werden dann Symbole von Funktionen, bzw. Varaiblen plaziert und evtl. miteinander

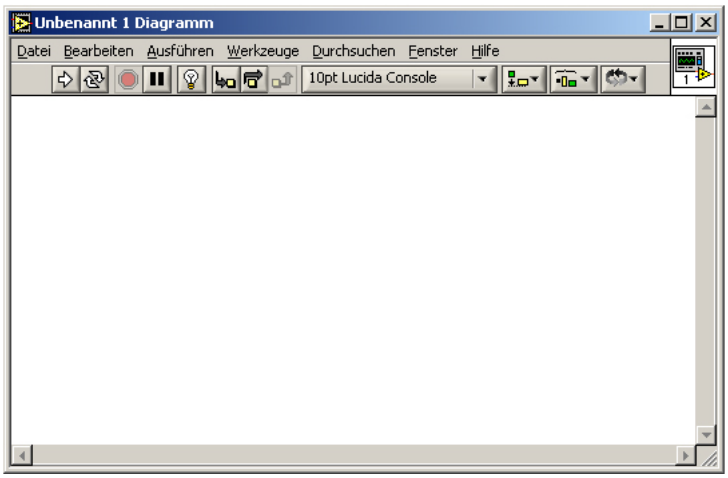


Abbildung 1: Flussdiagramm

verbunden.
Es stellt sich natürlich zuerst die Frage, wo ein Programm beginnt. Dies ist meist LabVIEW

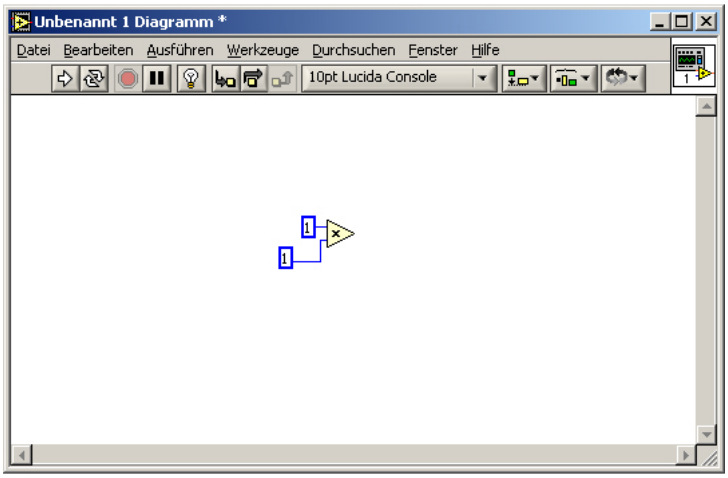


Abbildung 2: Flussdiagramm mit Symbolen (2 Konstanten und Multiplikation)

überlassen, da einige der Symbole Datensenken und andere Datenquellen sind. (strikt sequentielle Verarbeitung ist aber auch möglich! [Sequenz])
Achtung: Kreise (Rekursionen) werden nicht zugelassen, da sonst der „Anfang“ nicht eindeutig zugeordnet werden kann. Die Symbole sind meist so konzipiert, dass man aus ihrem Aussehen

schnell ihre Funktion erkennt. (Dies sollte insbesondere auch bei eigenen VI's berücksichtigt werden!) LabVIEW-Programme werden im weiterem als VI's (virtuelle Instrumente) bezeichnet. Es ist natürlich auch eine VI-Schachtelung möglich. Unterprogramme heißen SUB-VI's.

2.1 Ein- und Ausgabe von VI's

Die Ein- und Ausgabe findet über gesonderte Fenster statt, den sogenannten Frontpanels. Dort werden Datenquellen und -Senken plaziert.
Hier werden Eingaben getätigt und Ausgaben angezeigt. Das eigentliche Programm steht, wie

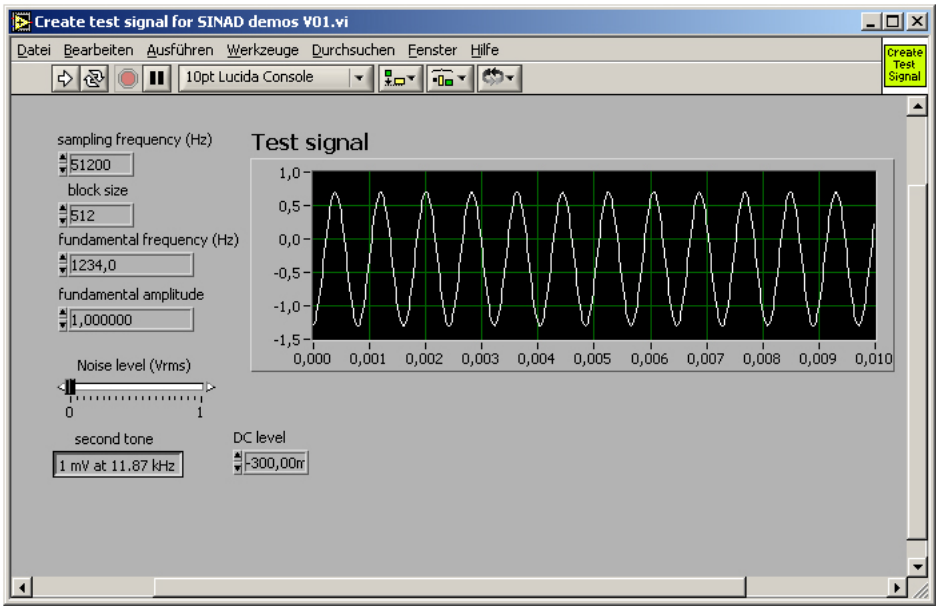


Abbildung 3: Frontpanel mit Inputs Links und Output (Graph)

schon gesagt, in einem anderen Fenster, dem Flussdiagramm. Darin werden die Symbole des Frontpanels (evtl. über Funktionen) miteinander verbunden. Klar ist damit: Jedes Element, das im Frontpanel erscheint, ist auch im Flussdiagramm.

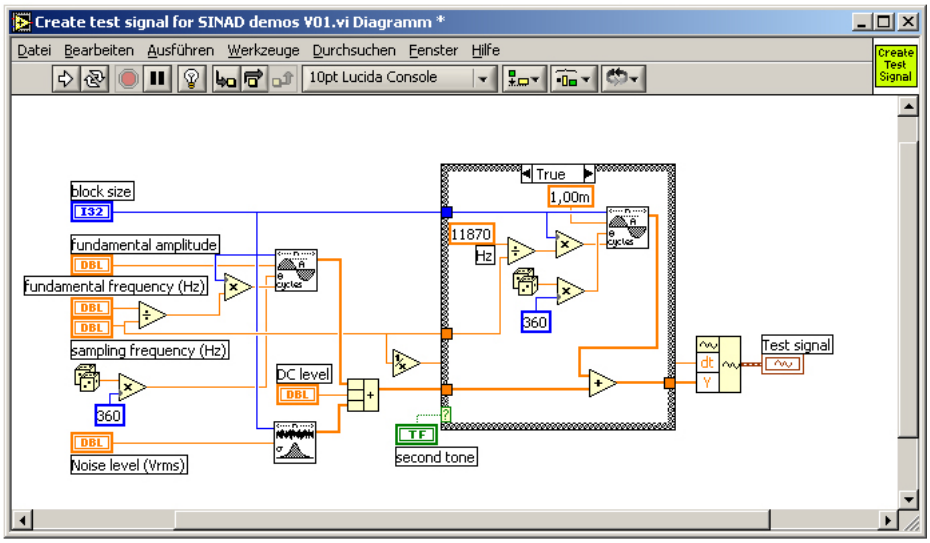


Abbildung 4: Beispiel Flussdiagramm

Wichtig: Elemente, die im Frontpanel erscheinen, können auch NUR dort entfernt werden.

Hinweis: Wenn man ein neues VI erstellt, so erscheinen zwei Fenster. Jenes mit der grauen Oberfläche ist das Frontpanel, das mit der weissen das Flussdiagramm!

2.2 Polymorphie

Polymorphie heißt übersetzt „Vielgestaltigkeit“ und ist hauptsächlich als Begriff für die Vererbungsstruktur in objektorientierten Sprachen bekannt. In LabVIEW bedeutet es, dass es sogenannte polymorphe Funktionen gibt. An diese kann man dann **verscheidene** Eingangstypen legen. Sie regelt selbst, ob zum Biepsiel zwei DBL- oder zwei INT- Zahlen addiert werden sollen.

3 Werkzeug Palette

Abbildung 5: Dies ist die WerzeugPalette.

Dieses Fenster bietet grundlegende Tools zum Plazieren, Verdrahten, und Modifizieren von Objekten. Die meisten Werkzeuge sind selbst erklärend, wenn man mit dem Cursor kurze Zeit über ihnen verweilt.

3.1 Einfügen von Objekten bzw. Symbolen (Konstanten und Funktionen)

Dies geschieht wahlweise mit der „Hand“, „Kabelrolle“ oder dem „Mauszeiger“, durch einen Rechtsklick. Es öffnen sich Menüs, aus denen man auswählen kann.(Im Frontpanel: „Elemente“ im Flussdiagramm: „Funktionen“)

3.2 Hinweise zum Verdrahten

Wie man zwei Objekte miteinander verdrahtet, ist intuitiv klar: Man klickt die Kabelrolle an, danach setzt man den Startpunkt (am Objekt [Objekt blinkt]), und nun kann man den Cursor (und damit die Leitung) an die gewünschte Stelle ziehen. Ist man an dem anderen Symbol angekommen, so genügt ein Linksklick. Linksklicks zwischendurch setzen Wegpunkte.
Es ist hin und wieder günstig, wenn man nur zwei Wegpunkte eines Drahtes setzt: Start- und Endpunkt. Dadurch kann man die Objekte bequem und schnell verschieben (die Drähte gehen mit). (Durch Vermeidung von Drahtkreuzungen und stark polygonen (übermäßig viele Ecken) Leitungen kann die Lesbarkeit enorm gesteigert werden.) Will man wissen, welchen Datentyp ein SUB-VI oder eine Funktion benötigt oder liefert, liest man entweder nach oder geht mit dem Verdrahtungstool auf das entsprechende Objekt. Es erscheinen nun Drahtenden in der Farbe des erwarteten Datentyps. Bleibt man nun noch auf der Stelle des VI's, wo der Draht hinein geht (Anschlußfeld), so kann man nähere Informationen über diesen Anschluß erhalten (Objektname, Eingangsstruktur) . Oft kann man den Datentyp auch sofort erkennen: Bei Datenquellen und -senken ist Farbe und Form des Kästchens entscheidend. (Kapitel 4 „DatenTypen“ geht genauer auf die Beschaffenheit der Kästchen in Abhängigkeit von ihrem Datentyp ein.)

3.3 Das Werkzeug: „Probedaten“

Hier vorab einige Worte zum Debugging. Hat man ein lauffähiges Programm, das aber nicht das erhoffte leistet, ist es wünschenswert, an beliebigen Stellen „in die Drähte hinein schauen“ zu können, um zu sehen, was für ein Wert dort „durchfließt“. Und genau das macht das Werkzeug „Probedaten“. Mit einem Linksklick kann man es auf jegliche Drähte (also insbesondere auch auf Arrays usw.) setzen, die „Meßstelle“ erhält eine Zahl und auch ein Fenster, in dem man die durchfließenden Werte ablesen kann. Zusammen mit dem Einzelschrittmodus kann man sehr einfach alle Werte kontrollieren.

3.4 Das Werkzeug: „Text bearbeiten“

Mit Hilfe dieses nützlichen Tools kann man Graphen skalieren, Cases eines Case-Stmts bearbeiten, oder auch nur seinen „Quelltext“ auskommentieren.

4 DatenTypen

Bedienelemente, wie zum Beispiel das numerische Bedienelement siehe Abbildung 6, sind von

Abbildung 6: numerisches Bedienelement im FronPanel (links) und Flussdiagramm (rechts)

einem speziellen DatenTyp. (Hier „numerisch“ [genauer Double]). In LabVIEW existiert, genauso wie in anderen Programmiersprachen auch, eine Vielzahl an Datentypen:

primitive Typen:	numerische:	Real	single,double & extended precision
		Complex	single,double & extended precision
		Integer	(signed und unsigned), in 8-32 Bit
	nicht numerische:	Boolean	True and False
		String	
zusammengesetzte:	Arrays		mehrdimensional möglich
	Cluster		Record ähnlich

Bedienelemente haben i.a. im Flussdiagramm immer die selbe Form: dicker farbiger äußerer Rand. Anzeigeelemente haben dagegen auch einen Doppelrand, aber die äußere Linie ist normal stark. In den folgenden Abbildungen 7 - 8 sind diese nur als Bedienelemente dargestellt.

Abbildung 7: integer signed [IXX], unsigned [UXX], real [SGL,...], complex [CSG,...]

Abbildung 8: Boolean, String, Array (ohne Element(e)), Cluster (mit Element(en))

Die Elementtypen von einem Array bzw. Cluster können primitive Typen sein, aber nicht nur: Es ist möglich, Arrays in Cluster (und umgekehrt) als Elementtyp einzufügen. Es gibt noch einige weitere DatenTypen: Path, SignalVerlauf, Variant, usw... schnell und übersichtlich in der Quickreferenz nachzulesen (download: ni.com). !!! Für unseren Praktikumsversuch sind sie allerdings (noch) nicht von Interesse. !!! Der Unterschied zwischen einer Datenquelle und einer Datensenke (sprich Bedien- und AnzeigeElement) ist in Abblidung 9 dargestellt.

Abbildung 9: oben: Quellen, unten Senken. Links: Skalar, rechts: Array.

Hinweis: Cluster, die nur numerische Elemente beherbergen, haben eine extra Farbe: braun.

4.1 Erzeugen von Arrays und Clustern

Es stellt sich schnell die Frage: Wie kann ich ein (zunächst) eindimensionales Array vom Typ double im Flussdiagramm erstellen? Dazu bedient man sich entweder der Schleifen und erzeugt somit ein Array oder der Funktion „Funktionen/Arrays/Array erstellen“.
Die letztere erlaubt es, aus einem oder mehreren Skalaren (eines Typs) ein Array zu stellen. Standardmäßig kann man nur aus einem (Startelement) Skalar ein Array erzeugen. Doch durch Hinzufügen weiterer Eingänge über das Kontextmenü kann man auch schnell mehrere Elemente zu einem Array zuusammenfügen. (Aber auch Arrays lassen sich schnell miteinander verschmelzen (Kontextmenü: „Eingänge verknüpfen“!)
Die erste Variante ist ein wenig „dirty“, möchte ich mal sagen. Man läßt z.B. eine For-Schleife bis ca. 99 laufen und setzt in sie nur eine Konstante, die dem Initialisierungs-Wert des Arrays entspricht. Ein Array wird es nur dann, wenn man eine Leitung nach außen (außerhalb der Schleife) zieht, und beim Ausgang die (automatische)„Indizierung“ eingeschaltet läßt (Rechtsklick auf den Ausgang).
[es gibt mit Sicherheit noch weit aus kreativere Varianten ein Array zu erzeugen ;-)]

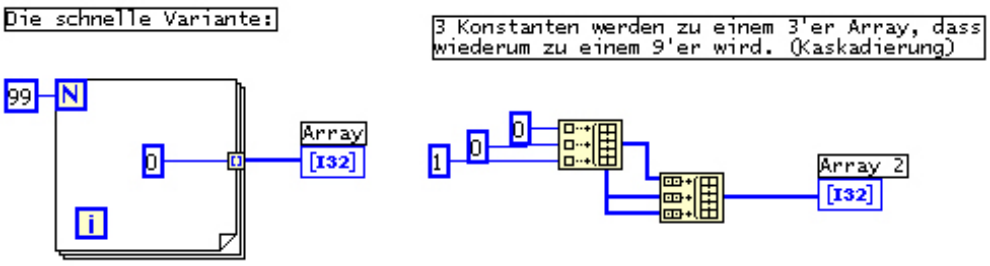


Abbildung 10: Links die „dirty“ Variante, rechts eine Kaskadierung der „Array erstellen“-Funktion

Will man ein Array schon als Bedien- bzw. Anzeigeelement erzeugen, so ist zu sagen: Man nehme ein leeres Array aus dem „Elemente/Arrays und Cluster/Array“-Menü und plaziere hier herein ein Bedien- bzw. Anzeigeelement.

4.2 Verbindungstypen

Man kann anhand der Form des Verbindungsdrahtes erkennen, welcher Datentyp hier verlegt wurde:

Skalar	(normale dünne farbige Line)
Vektor	(dickere Line, in Farbe des Elementtyps) (≡ eindim Array)
Felder	(Doppelline, in Farbe des Elementtyps) (≡ mehrdim. Array)
Cluster	(doppelte Linie mit Punkten drin)

Hinweis: Wo sind die Variablen?
Zum einen kann man auch in Labview lokale und globale Variablen definieren. Dennoch ist dies meist nicht erforderlich, da die Datenquellen einfach Daten liefern, die (ohne Zwischenvariablen) manipuliert werden können und dann wieder ausgegeben werden. Zählvariablen wie z.B. für eine For-Schleife sind schon vorhanden. Über Schiftregister kann man auf iterierte Elemente der Schleife zurückgreifen! (dazu mehr bei den Schleifen).

5 ProgrammStrukturen

Schon öfters erwähnt wurden die Schleifen. Schleife, Sequenz, Case und Formelknoten sind Programmstrukturen. Man kann sie erzeugen über das Funktionenmenü: „Funktionen/Strukturen/...“. siehe Abbildung 11
Die Art und Weise der Plazierung ist allen gemeinsam, einfach die entprechende Struktur ankli-

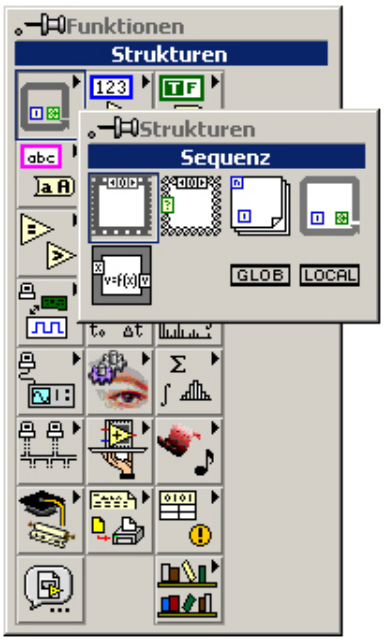


Abbildung 11: Auswahl der Struktur Sequenz. (Daneben Case, For-Schleife, While-Schleife,...

cken, dann links klicken an dem zukünftigen oberen linken Eckpunkt, und soweit ziehen wie nötig, loslassen. Ebenso werden die Kontextmenüs mit der rechten Maustaste meist auf dem StrukturRahmen erreicht. Die Semantik der einzelnen Strukturen sei(bis auf den Formelknoten) vorrausgesetzt.
Der Rahmen von Schleifen unterscheidet sich wesentlich von den Sequenz- und Case-Strukturen, denn in ihnen kann man nicht die einzelnen Schritte manipulieren, d.h. man kann die Schleifen nur von „außen“ steuern, während die Case-Struktur für jeden Case einen eigenen Rahmen mit unterschiedlicher Bearbeitung hat. Ebenso hat Sequenz für jeden Schritt einen eigenen Rahmen. Auf diese Zusatztrahmen kann man in der Kopfzeile des jeweiligen Rahmens zugreifen, und ggf. über das Kontextmenü neue hinzufügen oder entfernen.

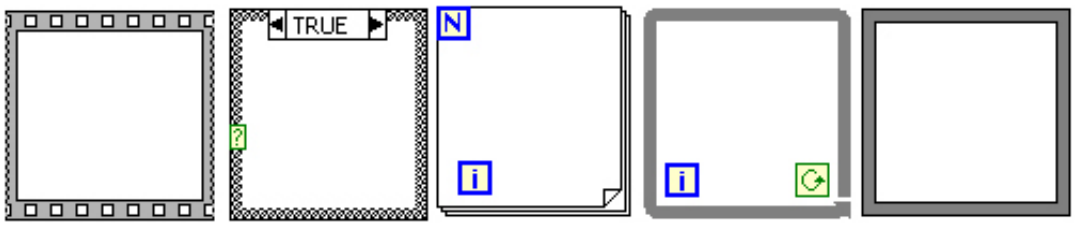
Wichtig :

- JEDES Objekt hat ein Kontextmenüs. Damit kann man die einzelnen Strukturen für seine eigenen Bedürfnisse anpassen.

- Bei Schleifen wird automatisch indiziert! D.h. bei dem Weg von innen nach außen wird z.B. ein Array ausgegeben, statt n Skalare (oder das ErgebnisSkalar). Dies kann man durch einen Rechtsklick auf den Ausgang an der Schleife abstellen.(Kontextmenü)
- Um von Ergebnissen in einem Rahmen in einem anderen (vorherigen) Gebrauch zu machen, kann man sich in der Sequenz mit sogennanten „lokalen SequenzVariablen“ behelfen. Fügt man sie im entsprechenden Rahmen ein und verbindet sie mit einem Wert aus dem Rahmen, so kann man diesen Wert in allen anderen Rahmen „abgreifen“. Bei Schleifen exitstiert ein ähnliches Prinzip. Da Schleifen nur aus einem Rahmen bestehen aber mehere Durchläufe haben, kann man hier mit sogenannten ShiftRegistern arbeiten. (Auch über das Kontextmenü zu erreichen!) Damit kann man jeweils auf die vorherigen Iterierten zurückgreifen.

Hinweis: In LabVIEW gibt es keine direkte IF-Anweisung, sie wird als Spezialfall des Case-Stms angesehen.

Die leeren Strukturen sehen wie folgt aus. Sie werden nun im einzelnen näher beschrieben:



v.l.n.r.: Sequenz,Case,For-Schl.,While-Schl.,FormelKnoten.

5.1 Sequenz

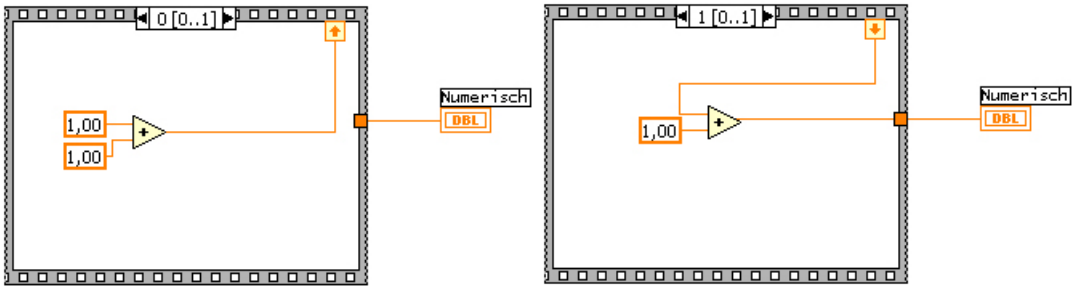


Abbildung 12: Sequenz mit zwei Rahmen, einer SequenzVariable. Ausgegeben wird der Wert 3

In Abb. 12 sieht man zwei Sequenzrahmen, in Wirklichkeit kann man den anderen Rahmen durch Betätigen der kleinen schwarzen Pfeile erscheinen lassen (insofern schon einer vorhanden ist). Weiterhin wirkt hier der Gebrauch von der sogn. Sequenzvariablen selbsterklärend.

Funktion: Jeder Rahmen entspricht einem Verarbeitungsschritt. (Damit wird strikt sequentielle Verarbeitung möglich!), Rahmen danach, davor einfügen bzw. löschen, ... usw. siehe Kontext Menü.

5.2 Case

Ein kleines Beispiel, siehe Abbildung 13. Eine boolsche Caseanweisung: wird der Schalter im Frontpanel auf False gelegt, erscheint im AnzeigeElement: „Bye, bye World.“, sonst „Hello World“.

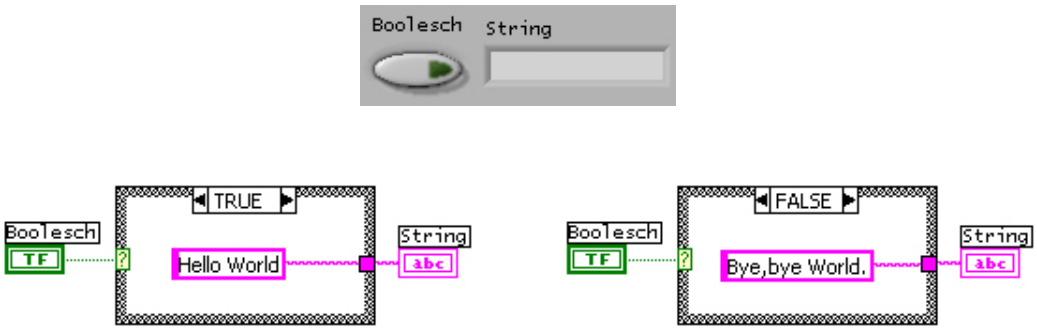


Abbildung 13: Ein einfaches Case-Stmt

Funktion:	Jeder Rahmen entspricht einem Casezweig. Auch hier: Rahmen danach, davor einfügen bzw. löschen, ... usw. siehe Kontext Menü.
Case-Argumente:	z.B.: Boolean, Ganzzahlen, und Strings. (Mit dem „Text bearbeiten“-Tool der Werkzeugpalette bearbeitbar)
Bem.:	Über das KontextMenü kann man einen Default-Wert festlegen, ähnlich wie in textbasierten Hochsprachen.

5.3 For-Schleife

Für die For-Schleife existiert nur ein SubDiagramm, wobei es standardmäßig einen Anschluss hat: „N“-Anzahl der Iterationen und innerhalb die Zählvariable „i“. Beide können vom Typ long sein, auch double ist anschließbar, wird allerdings gerundet. („nochmal“: es gibt nur einen Rahmen im Gegensatz zum Casekonstrukt) Im folgendem kleinen Beispiel, siehe Abblidung 14,

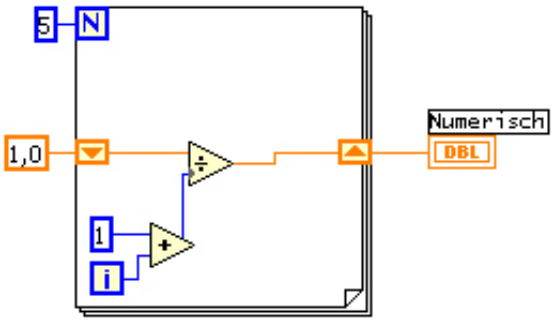


Abbildung 14: Eine einfache For-Schleife mit Shiftregistern

geht die For-Schleife 5 Iterationen durch:

$$ForSchleife_{[0]} = \frac{1}{\underbrace{0}_{i_0=0}+1} = \frac{1}{1} \tag{1}$$

$$ForSchleife_{[i+1]} = \frac{ForSchleife_{[i]}}{i} = \frac{1}{i!} \tag{2}$$

$$\rightarrow (\tfrac{1}{1}, \tfrac{1}{2}, \tfrac{1}{6}, ..., \tfrac{1}{120})$$

Die beiden roten Pfeile am Rand der Schleife sind die Shiftregister. Das linke liefert den Wert, der in der vorherigen Iteration durch das rechte abgelegt wurde. Die double-Konstante „$1,0$“ legt den Startwert eines Shiftregisters fest.

Funktion:	Der Rahmen entspricht einem SubDiagramm für die Bearbeitungsvorschrift der Schleife. N-Anschlussfeld gibt die Anzahl der Iterationen an.
ZählerTypen:	(bis zu) Long, Double(gerundet)
Bem.:	„i“ beginnt bei 0

5.3.1 Automatische Indizierung von Schleifen

Will man ein Array Element für Element abarbeiten, so kann man die Datenquelle (vom Array) direkt in die Schleife legen und dort mit den Elementen arbeiten, dass eine automatische Indizierung erfolgt. D.h. im wesentlichen N=length(Array), und dass im i-ten Schritt das i-te Element des Arrays be/verarbeitet wird. Weiterhin kann man durch das Kontextmenü des Schleifen-Ein- und -Ausgangs diese „automatische“ Indizierung ein- oder ausschalten.
Will man das Array selbst in der Schleife haben, so muß man die automatische Indizierung abschalten.

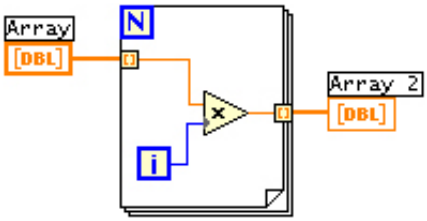


Abbildung 15: Automatische Indizierung bei SchleifenEin- und -Ausgängen.

5.4 While-Schleife

Die Funktionsweise ist klar, die Abbruchsbedingung stellt sich in Form von einem kleinen grünen Kästchen dar, das boolsche Werte akzeptiert. Schleife stoppt standardmäßig, wenn der Ausdruck False ist (im Kontextmenü zu ändern). Weiterhin ist zu sagen, dass eine Zählvariable existiert, die manchmal ganz nützlich ist.

5.5 FormelKnoten

Formelknoten ermöglichen schnell eine komplizierte Formel zu berechnen, ohne sich durch Symbole und Verdrahten den „Hals zu brechen“. z.B.: f(x,y,z)=exp(x^2-sin(y^3-z*x))+cos(z^y); Solch eine Formel kann man mittels dem „Text bearbeiten“-Tool in den Rahmen eines Formelknotens in gewohnter Weise eingeben. Die Variablen müssen dann Eingänge besitzen und die Funktionsbezeichner Ausgänge. (Hier existiert nur EIN Ausgang nämlich f, bei Vektorfunktionen ist das i.a. nicht mehr der Fall.) Eingänge und Ausgänge erstellt man mittels dem Kontextmenü. Es folgt ein Beispiel:

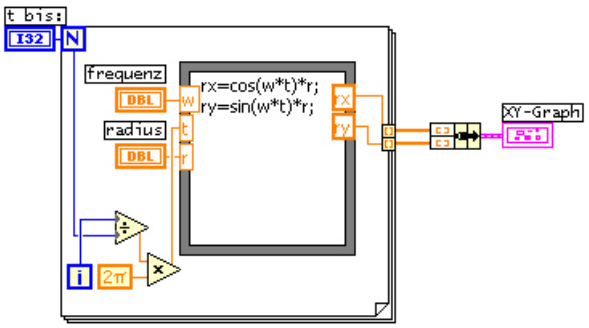


Abbildung 16: ein Formleknoten in For-Schleife

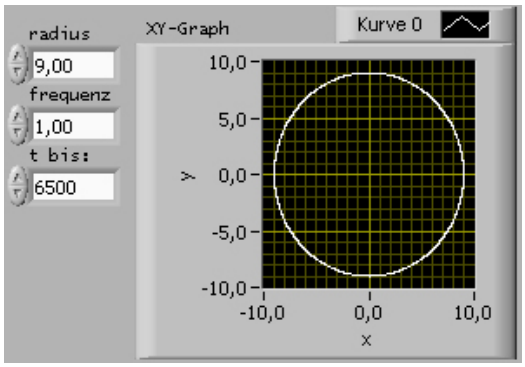


Abbildung 17: und das Frontpanel zur Abb.: 16

6 Programmausführung

Diese kann in zwei Modi geschehen:

1. Einzellauf: das Programm wird einmal durchlaufen und danach beendet
2. Dauerlauf: das Programm wird fortwährend durchlaufen. (kann aber auch mit einer While-Schleife simuliert werden)

Im Frontpanel und im Flussdiagramm sind unter der Menüleiste verschiedenste Icons zu finden. Das erste von links (ein weisser Pfeil), kann zerbrochen dargestellt werden oder ganz. Zerbrochen bedeutet, das Programm hat syntaktische Fehler, die vor dem Programmlauf behoben werden müssen. Klickt man dennoch darauf, so erfährt man etwas über die Fehler (Ausführungsmodus). Ist der Pfeil ganz dargestellt, kann die Programmausführung begonnen werden. Sie endet nach einem Durchlauf. (Variante 1)
Die Schaltfläche rechts daneben ist diejenige für den Dauermodus. Stoppen kann man den Lauf eines Programmes mit dem Roten Knopf. (Er wird erst bei Ausführung richtig rot) Der Pause-Knopf (2 senkrechte Striche), unterbricht einen Programmlauf und wechselt zum Flussdiagramm. Es blinkt die Stelle, bei der angehalten wurde.

Abbildung 18: vlnr. Einzelausführung, DauerAusführung, Stopp, Pause.

7 Fehlersuche

Hauptfehlerquellen sind falsch verdrahtete Ein- und Ausgänge von SUB-VI's. Richtig verdrahtete Symbole haben farbige Linien. Ungültige Verdrahtungen sind schwarzweiss gestrichelt. Ist ein Programm nicht lauffähig, kann man durch Klicken auf den zerbrochenen Pfeil oder durch Menü:„Ausführen/Ausführungsmodus“ Hinweise auf syntaktische Fehler erhalten. (Verweilt man einige Zeit über schwarzweissen Verbindungen, so wird einem auch mitgeteilt, was hier falsch ist.
Es existieren noch zwei sehr interessante Modi, mit denen man debuggen kann.

Highlighting:	Diesen Modus kann man durch Klicken auf die Glühbirne im Flussdiagramm ein- und ausschalten. Er zeigt bei Ausführung den Datenfluss an! (sehr schön zum Nachvollziehen!)
Einzelschrittausführung:	Das Symbol daneben leitet die Einzelschrittausführung ein. D.h. LabVIEW wartet, bis man signalisiert, „den nächsten Schritt bitte“. (LabVIEW geht fast sequentiell SEINE Schritte ab. Dabei kann es natürlich vorkommen, dass LabVIEW in einem Schritt mehrere Schritte parallel simuliert. SEINE Schritte deshalb, weil LabVIEW seine Abarbeitungsrichtung nach den vorhandenen Datenquellen und -Senken orientiert.)

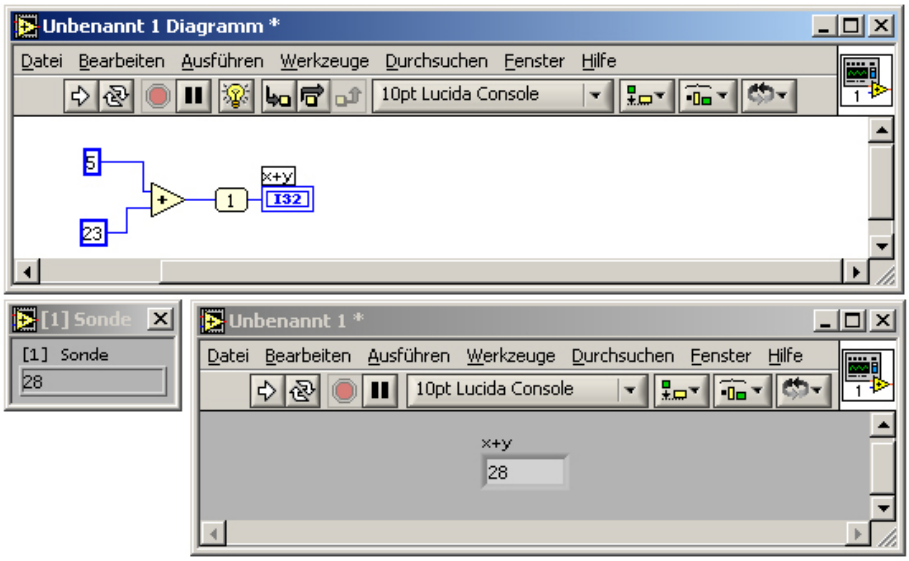


Abbildung 19: Ein simples Beispiel für eine Sonde

Zusammen ergeben Einzelschrittausführung und Highlighting ein schönes Debuggsystem, das mit dem Konzept der Sonden noch mächtiger wird.
Sonden sind quasi Wanzen, die man an beliebige Drähte setzten kann. Sie verfügen über ein geeignetes Anzeigefenster. Während der Ausführung kann man sehen, welcher Wert „im Draht“ ist bzw. am SUB-VI anliegt.
Man erzeugt Sonden, indem man das Werkzeug „Probedaten“ aus der Werkzeugpalette benutzt.

8 Erstellen von SUB-VI's

Hat man ein VI geschrieben, das ein Problem gut löst, und dieses Problem tritt an anderer Stelle wieder auf, so muß man nicht alles neu schreiben, sondern kann sein schon fertiges VI zu einem sogn. SUB-VI erweitern. Dazu muß man i.a. zwei Punkte beachten:

1. Man muß dem VI ein Symbol zuordnen.

2. Es muß festgelegt werden, welche der Bedienelemente zu Eingängen werden.

Ein noch nicht zum SUB-VI erweitertes VI hat ein Standardsymbol. Es liegt in der rechten oberen Ecke vom Frontpanel und vom Flussdiagramm. Durch Rechtsklicken kann man über „Symbol bearb..“ dem Symbol eine aussagekräftigere Gestalt geben. Jetzt zu Punkt 2: Dazu klickt man ebenfalls rechts auf das Symbol, nun aber im Frontpanel. Dort „Anschluß anzeigen“ lassen, und noch einmal rechtsklicken. Es erscheint ein neues Kontextmenü. Über „Anordnung der Anschlüsse“ kann man sich eine Anschlussmaske auswählen oder man kann eine selbst erstellen. Ist man damit fertig, verbindet man jeden Anschluss mit dem jeweiligen Bedienelement

(Anschluß erhält eine Farbe), und fertig ist das SUB-VI. (Die Verbindung erfolgt mittels Verdrahtungstool. D.h. Verdrahtungstool auswählen und normal verbinden)
Nun kann man das SUB-VI über die Funktionenpalette einbinden und benutzen. Dazu ruft man

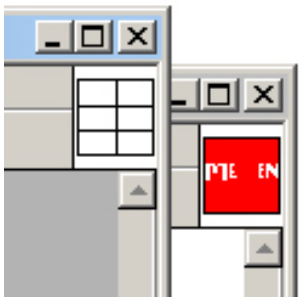


Abbildung 20: Frontpanel und Flussdiagramm mit dem Symbol und den Anschlussfeldern

die Funktion „Funktionen/Wählen Sie ein VI..“ auf (ganz links unten), nur noch verdrahten und fertig.

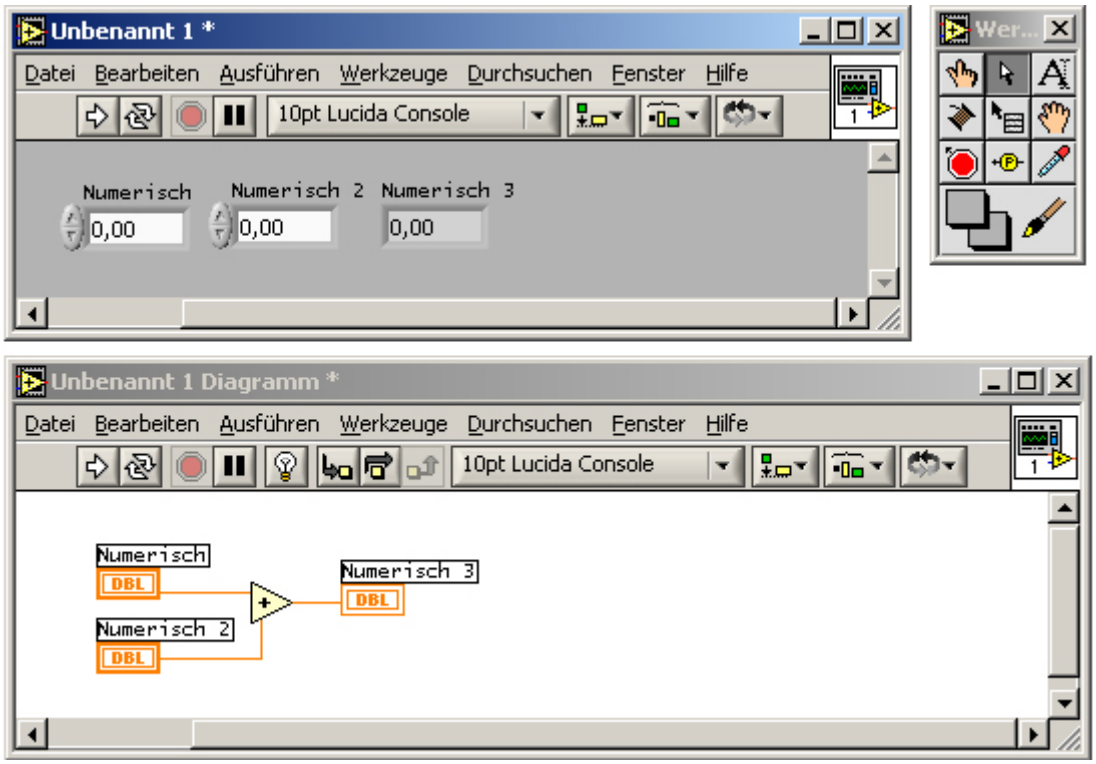
9 Measurements mit spezielle Hardware

LabVIEWs Hauptanliegen ist die einfache Kommunikation von einem PC mit externen Quellen, die Messwerte liefern. Dazu existieren mittlerweile vielerlei Meßkarten, unter anderem auch welche, die schon eine Schnittstelle zu LabVIEW besitzen. So kann man bequem mittels LabVIEW Daten erfassen, senden, Prozesse steuern, Meßwerte archivieren und analysieren,...
Einige Meßkarten sind speziell an LabVIEW angepasst (bzw. LabVIEW hat sich ihrer angenommen). Sie können mit dem sogn. DAQ-Explorer in LabVIEW erreichbar gemacht werden (DAQ-Hardware). Es existiert noch ein anderer Weg: In LabVIEW ist es möglich, Treiberbibliotheken anzuprechen, um so mit der Karte zu kommunizieren. In Windows sind dies insbesondere sogn. dll-Files (Dynamic-Link-Libaries). Dadurch sollte auch anderer externer Code (nicht nur Treiber) ansprechbar sein.
Dies ist im Endeffekt auch die Stärke von LabVIEW, denn wieviel Java/C++ oder C-Code kostet es, Messwerte aus einer Meßkarte auszulesen, über z.B. einen Graphen zu visualisiern, ggf. Werte auszugeben,... ?

10 Beispiele

10.1 Addition

Ziel: Ein VI, das zwei gegebene Zahlen addiert. Das leistet auch eine schon fertige Funktion.
Dazu wechselt man zum Frontpanel, fügt hier jeweils zwei numerisch-digitale Bedienelemente ein und ein numerisch-digitales Anzeigeelement. Im Flussdiagramm sind nun drei Objekte vorhanden. Nun fügt man die Funktion „Addition“ aus dem Funktionen-Menü ein („Funktionen/Numerisch/Addieren“), verbindet die beiden numerisch-digitalen Bedienelemente mit den Eingängen der Addition, und den Ausgang mit dem numerischen Anzeigeelement. und fertig ist das 1. VI:



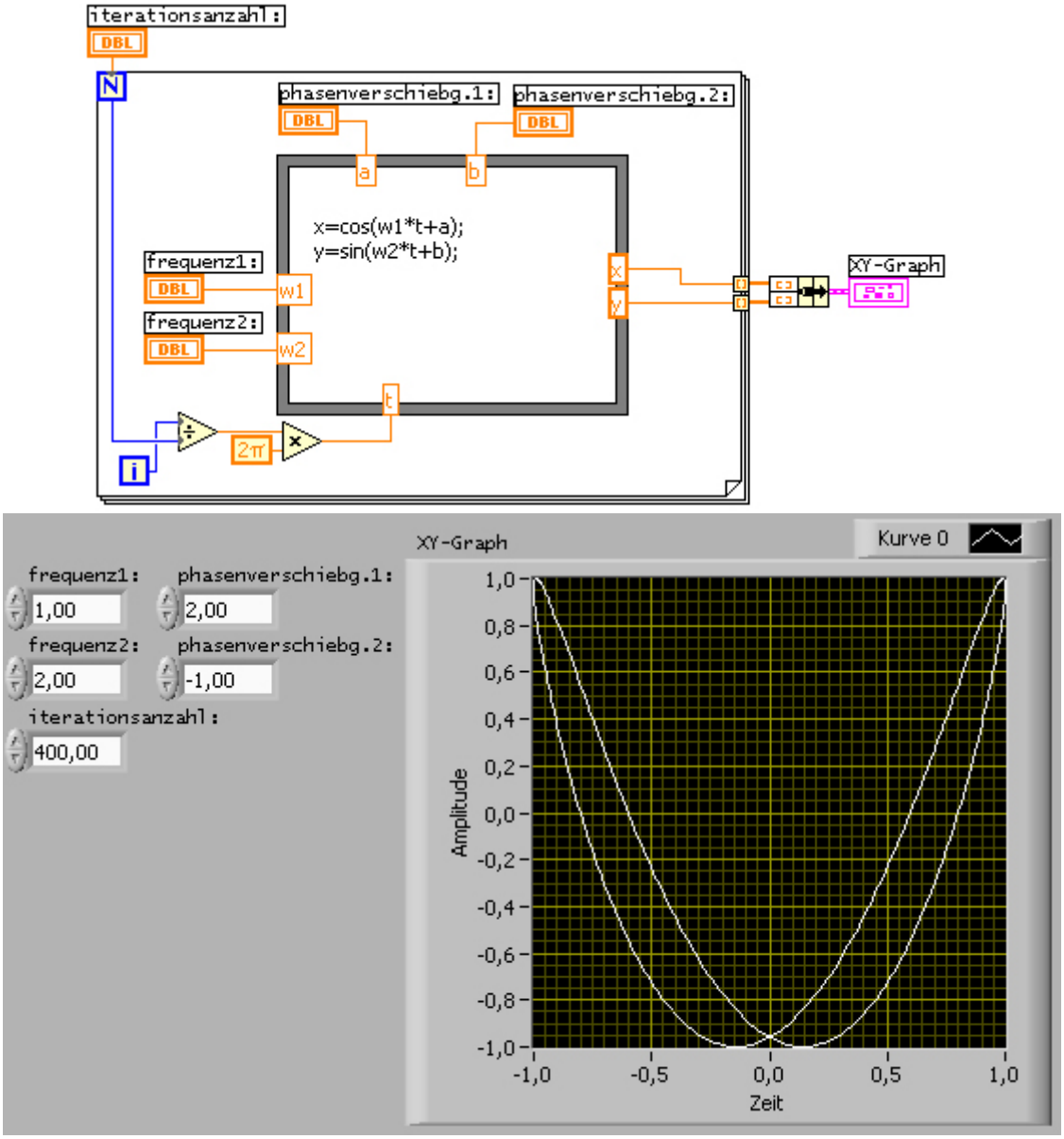
10.2 Lissajoussche-Frequenz-Figuren

In diesem Abschnitt wollen wir ein VI behandeln, das die Lissajous-Figuren erzeugt. Dazu müssen alle nötigen Konstanten wie Frequenzen, Phasenverschiebungen und ggf. Amplituden eingeben werden. Diese Figuren werden durch eine Vektorfunktion beschrieben (mathematisch: Parameterdarstellung einer Funktion; die Zeit t ist hier ein Parameter):

$$x = cos(\omega * t + \varphi_0) \tag{3}$$

$$y = sin(\omega * t + \varphi_1) \tag{4}$$

Dazu benutzen wir vier numerische Bedienelemente, einen Graphen und evtl. einen Formelknoten. Zur Berechnung der Werte mit dem variablen Parameter t ist eine Schleife erforderlich. Das Ergebniss sieht so aus:



Etwas Neues kam hier hinzu: die Funktion „Elemente bündeln“. Sie macht aus zwei oder mehreren Elementen ein Cluster. Sind die Elemente Arrays, so wird eine Reihung von Clustern geschaffen, die z.B. der XY-Graph benötigt.

11 ANHANG:

Wichtiger Merksatz:

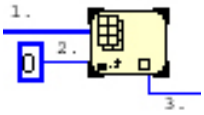
JEDES ELEMENT ODER FUNKTION HAT EIN KONTEXTMENÜ!!! (rechte Maustaste)

Auf einige Funktionen sei nochmals verwiesen:

Problem: Arrayelemente addieren, multiplizieren.
Lösg.: Es existieren dazu numerische Funktionen, mit dem Summenzeichen, bzw. dem Produktzeichen, die das bequem und schnell leisten.

Problem: Wie kann man das i-te Element von einem Array auslesen?
Lösg.: Auch hier existieren verschiedenste Arrayroutinen: „Funktionen/Array/Array indizieren“. Diese Funktion hat standardmäßig drei Anschlüsse: 1. links oben: Eingangs Array, 2.

links unten: Nummer des i-ten Elements, und 3. rechts: Ausgang für angesprochenes Element oder Teilarray. Um ein Teilarray zu erhalten, kann man beim 2. Anschluss das Icon der Funktion mit dem „Position/Größe/Auswahl“-Tool vergrößern und somit Bereiche angeben. (Ähnlich wie Verändern der Größe einer For-Schleife)



Problem: Wie kann man das i-te Element von einem Array verändern?
Lösg.: erst die Funktion: „Array entfernen“ und dann „Array einfügen“ benutzen. Beide sind intuitiv zu benutzen. (evtl. Hilfe von LabVIEW)

Problem: Wie benutze ich die Hilfe von LabVIEW?
Lösg.: hat man eine Funktion die man näher beschrieben haben möchte schon im VI, dann Rechtsklick und Hilfe auswählen. Sucht man hingegen neue Funktionen, so kann man auch in der Hilfe herumstöbern: „Hilfe/Inhalt und Index“ Dann zum Beispiel: „Array indizieren“ eingeben und man findet viele Arrayfunktionen.
Achtung: Die Hilfe besteht aus englischer und deutscher Hilfe. (d.h. viele Themen sind doppelt.)