LabVIEW für Anfänger[1]

Inhaltsverzeichnis

WAS IST LABVIEW? ... 2

BENUTZEROBERFLÄCHE UND BLOCKDIAGRAMM ... 3

TERMINALS, KNOTEN UND DRÄHTE ... 4

ABSPEICHERN UND EINLESEN VON PROGRAMMEN ... 6

FEHLERSUCHE, DEBUGGING ... 6

ERSTELLEN VON SUBVI'S ... 7

PROGRAMMSTRUKTUREN ... 8

DIE FOR-SCHLEIFE UND DIE WHILE SCHLEIFE ... 8
SHIFT-REGISTER ... 9
LOKALE VARIABLE IN LABVIEW ... 10
GLOBALE VARIABLE ... 11
ENTSCHEIDUNGEN (ALTERNATIVEN): IF ... THEN ... ELSE ... ODER CASE-STRUKTUR ... 12
DIALOG-BOXEN ... 14
ABLAUFSTRUKTUR ODER AUCH: SEQUENZ ... 14
FORMELKNOTEN ... 15
PROBLEME BEIM VERDRAHTEN VON PROGRAMMSTRUKTUREN ... 16

ARRAYS UND CLUSTER - ALIAS: DATENFELDER UND DATENSTRUKTUREN ... 17

FELDER ... 17
FUNKTIONEN ZUR BEARBEITUNG VON FELDERN ... 19
POLYMORPHISMUS ... 20
CLUSTER ... 21

DIAGRAMME UND KURVEN ... 23

SKALIERUNG VON CHARTS UND GRAPHEN ... 24
DIE LEGENDE ... 24
DIE PALETTE ... 24

EINFACH- UND MEHRFACH-PLOT-GRAPHEN ... 25

EINFACH- UND MEHRFACH-PLOT-XY-GRAPHEN ... 26

ATTRIBUTKNOTEN ZUM STEUERN VON BEDIENELEMENTEN UND DIAGRAMMEN ... 27

ERZEUGUNG UND VERÄNDERUNG VON ZEICHENKETTEN (STRINGS) ... 28

EIN- AUSGABE IN DATEIEN ... 30

MESSEN MIT LABVIEW ... 33

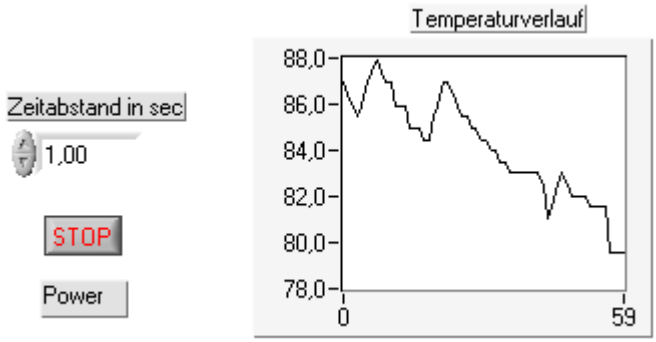
DER GPIB-BUS ... 33
DAQ - DATA ACQUISITION – DATENERFASSUNG PER EINBAUKARTE ... 35
ANALOG INPUT ... 36
ANALOG OUTPUT ... 37
DIGITAL I/O ... 38

[1] Skript für den PC-Pool-Ferienkurs „LabVIEW für Anfänger" Version 9 (10/05, Blersch)

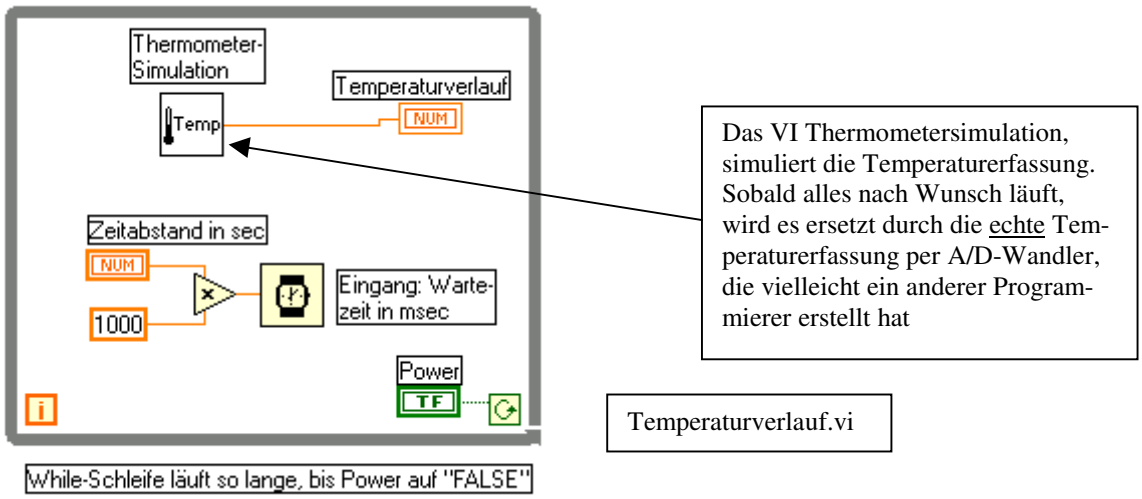
Was ist LabVIEW?

LabVIEW ist eine Programmierumgebung, mit der man effizient und ökonomisch Programme zur Steuerung von Messgeräten und zur Verarbeitung von Messdaten erstellen kann. Eine große Zahl vorgefertigter Funktionen erspart Ihnen die Mühe, diejenigen Räder, die andere schon vor Ihnen erfunden haben, noch einmal erfinden zu müssen. Das, was mich am meisten an LabVIEW verblüfft, ist die „graphische Programmiersprache“, die es erlaubt, Programme zu schreiben, die besser strukturiert und dokumentiert sind, als herkömmliche, z. B. in C oder Pascal geschriebene Programme. Die Grundidee dieser graphischen Programmierung wurde von einer Kursteilnehmerin einmal so zusammengefasst: „LabVIEW ist wie Lego!“. Ganz so einfach ist es sicher nicht, dennoch gilt, dass LabVIEW-Programme leichter erstellt, verbessert, erweitert und verstanden werden können, als textorientierte Programme. LabVIEW ist übrigens die Abkürzung für „**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench.

Angenommen, ein Meteorologe will ein Programm schreiben, das die Aussentemperatur einlesen und in einem Fenster graphisch darstellen soll. Der Zeitabstand zwischen zwei Messungen soll einstellbar sein. Dann könnte dies mit Hilfe von LabVIEW zu folgender Benutzeroberfäche führen:



Das zugehörige Programm – in „graphischer Schreibweise“ – finden Sie hier:

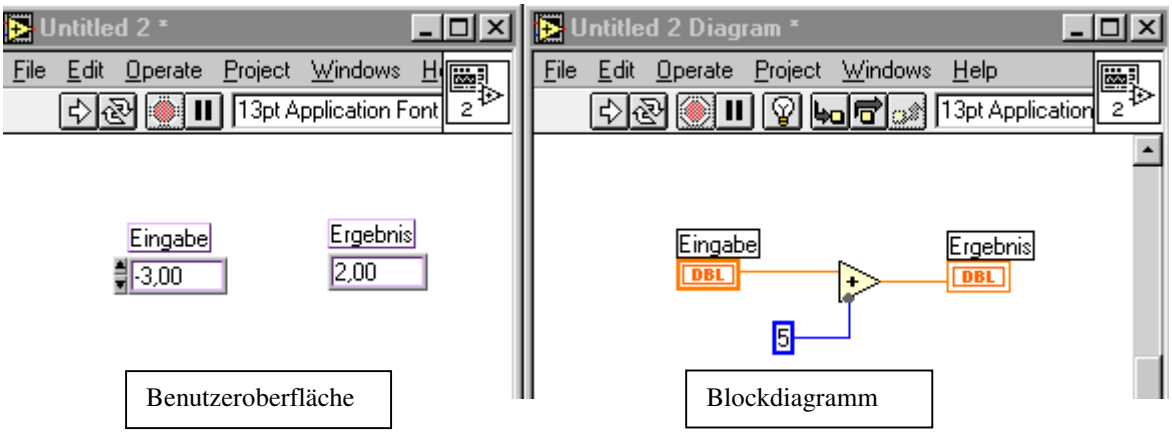


LabVIEW enthält eine große Anzahl von Funktionen, die speziell im Bereich der Meßdatenerfassung und Meßdatenverarbeitung benötigt werden, d.h. in Bereichen, die etwa in der experimentellen Physik von großer Be-

deutung sind. Natürlich können Sie mit diesem Programm auch Daten weiterverarbeiten, die nicht mit LabVIEW erfaßt wurden. LabVIEW kann ebenfalls zur Simulation von physikalischen Phänomenen eingesetzt werden. Noch eine kleine Vorbemerkung zu diesem Skript: weil das Skript sonst zu langweilig und mühselig geworden wäre, gibt es keine genaue Erläuterung aller Menüpunkte des Programms – die meisten darunter sind intuitiv erfassbar. Wenn sich Fragen ergeben: probieren Sie, fragen Sie Ihre Kommilitonen, fragen Sie uns, verwenden Sie die Online-Hilfe oder lesen Sie im Handbuch nach! Dieses Skript soll lediglich einige LabVIEW-typische Probleme und Denkweisen erläutern und darstellen und Ihnen damit den eigenen Einstieg erleichtern. Bearbeiten Sie unbedingt auch die Beispiele und das LabVIEW Tutorial, das beim Aufruf von LabVIEW angeboten wird. Etliche der hier aufgeführten Beispiele stammen von dort. Wenn Sie wirklich ernsthaft vorhaben, ein Projekt mit LabVIEW zu bearbeiten, oder wenn Sie vermuten, daß es für Ihr Problem eigentlich schon eine (Teil-)Lösung geben müsste, sollten Sie unbedingt im Internet nachsehen. Eine wichtige Adressen ist: http://ni.com/devzone. Dort gibt es unter dem Stichpunkt Development Library tausende von kostenlosen Quellcode-Beispielen.

Benutzeroberfläche und Blockdiagramm

Nach dem Start von LabVIEW erhält man zwei Fenster: Benutzeroberfläche (Front Panel) und Blockdiagramm. Die Benutzeroberfläche ist die dem Anwender zugewandte Seite von LabVIEW, d.h. dies ist der Ort. an dem der Benutzer Parameter eingibt, die an das Programm und eventuell an die Geräte weiter gegeben werden und wo auch die Meßdaten (in Form von Tabellen oder Kurven) erscheinen. Hinter dem „Front-Panel“ versteckt sich das Programm, das in der LabVIEW-Sprache „Blockdiagramm“ genannt wird. Dieses Blockdiagramm – obwohl es wie eine Grafik aussieht – **ist** das Programm. Die Zusammenfassung von Panel und Blockdiagramm heißt „Virtuelles Instrument“, kurz: „VI“. Im Normalfall ist das Blockdiagramm selbst wieder aus einer Reihe von Sub-Programmen oder Sub-VIs aufgebaut, usw. Im Unterschied zu herkömmlichen Programmiersprachen erfolgt der Datenfluß in LabVIEW über „Drähte“, die die Ein- und Ausgänge der SubVI’s auf der Diagrammseite miteinander verbinden. (Bei klassischen Programmiersprachen erfolgt dieser Datenfluß über die Parameterübergabe.)
Aufgabe: „Tippen“ Sie das folgende Programm ab und bringen Sie es zum Laufen:

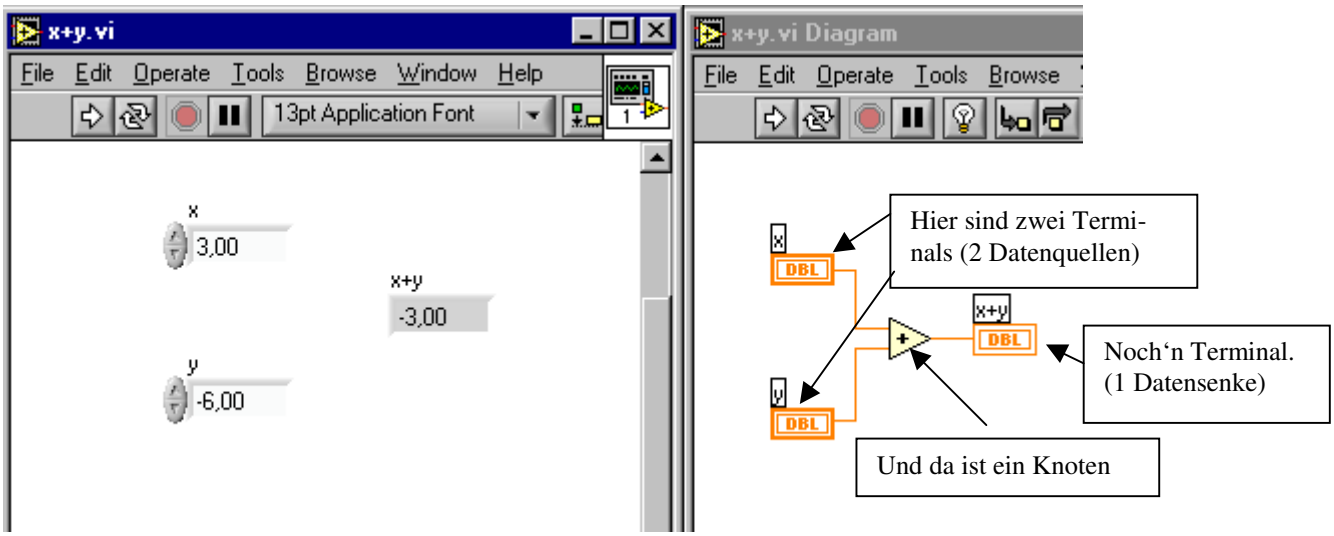


Aufgabe: Rufen Sie LabVIEW auf und erstellen Sie ein Programm, das zwei einzugebende Zahlen addiert und das Ergebnis wieder als Zahl darstellt. Sie benötigen dazu zwei „digital controls“ und ein „digital indicator“-Ein- bzw. Ausgabefelder auf der Panel-Seite. Geben Sie Ihren Objekten immer aussagekräftige Namen, erstellen Sie

das Programm auf der Diagramm-Seite und testen Sie es. Bauen Sie Fehler ein und versuchen Sie, die Fehlermeldungen des Programms zu verstehen.

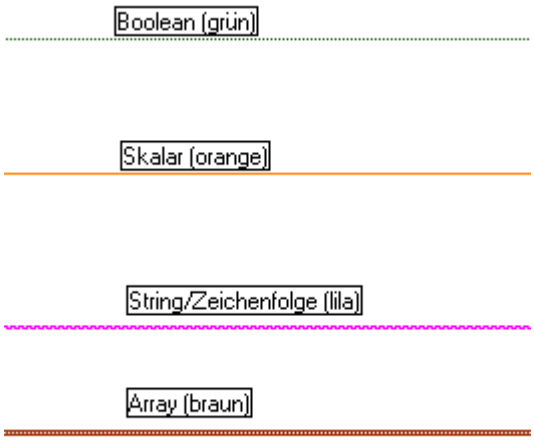
Terminals, Knoten und Drähte

Immer wenn auf der Panelseite ein Ein- oder Ausgabeobjekt plaziert wird, entsteht im Blockdiagramm eine Entsprechung dazu – ein sogenanntes *„Terminal“*. Ein Eingabe-Terminal (ein „digital control“) kann als „Datenquelle“ verstanden werden – aus ihm kommen Daten raus, während man sich ein Ausgabeterminal (ein „digital indicator“) als „Datensenke“ vorstellen kann – in ihm verschwinden Daten. Bestimmen Sie im folgenden Programm die Datenquellen und die Datensenken! Wodurch wird in einem textbasierenden Programm die Reihenfolge der Abarbeitung der Befehle festgelegt? Wie, stellen Sie sich vor, könnte das bei LabVIEW gehen? Es gibt einen Modus bei der Programmausführung, bei dem Sie richtig sehen können, wie kleine Kügelchen (in deren Innern sich die Daten befinden ;-)) aus den Datenquellen herauskommend, die Drähte entlang laufen, um schließlich in einer Datensenke am Drahtende zu verschwinden. (Und was passiert dann?)



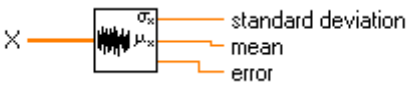
Ein *„Knoten“* ist ein „Programmausführungselement“ – er entspricht in klassischen Programmen einem Operator oder einer Subroutine. Und wie immer müssen Werte/Parameter übergeben und (etwa daraus berechnete) andere Werte zurückgegeben werden. Simples Beispiel: der Plus-Operator im obigen Beispiel ist ein Knoten mit zwei Eingabeparametern und einem Rückgabeparameter.

Ein *„Draht“* ist ein Datenpfad zwischen einem Eingabe-Terminal und einem Ausgabe-Terminal. Drähte können verschiedene Farben und Dicken haben, je nach dem Datentyp, der darüber geschickt wird. Im Folgenden sehen Sie einige Beispiele (allerdings ohne die Farben). Eine Boolesche Variable kann nur zwei Werte annehmen: TRUE oder FALSE. Ein Skalar ist eine „Zahl mit Komma“. Ein String ist eine Zeichenkette aus Buchstaben, Ziffern und Sonderzeichen, wie %,$,>,§. Arrays kriegen wir später.



Ein Array-Draht ist dicker, als ein normaler Draht. Er hat die Farbe seiner Elemente

Klassische Programmiersprachen (Fortran, Pascal, C) beruhen auf Prozeduren. Prozeduren sind selbst Programmstücke, denen Parameter übergeben werden, die dann aus diesen Parametern „etwas“ machen und dann dieses „etwas“ an die aufrufende Instanz zurückgeben. (Beispiel: einer Sortierprozedur wird ein unsortiertes Feld von Zahlen als Parameter übergeben, das dann zunächst der Grösse nach geordnet und anschliessend an die aufrufende Prozedur zurück gegeben wird.) Wie jedes anständige (d.h. gut überlegte und gut strukturierte) textbasierte Programm aus textbasierten Prozeduren aufgebaut wird, wird jedes VI aus SubVI’s aufgebaut. Ein SubVI auch als Icon bezeichnet, entspricht einer graphischen Darstellung mit Ein- und Ausgängen, sogenannten „Connectors“ („Anschlüssen“), mit deren Hilfe die Ein- und Ausgabeparameter übergeben werden können. Üblicherweise liegen die Eingänge auf der linken Seite des Kästchens und die Ausgänge auf der rechten Seite. LabVIEW stellt dem Benutzer eine große Anzahl SubVI’s zur Verfügung. So muß man zur Berechnung der Standardabweichung selbstverständlich kein eigenes VI mehr programmieren, sondern kann auf das entsprechende VI von LabVIEW zurückgreifen:



Obiges VI hat vier Anschlüsse (Connectors): einen Eingabe- und drei Ausgabe-Anschlüsse, jeweils für die Meßwerte (als Eingabe) und die Standardabweichung, das arithmetische Mittel und die Fehlermeldung (als Ausgabe).

Aufgabe: Versuchen Sie, sich einen Überblick über die von LabVIEW angebotenen SubVIs zu verschaffen.

Aufgabe: a) In LabVIEW gibt es im Menü Arithmetik einen einfachen Zufallsgenerator. Platzieren Sie eine digitale Anzeige auf dem Panel und lassen Sie sich einige Zufallswerte anzeigen.

b) Wie kann man ganzzahlige Zufallszahlen zwischen 1 und 49 erzeugen?

Aufgabe: Erzeugen Sie eine zufällige Temperatur t zwischen –35 und 150 Grad. Für t<=0 soll eine blaue, für 0<t<=80 eine grüne und für t>80 soll eine rote Lampe angehen.

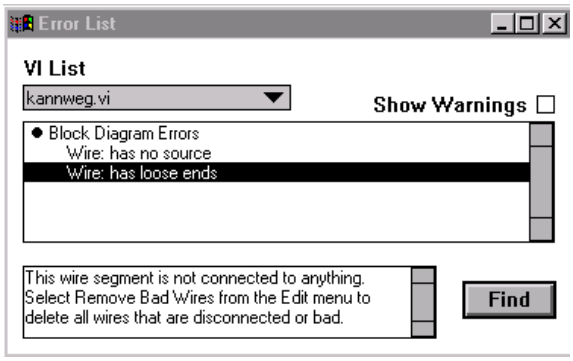
Nach kurzer Zeit stört es, wenn man bei sich häufig wiederholende Aufgaben immer zuerst eine ganze Menge von Menüpunkten durchklicken muß. Deshalb gibt es auch in LabVIEW sogenannte „Shortcuts“, Tastenkombinationen, mit denen Sie schneller ihr Ziel erreichen:
<Ctrl><r> - Run a VI, <Ctrl><z> - UnDo, Letzten Schritt rückgängig machen, <Ctrl><h> - Hilfe-Fenster an/aus, <Ctrl><w> - Aktives Window schließen, <Ctrl><b> - Bad Wires entfernen, <Ctrl><f> - Zwischen Panel-Fenster und Diagram-Fenster hin und her springen. Während des Verdrahtens kann man einen Fehler rückgängig machen, indem man die rechte Maustaste drückt. Objekte können dupliziert werden, wenn man nach dem Markieren die „Ctrl-Taste“ drückt und das zu kopierende Objekt mit der Maus dann an die gewünschte Stelle verschiebt.
Aufgabe: Üben Sie diese Shortcuts in einem kleinen von Ihnen selbst erdachten Testprogramm.

Abspeichern und Einlesen von Programmen

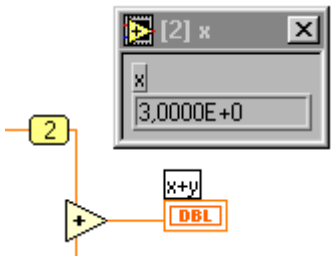
Wie bei jedem anderen Programm können die mit LabVIEW erzeugten Programm-Dateien abgespeichert werden. LabVIEW bietet dazu zwei Möglichkeiten. Einmal können die Dateien mit „Save“ in irgendeinem Verzeichnis abgelegt werden. Zum anderen kann man VI’s in komprimierter Form in einer sogenannten VI-Library abgespeichert werden. Dabei werden die Dateien komprimiert. VI-Librarys stellen sich gegenüber dem Betriebssystem als eine einzige Datei dar. Auf die darin enthaltenen einzelnen VI’s kann man nur per LabVIEW zugreifen. Wenn eine solche Datei beschädigt wird, sind allerdings alle darin enthaltenen VI’s unbrauchbar. Andererseits kann man diese Dateien leichter von einer Rechnerplattform auf eine andere Plattform transferieren. In Dialogboxen sehen Libraries wie Ordner aus. Die Namen dieser Bibliotheken enden mit „.llb“. Falls Sie mit LabVIEW-Programmierung Ihren Lebensunterhalt verdienen müssen, können Sie für Ihre Kunden reine Exe-Dateien erzeugen, die auf dem Kundenrechner installiert werden. Dadurch bleibt der eigentliche Programmcode im Besitz des Programmierers und der Kunde benötigt für seine konkrete Anwendung keine LabVIEW-Entwicklungsumgebung.

Fehlersuche, Debugging

Wenn das Programm nicht lauffähig ist, wenn es also Syntaxfehler enthält, kann man das daran erkennen, daß der „Run-Pfeil“ zerbrochen ist: Ein VI hat normalerweise solange einen zerbrochenen Pfeil, bis es vollständig verdrahtet ist. Ein nicht zerbrochener Pfeil



deutet natürlich nur darauf hin, daß das Programm ablaufen kann, daß also keine Syntaxfehler vorliegen. Er bedeutet auf keinen Fall, daß das Programm macht, was es machen soll! Daß der Pfeil zerbrochen ist, kann auch daran liegen, daß von einem mißlungenen Verdrahtungsversuch noch kleine Drahtstücke im Diagramm herum liegen, die entfernt werden müssen. Dies kann man mit dem Kommando „Remove Bad Wires“ aus dem Edit-Menü (oder <Ctrl><b>) erledigten. Des weiteren kann man sich mit „Show Error List“ die von LabVIEW erzeugte Fehlerliste ansehen und eventuell mit dem „FIND“-Button anzeigen lassen, welches Objekt den Fehler verursacht.



Wenn Sie den Verdacht haben, daß über ein Drahtstück nicht die richtigen Zahlenwerte laufen, können Sie dies nachprüfen, indem Sie (im Run-Modus!) den Draht mit der rechten Maustaste anklicken und eine „Probe“ entnehmen: Sie erhalten ein Fenster in der Nähe des Drahtes, in dem die aktuellen Werte angezeigt werden!

Wenn alles zu schnell abläuft, können Sie entweder Verzögerungsglieder („Time&Dialog / Wait“) einbauen oder das Programm im „Single-Stepping-Mode“, also „Schritt-für-Schritt“ durchtesten. Wenn Sie dabei die kleine Glühlampe anklicken, sehen Sie wie die Daten als kleine Kugeln von den Quellen zu den jeweiligen Senken laufen – spätestens jetzt müßte ihnen allmählich ein (Glüh-)Licht aufgehen über den/die vermutlichen Fehler in Ihrem Programm. Wenn das aber trotzdem immer noch nicht geholfen hat, haben Sie noch die Möglichkeit so-genannte Breakpoints zu setzen. Das sind Punkte an denen das Programm anhält, so daß man in Ruhe die zum Zeitpunkt des Breakpoints berechneten Werte untersuchen kann. Einzelheiten dazu finden Sie im Handbuch!

Erstellen von SubVI’s

Erstellen Sie ein VI, das den Mittelwert von drei Werten x, y und z berechnet. Angenommen Sie benötigen diese Funktion sehr oft, dann ist es wünschenswert, dafür ein eigenes SubVI zu schaffen, das immer wieder eingesetzt werden kann. Wie geht das? Die eigentliche Arbeit, das Programm zu erstellen und zu testen, haben Sie schon gemacht. Die Frage bleibt nur noch: wie macht man daraus ein kleines Kästchen (einen speziellen Legobaustein eben) mit drei Eingängen und einem Ausgang, das im Laufe der weiteren Arbeit immer wieder eingesetzt werden kann? Die Antwort besteht aus drei Schritten:

Erster Schritt: Entwurf eines aussagekräftigen Icons. Dazu klickt man panelseitig mit der rechten Maustaste in das rechts oben plazierte Standard-Icon. Sie finden dort eine einfache Palette von Werkzeugen um ein neues Icon zu zeichnen. Denken Sie daran aussagekräftige Icons zu entwerfen, denn ein gutes Bild ist mehr wert als 1000 Worte – das bestätigt Ihnen jeder Comic-Leser!

Zweiter Schritt: Zuordnung der Anschlüsse. Zuerst im Panel auf die Drahtspule klicken. Durch einen Klick mit der rechten Maustaste in das VI-Symbol in der rechten oberen Panelecke kann man sich eine Tabelle verschiedener Anschlüsse/Connectors anzeigen lassen und darin den geeigneten Typ auswählen. Die Zuordnung der Ein-Ausgabeobjekte aus dem Panel zu den jeweiligen Anschlüssen erfolgt in einem Dreisprung: zuerst wird der Anschluß angeklickt, dann das diesem Anschluß zuzuordnende Objekt und schließlich irgendwo daneben. Die Anschlußfarbe wechselt dabei von weiß zu schwarz zu grau. (Bei neueren LabVIEW-Versionen ändert sich die Farbe). Wenn Sie dann die Help-Ebene einschalten (<Ctrl><h>) erhalten Sie ein Bild, in dem auch schon die Namen der Terminals eingetragen sind (s.u.). Wenn Sie noch weitere Anschlüsse benötigen kann man mit „Show Connectors/Patterns“ weitere Anschlüsse hinzufügen.

3-MW.vi

Dieses VI berechnet den Mittelwert der 3 Eingänge x1, x2 und x3

Dritter Schritt: Dokumentation. In Datei/VI-Einstellungen gibt es eine Kategorie „Dokumentation“ und einen Bereich VI Beschreibung. Dort sollten Sie (unbedingt!) den Dokumentationstext (Prägnanz!) eintragen.

Aufgabe: Erstellen Sie ein SubVI (mit Dokumentation!!) für die Berechnung von

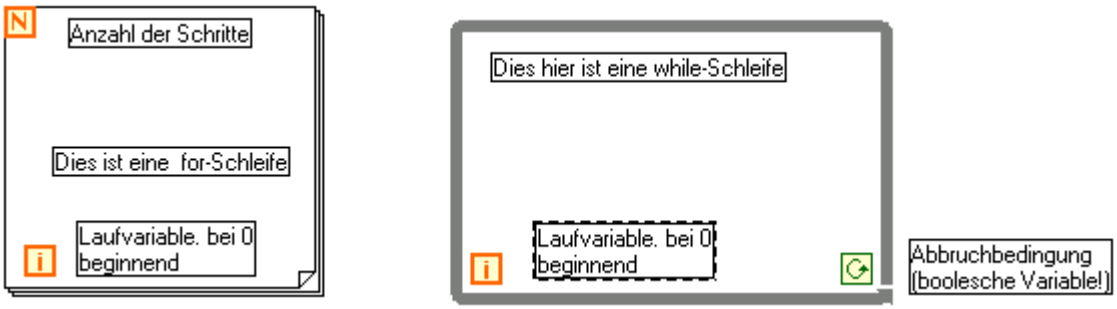
$$\sqrt[3]{\sin(x_1)^2 + \exp(x_2)}$$

Schalten Sie mit Ctrl h die Kontext-Hilfe ein, prüfen Sie, ob Ihr Dokumentationstext erscheint.

Programmstrukturen

Software, so könnte man sagen, besteht aus Wiederholungen mit kleinen Variationen, daher benötigen wir auch in LabVIEW Kontrollstrukturen, die den Ablauf der Wiederholung steuern. Es handelt sich dabei im wesentlichen um for- und while-Schleifen, Alternativen (if ...else, case) und einige LabVIEW-spezifische Strukturen. Alle Strukturen, die im folgenden beschrieben werden, finden Sie diagrammseitig im Menüpunkt „Strukturen“.

Die for-Schleife und die while Schleife



Die for-Schleife führt die Programmteile, die sie im Inneren enthält, genau N-mal aus. Die Variable N muß gesetzt werden, indem eine numerische (ganzzahlige) Konstante mit dem Anschluß „N“ verdrahtet wird. Die Laufvariable „i“ enthält die Nummer der gerade anstehenden Iteration. Die Zählung von „i“ beginnt bei 0! Bei jedem Schritt wird „i“ um den Wert 1 erhöht. Die LabVIEW-„for-Schleife“ ist äquivalent zu folgendem C-Code:

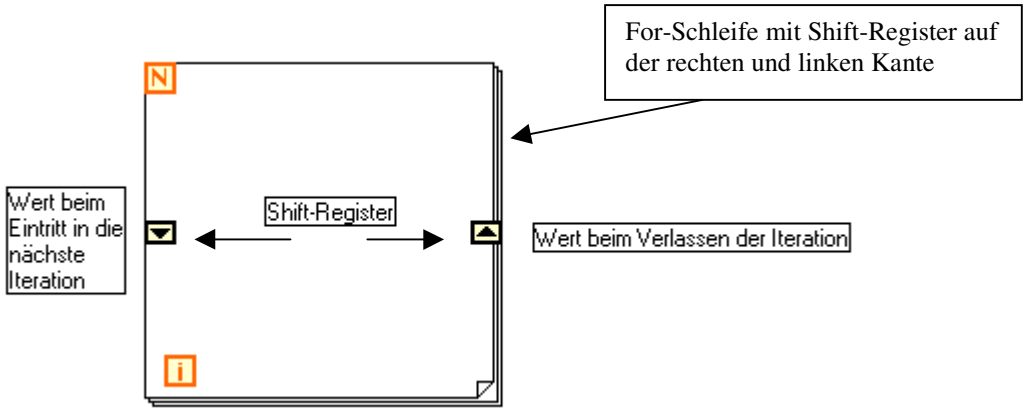
```
for (i=0; i<N;i++) {statement1;statement2; ....}
```

Die while-Schleife führt die in ihrem Inneren enthaltenen Programmteile so lange aus, wie die Abbruchbedingung wahr ist. An die Abbruchbedingung muß also ein logischer Ausdruck angeschlossen werden. Wenn nichts angeschlossen wird, wird der logische Wert „false“ angenommen, was zur Folge hat, daß die Schleife dann genau einmal ausgeführt wird. Die while-Schleife ist äquivalent zu folgendem C-Code:

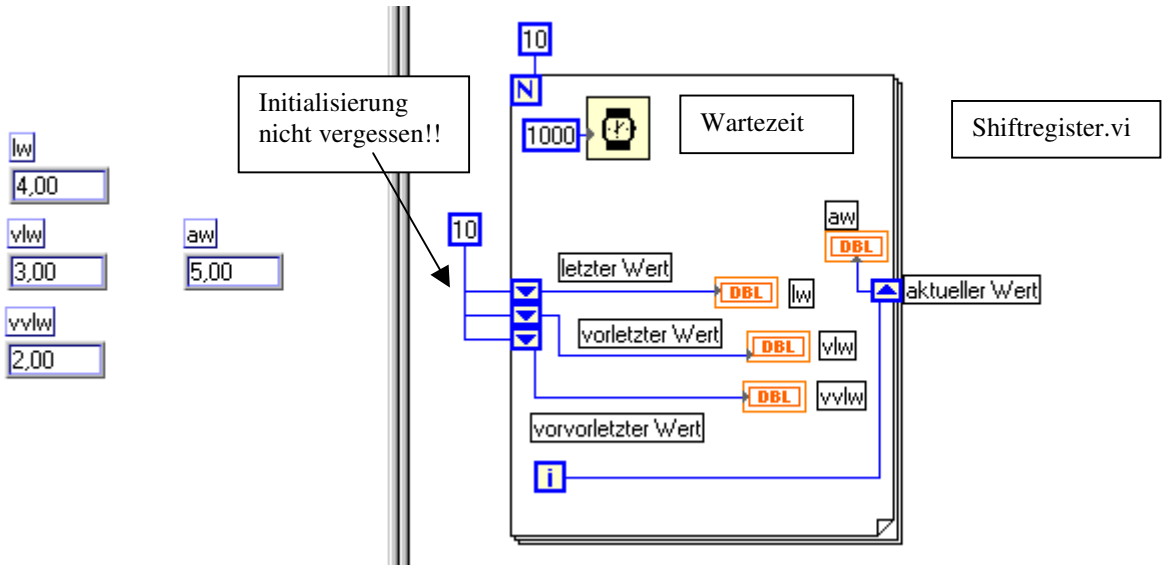
```
do {statement1;statement2; ....} while (condition == TRUE )
```

Aufgabe: Schreiben Sie ein Programm, das bei 10 beginnend, in Abständen von einer Sekunde und in Dreierschritten die natürlichen Zahlen unter 100 ausgibt.

Shift-Register



Shift-Register - verfügbar für while-Schleifen und for-Schleifen - sind lokale Variable, die Werte von einer Iteration der Schleife in die nächste Iteration transferieren. Das Konzept dieser Register ist LabVIEW-spezifisch. In anderen Programmiersprachen wird es nicht benötigt.



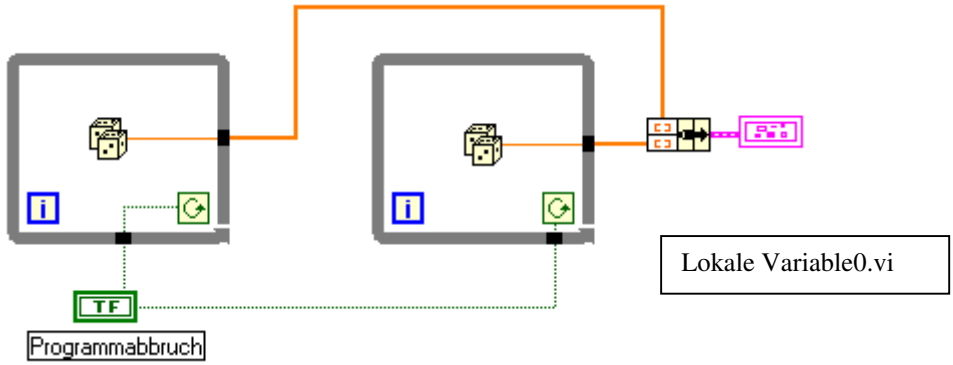
Ein Shift-Register besteht aus einem Paar von Terminals, das man erhält, wenn man mit der rechten Maustaste auf den rechten oder linken Rand der Schleife klickt. Erstellen Sie das obenstehende Programm und studieren Sie sein Verhalten. Die eingebaute Verzögerung von 1000 ms in der for-Schleife hat nur den Sinn, daß Sie den Ablauf Schritt für Schritt verfolgen können. An der linken Seite der for-Schleife können Sie die Initialisierung der Shift-Register erkennen: bevor die for-Schleife anfängt zu laufen werden alle drei Register anfangs auf den Wert 10 gesetzt.

Die Register sind nicht immer paarweise vorhanden! Die Anzahl der Register auf der linken Seite kann durch einen rechten Mausklick erhöht oder erniedrigt werden. Ein typisches Beispiel für die Verwendung ist das sogenannte „averaging“, d.h. die Mittelwertbildung etwa über die 5 letzten Meßwerte.

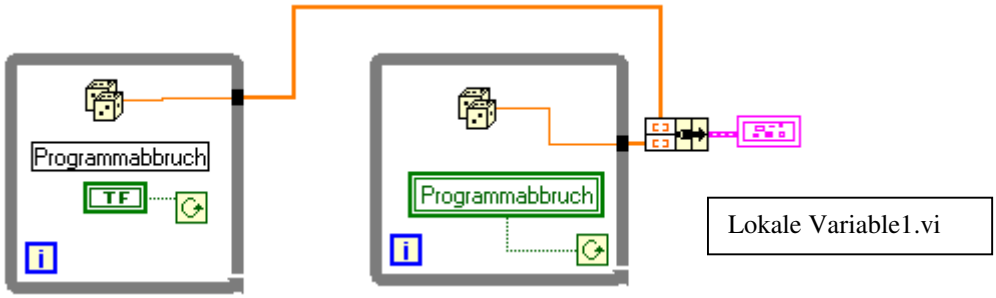
Aufgabe. Schreiben Sie zunächst ein Programm, in dem so lange Zufallswerte ausgegeben werden, bis Sie auf eine Stop-Taste drücken. Ändern Sie das Programm dann so ab, daß immer der Mittelwert über die letzten 5 Zufallswerte ausgegeben wird. Was könnte man machen, damit der Mittelwert über alle bis dato erzeugten Zufallswerte berechnet wird?

Lokale Variable in LabVIEW

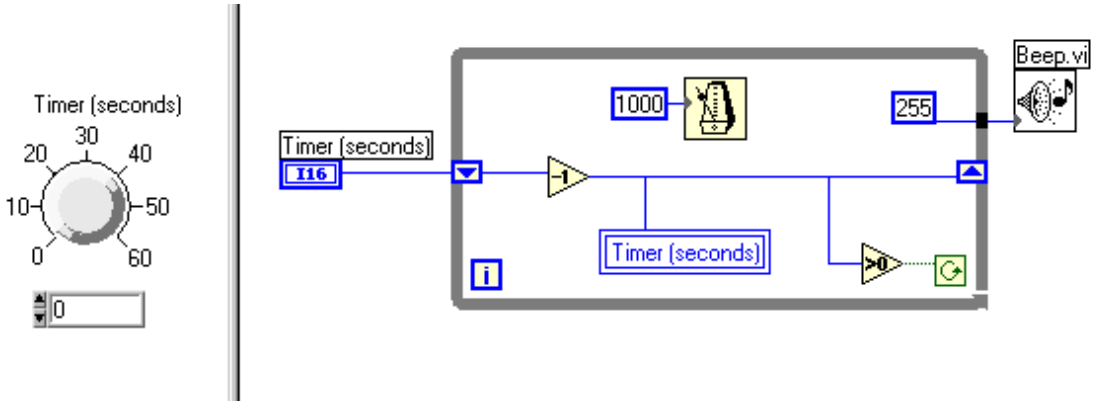
Da wir schon bei lokalen Variablen sind, führen wir hier gleich noch die andere Art lokaler Variabler ein, die dem LabVIEW-Programmierer zur Verfügung stehen. Stellen Sie sich vor, daß Sie in einem Programm den Zugriff auf eine Variablen an vielen verschiedenen Stellen benötigen. Mit vielen Drahtmetern könnten Sie dieses Problem manchmal lösen. Aber Sie erhalten bei dieser Vorgehensweise ein verwirrendes Programm! Tatsächlich gibt es sogar Fälle, wo man grundsätzlich, selbst mit noch so viel Draht nicht weiter kommt. Stellen Sie sich vor, Ihr Programm enthält zwei while-Schleifen, die beide mit der gleichen Bedingung abgebrochen werden sollen, dann bekommen Sie Schwierigkeiten, die Sie am besten selbst am folgenden Beispiel ausprobieren.



Egal wohin Sie die Programmabbbruch-Variable schieben: Sie werden es nicht schaffen, das Programm vernünftig zum Laufen zu bringen. (Schalten Sie in den „High-Lighting-Modus“ um zu verstehen warum es nicht gehen kann).

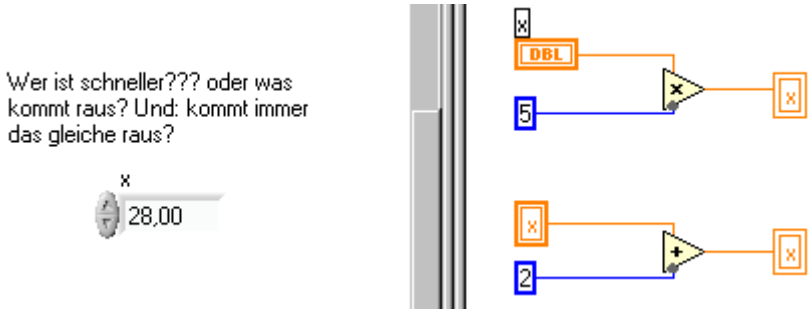


Dieses Problem kann man lösen, wenn in der rechten Schleife eine Kopie der Programmabbruchvariablen eingebaut wird. Lokale Variable finden Sie unter dem Programmpunkt „Structures“, dort wo auch die „while-Schleife“ zu finden ist. Nach dem Anklicken müssen Sie diese lokale Variable mit der Programmabbruchvariablen verbinden – damit erhalten Sie eine Kopie dieser Variablen, die Sie in der zweiten Schleife plazieren können, um so den Wert von „Programmabbruch“ in die zweite Schleife hineinzutransferieren. Lokale Variable haben noch eine wichtige Eigenschaft: Sie können/müssen als Datenquelle oder als Datensenke definiert werden – beachten Sie diesen Punkt, wenn Sie Verdrahtungsprobleme bekommen! Eine lokale Variable kann an der einen Programmstelle im Lesemodus (Quelle) sein und an einer anderen Stelle im Schreibmodus, also Senke! Damit können Sie z. Bsp. eine virtuelle Eieruhr programmieren: eine Uhr also, die auf einen gewissen Wert gestellt (control-Modus) wird und dann bis auf Null zurückläuft (indicator-Modus)



Race-Conditions

Im Zusammenhang mit lokalen Variablen kann man sich nicht ungefährliche Probleme einhandeln. Betrachten Sie das folgende einfache Programm, das drei lokale Variable enthält. Diese Variablen sind alle insgesamt Kopien einer einzigen Variablen: „x“, eine davon im Lesemodus und zwei im Schreibmodus.



Im LabVIEW-Handbuch steht in diesem Zusammenhang: „To avoid race conditions, do not write to the same variable you read from. „

Aufgabe: Ein Messprozess läuft immer in mehreren Schritten ab. Zunächst müssen die angeschlossenen Geräte getestet und die Parameter eingestellt werden. In einem zweiten Schritt erst erfolgt dann die eigentliche Messung. Zum Schluss werden die benutzten Geräte wieder abgeschaltet und die Messung ausgewertet.
Schreiben Sie ein Dummy-Programm, das diese drei Schritte simuliert und in einer String-Variablen Meldungen über den jeweiligen Zustand (Initialisierug, Messung läuft, Messung abgeschlossen) ausgibt. Wenn die Messung läuft soll gleichzeitig noch eine grosse rote blinkende Lampe und ein sogenannter „Fortschrittsanzeiger“ erscheinen, die dann anschliessend wieder ausgehen (oder vielleicht sogar verschwinden) soll.

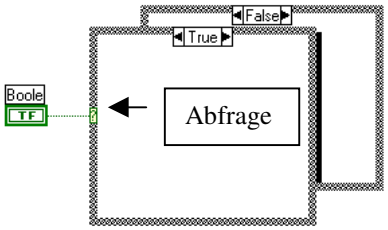
Globale Variable

Wenn Sie schon mit klassischen Programmiersprachen gearbeitet haben, ist Ihnen dieser Begriff sicher bekannt. Lokale Variable sind nur in einem begrenzten Programmbereich gültig, während auf globale Variable aus dem gesamten Programm (lesend und schreibend) zugegriffen werden kann. Das kann zu verwirrenden Situationen und gut versteckten Denk- und Programmfehlern führen, wenn man den Überblick darüber verliert, wer, wann, von weiß nicht wo, eine globale Variable lesen oder ändern darf.
Auch in LabVIEW gibt es lokale Variable (s.o.), die innerhalb eines VIs an verschiedenen Stellen zugänglich sind. Weiterhin gibt es ebenfalls globale Variable, auf die von mehreren quasiparallel arbeitenden VIs zugegrif-

fen werden kann. Und auch in LabVIEW gibt es, wen wundert es noch, die eben beschriebenen Fehlermöglichkeiten. Weitere Ausführungen zu diesem Thema würden den zeitlichen Rahmen eine einwöchigen Einführungskurses jedoch sprengen. Wenn Sie globale Variable einsetzen müssen, empfiehlt es sich, die in LabVIEW zu diesem Thema enthaltenen Beispielprogramme zu studieren.

Entscheidungen (Alternativen): if ... then ... else ... oder Case-Struktur

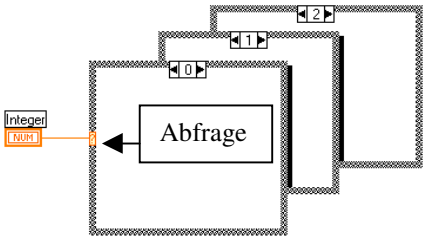
Ein Programmablauf muß sich verzweigen können. Solche Verzweigungen werden in herkömmlichen Programmen mit den Sprachelementen „if {Bedingung == TRUE} then {Anweisung1} else {Anweisung2}“ erzeugt.



Eine weitere Verzweigung, die wahrscheinlich noch häufiger benötigt wird, ist die „switch-case-Verzweigung“. In C lautet sie folgendermaßen:

```
switch(ausdruck) {
        case w1:        anweisung1;
        case w2:        anweisung2;
        case w3:        anweisung3;
        .....           .....;
        default:        sonst-anweisung; }
```

Dabei ist „ausdruck“ eine Variable vom Typ integer. „w1“, „w2“, „w3“,... sind Werte, die „ausdruck“ annehmen kann. „anweisung1“, „anweisung2“, ... sind die den Werten „w1“, „w2“, ... zugeordneten Anweisungen.
In LabVIEW können Sie Alternativen mit einer einzigen Form realisieren: die „case“-Struktur, die entweder eine boolesche „if ... then ... else ... „ – Struktur darstellt, oder die „switch-case“-Struktur:
Welcher der beiden Fälle eintritt, hängt vom Typ der Eingabevariablen ab, die mit der Abfrage verdrahtet wird.

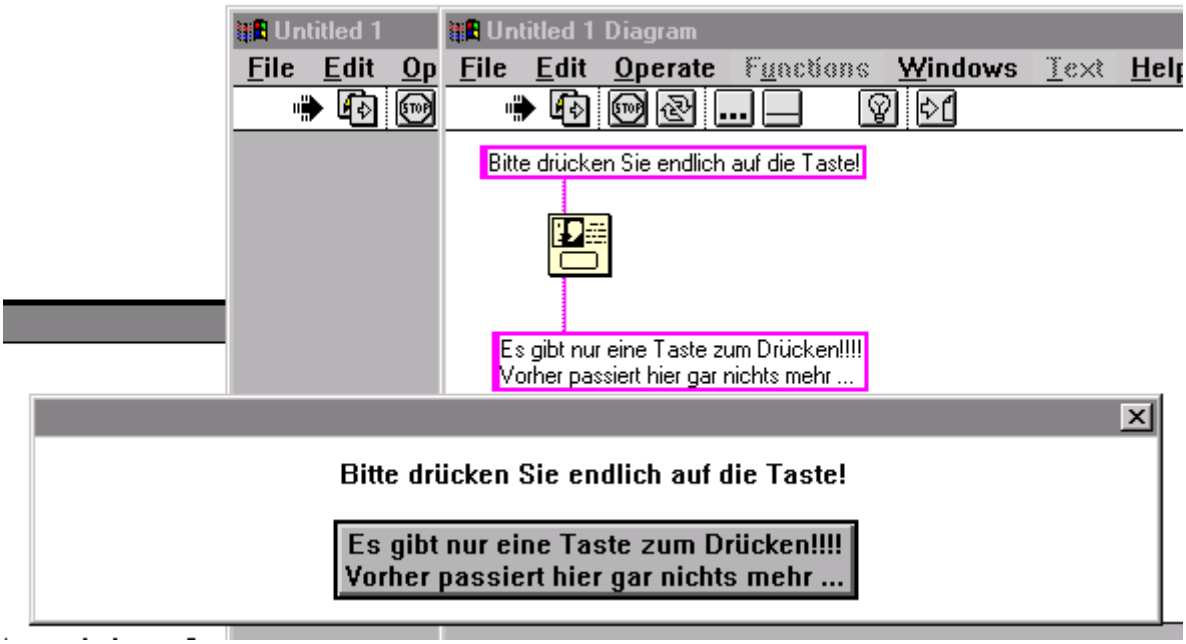


Aufgabe: Schreiben Sie ein kleines Test-Programm, mit dem Sie lernen, mit dieser Alternativstruktur umzugehen, insbesondere, wie man Fälle hinzufügt, verdoppelt, entfernt und verschiebt, und wie man von Fall zu Fall kommt.
Aufgabe: Bauen Sie ein VI, das einen Würfel simuliert und schreiben Sie dann ein Programm zum Testen des Würfels, d.h. zur Klärung der Frage, ob alle Zahlen mit der gleichen Häufigkeit vorkommen.

Aufgabe: Angenommen Sie schiessen auf eine Zielscheibe mit Durchmesser 2, die in ein Quadrat der Kantenlänge 2 eingebettet ist. Angenommen alle Schüsse treffen das Quadrat – das schaffen Sie mit einem Zufallsgenerator, der Punkte (x,y) mit x und y zwischen –1 und 1 würfelt. Wenn x*x+y*y <=1 ist haben Sie ausserdem auch noch die Scheibe getroffen. Das Verhältnis von Treffern zur Anzahl der Schüsse insgesamt konvergiert (ziemlich langsam – aber wozu gibt's Computer?) gegen $\pi/4$. Bauen Sie ein VI um dieses Verhalten nachzuprüfen.

Dialog-Boxen

Mit Dialog-Boxen können aus dem Programm Meldungen an den Nutzer geschickt werden. Sie finden sie im „Zeit & Dialog“-Menüpunkt. Am folgenden Beispiel können Sie erkennen, wie sie eingesetzt werden.



Ablaufstruktur oder auch: Sequenz

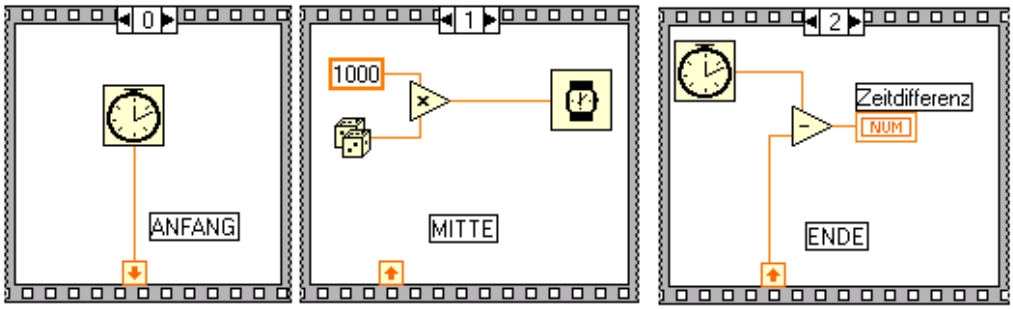
Alle klassischen Programmiersprachen enthalten einen inhärent durch die zeilenweise Abfolge der Programmanweisungen festgelegten Ablauf. Bei LabVIEW ist das anders, denn LabVIEW selbst legt diese Abfolge fest. Wenn Sie dies verhindern wollen, weil Sie auf einen fest (von Ihnen, vom Problem, vom Lauf der Dinge) bestimmten Ablauf angewiesen sind, müssen Sie die Sequenzstruktur einsetzen.

Aufgabe: Schreiben Sie ein Programm, das zwei Eingaben addiert und das Ergebnis ausgibt und mit zwei anderen (von den ersten beiden unabhängigen) Eingaben einige Rechnungen mehr ausführt. Lassen Sie das Programm mit dem Lämpchen laufen (Highlight-Funktion) und beobachten Sie die Reihenfolge der Abarbeitung der einzelnen Programmteile. Sehen Sie jetzt, wie LabVIEW den Ablauf bestimmt?

Aufgabe: Bauen Sie einen „Kleines-Einmaleins-Trainer“, der nicht nur auf richtige Ergebnisse, sondern auch auf die Schnelligkeit der Antworten achtet. Die Programmstruktur sollte wie folgt aussehen:

1. Schritt:	2. Schritt:	3. Schritt:
a) Startzeit bestimmen	a) 2 Zahlen würfeln b) warten bis Ergebnis richtig	a) Stopzeit bestimmen b) Zeitdifferenz berechnen

Die Sequenzstruktur sieht aus wie die aufeinanderfolgenden (in LabVIEW übereinander liegenden!!!) Bilder eines Films, wobei das mittlere Bild durch andere Programmteile ersetzt werden muß, um die oben gestellte Aufgabe zu lösen!

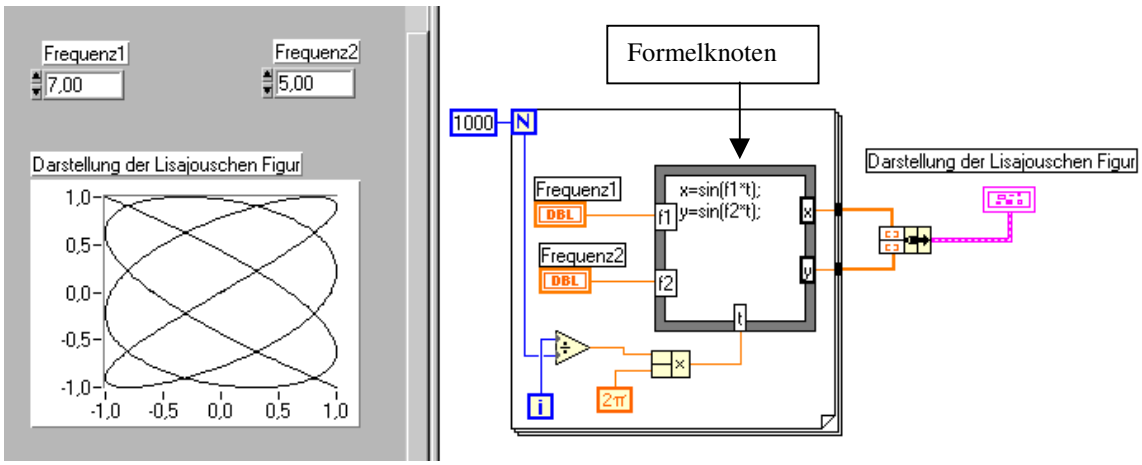


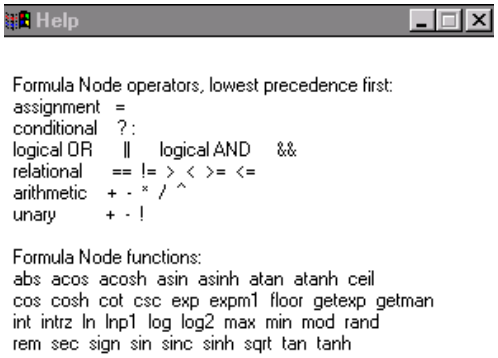
Beachten Sie am unteren Filmrand die sogenannten „lokalen Sequenzen“. Sie dienen der Übergabe eines Wertes aus einem Bild in ein anderes Bild. Man erhält sie durch einen rechten Mausklick auf den Filmrand. Am Anfang sind sie ohne Pfeil, erst wenn sie verdrahtet werden, erkennt LabVIEW, ob hier was raus oder reingehen soll und trägt die entsprechende Pfeilrichtung ein. Achtung: einer lokalen Sequenz kann nur einmal ein Wert zugewiesen werden, in allen anderen Frames kann der Wert nur ausgelesen, nicht aber verändert werden!
Das obige Beispiel mag als Inspiration für die vorstehende Programmieraufgabe genommen werden.
Tatsächlich liegen die drei Bilder des Films hintereinander. Nur hier im Ausdruck sind sie nebeneinander gelegt worden.
In neueren LabVIEW-Versionen gibt es neben der bisherigen Sequenzstruktur eine sogenannte „Flat Sequence Structure“ im Gegensatz zur früher benutzten „Stacked Sequence Structure“. Sehen Sie sich das Teil mal an.

Aufgabe: Erweitern Sie den „Kleines-Einmaleins-Trainer“, indem Sie ein Bewertungssystem einbauen: zuerst eine Note für richtige und falsche Antworten, dann noch eine Note für die Geschwindigkeit.
Aufgabe: Wie lange braucht LabVIEW, um den Wert „1+1“ zu berechnen? Wie lange für √2 (Quadratwurzel von 2)? (Ein winzig kleiner Trick ist dabei notwendig!)

Formelknoten

Der Formelknoten ist eine Box, in die mathematische Formeln eingetragen werden können zur Herstellung von Beziehungen zwischen Ein- bzw. Ausgängen. Unten finden Sie eine Übersicht, der Operatoren und Funktionen die verwendet werden können. Sie können diese Liste jederzeit bekommen, wenn Sie mit dem Cursor auf den Formelknoten zeigen und das Hilfefenster einschalten. Durch Klicks mit der rechten Maustaste auf den Rand der Box kann man entweder Eingänge oder Ausgänge erzeugen, die einen Namen bekommen müssen. Dann werden



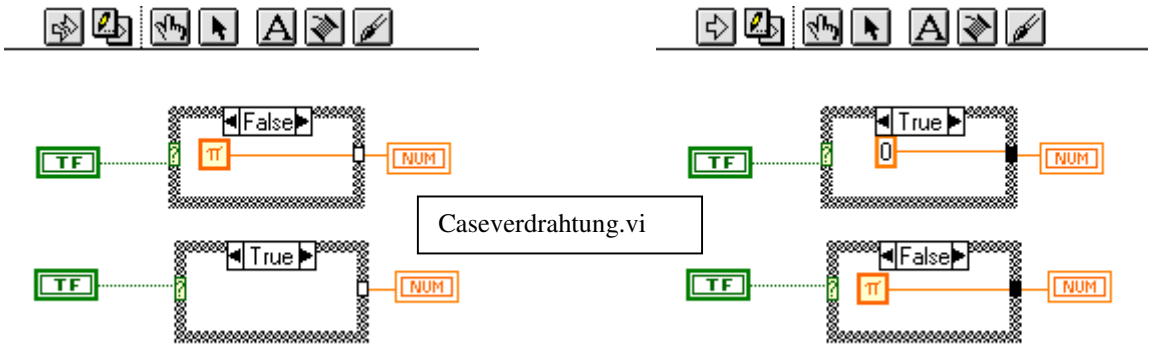


in die Box die jeweiligen Anweisungen eingetragen. Jede Anweisung muß mit einem Semikolon abgeschlossen werden. Das nachstehend Programm könnte man dazu verwenden um die Lissajouschen Figuren zu untersuchen.

Links sehen Sie die Operatoren und Funktionen, die innerhalb eines Formelknotens verwendet werden dürfen.

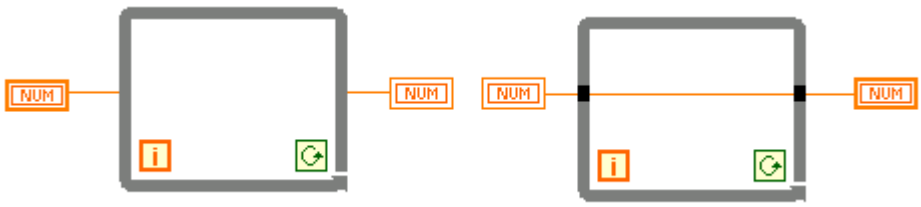
Probleme beim Verdrahten von Programmstrukturen

- Ein häufiger Fehler besteht darin, daß die Verdrahtung nicht korrekt ist: man hat einfach den falschen Draht an den falschen Anschluß angeschlossen!
- Lokale Variable dürfen nur einmal mit einem Wert belegt werden
- Wenn man in einer „Case-Struktur“ einen Draht zu einem äußeren Objekt legt, entsteht ein sogenannter Tunnel. Dieser Anschluß muß in jedem Frame mit einem Wert belegt werden.



Im linken Beispiel ist der Run-Pfeil zerbrochen, weil der „TRUE-Case“ keinen Wert nach draußen liefert.

- Genauso wie Anschlußkontakte zu nahe liegen können, können sich Tunnels überlappen, so daß man Schwierigkeiten beim Anschließen der Drähte erhält.
- Vorsicht beim Verdrahten in Strukturen hinein und aus Strukturen heraus: Es kann passieren, daß der Draht hinter der Struktur verläuft.



Arrays und Cluster - alias: Datenfelder und Datenstrukturen

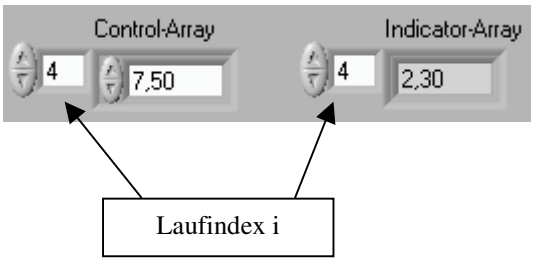
Datenfelder kurz Felder, meist Arrays, sind Listen von Daten vom gleichen Datentyp. Beispiel: eine Tabelle von Meßwerten – alle Elemente der Tabelle sind vom gleichen Typ: Zahlen, nichts als Zahlen, meistens: floating-point Zahlen.

Datenstrukturen (oder auch Cluster) sind Zusammenfassungen von Variablen mit verschiedenen Datentypen in eine neue Datenstruktur. Beispiel: Personaldaten. Ein Personaldatum besteht etwa aus: Name, Vorname, Geburtsdatum, Alter, Monatsgehalt, Anzahl der Kinder, Familienstand. Diese Datenstruktur besteht zunächst aus einigen Zeichenketten, auch Strings genannt (Name, Vorname etwa), Zahlen (Gehalt, Kinderzahl), usw. Für eine effektive Programmierung sind solche Konstrukte sehr brauchbar. Natürlich können Sie auch daraus wieder Arrays aufbauen um darin z.B. den Personalbestand aus einer Personaldatei abzulegen, zu sortieren, usw.

Felder

Zurück zu den Feldern. Felder erzeugt man z.B. bei der Aufnahme einer Meßreihe: wenn Sie 20 verschiedene Temperaturwerte messen, tragen Sie diese Werte in eine Liste ein. So etwas nennt sich dann: Feld. Ein Array können Sie sich auch wie ein 1- oder 2-dimensionales (oder noch höher-dimensionales) Gebilde vorstellen: a[i], oder a[i,j]. Auf die Einzelkomponenten können Sie per Index zugreifen. Beachten Sie immer, daß sowohl in C als auch in LabVIEW die Zählung der Indizes bei Null beginnt!

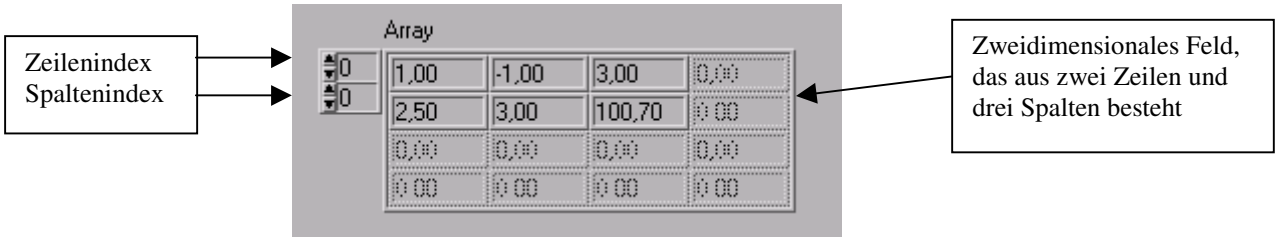
In LabVIEW erzeugt man einen Array in zwei Schritten auf der Panelseite. Dort gibt es zunächst einen Menüpunkt Array & Cluster/Array mit dem ein leeres Array, eine Array-Schale, erzeugt werden kann. Im zweiten Schritt muß gesagt werden, mit welchem Datentyp das Feld gefüllt wird. Angenommen wir wollen ein Feld mit Meßdaten haben, dann muß man sich überlegen, ob die einzelnen Werte des Feldes dargestellt oder verändert werden können sollen. Je nachdem wird man ein „numeric control"-Objekt oder ein „numeric-indicator"-Objekt in das bisher leere Feld schieben.



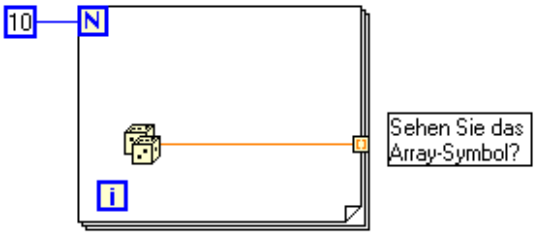
Das linke Beispiel stellt ein Feld von "numeric-controls" dar, während das rechte Beispiel ein Feld von numerischen Anzeigen ist. Der Unterschied ist nicht sofort ersichtlich.

Wenn Sie den Laufindex durchklicken, können Sie die Array-Werte ablesen, beim linken Beispiel können Sie diese Werte auch noch abändern.

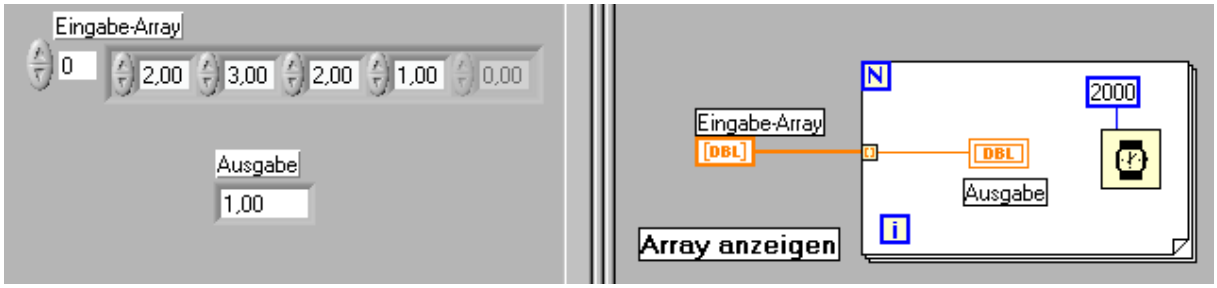
Ein zwei- oder höherdimensionales Feld erhält man mit einem rechten Mausklick auf den Indexbereich, wenn man dann „Add dimension" wählt:



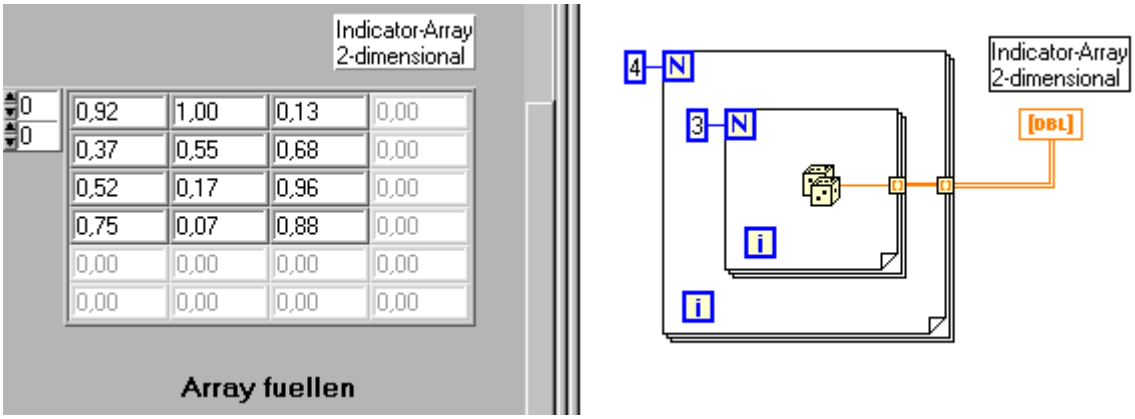
Die for- und die while-Schleife können Arrays auf ihren Rändern aufbauen und indizieren. Das nebenstehende Programm etwa füllt das Numeric-Indicator-Array mit 10 Zufallszahlen. Nach Ablauf des Programms können Sie den Array durchklicken und die erzeugten Zufallswerte nachsehen.



LabVIEW erkennt selbständig, wieviele Werte in einem Array abgelegt sind. Im nächsten Beispiel können Sie das Feld von Hand z. B. mit 4 Werten füllen. Beim Programmablauf muß die Endebedingung der for-Schleife (N) nicht mit der Zahl 4 belegt werde. LabView erkennt selbst, daß das Feld mit 4 Werten belegt worden ist. Diese Möglichkeit des Indexierens kann verändert werden. For-Schleifen indexieren standardmäßig, während while-Schleifen standardmäßig **nicht** indexieren.



Mit dem folgenden Programm können Sie ein zweidimensionales Feld füllen. Die innere Schleife füllt nacheinander die 0-te, erste, zweite und schließlich die dritte (und letzte!) Zeile des 2-dimensionalen Arrays.



Das vorstehende Programm läßt sich übrigens nur dann problemlos verdrahten, wenn der Array panelseitig auch tatsächlich 2-dimensional ist! Wenn die Felder nicht zu groß sind, lassen sie sich aufziehen, so daß man jedes einzelne Feldelement sehen kann. Oben sehen Sie ein zweidimensionales Array, das aus vier Zeilen und drei Spalten besteht.

Aufgabe: Schreiben Sie ein Programm, das bestimmt, wieviele negative und nichtnegative Zahlen in einem eindimensionalen Array reeller Zahlen enthalten sind.

Aufgabe: Erzeugen Sie ein 2-dimensionales (12x7)-Array, das zunächst mit –1 gefüllt wird und das dann (zum Zugucken) zeilenweise mit dem Wert 1000 gefüllt wird. (12x7-Array.vi)

Aufgabe: Erzeugen Sie ein Array von Booleschen Anzeigeelementen. Was kann man machen, damit alle Anzeigen leuchten? Wie kann man zufällige Elemente anschalten. Kennen Sie das „Game of life“? Haben Sie genü-

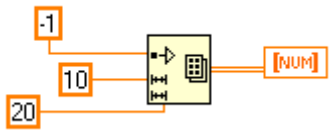
gend Zeit um es in LabVIEW zu programmieren? Ein weiteres nicht einfaches Problem: Programmieren Sie ein Schriftband, bei dem die Zeichen nach links oder rechts wandern.

Aufgabe: Können Sie sich einen dreidimensionalen Array vorstellen? Bauen Sie sich einen mit LabVIEW! Können Sie ihm auch eine physikalische Interpretation geben?

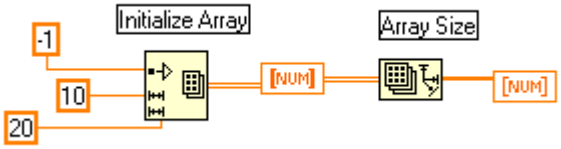
Funktionen zur Bearbeitung von Feldern

LabVIEW bietet Ihnen unter dem Menüpunkt „Array“ eine ganze Reihe von Funktionen zur Bearbeitung von Feldern. Dazu gehören:

Initialisierung eines Arrays – im folgenden Beispiel (Initialize Array) werden alle Elemente einer 10x20-Matrix auf den Wert –1 gesetzt.

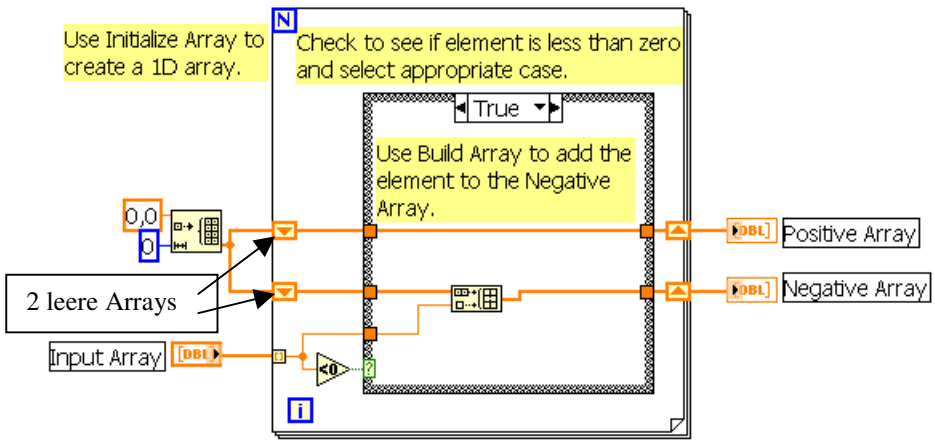


Array-Größe – size(s) gibt einen 1-dimensionalen Array zurück, bei dem das 0., 1., 2., ... Element jeweils die Größe in der 1., 2., 3., ... Dimension darstellt.



Aufgabe: Experimentieren Sie mit den Funktionen „Build Array“, „Array Subset“, „Index Array“. Bei Anfangsschwierigkeiten hilft oft die Hilfefunktion!

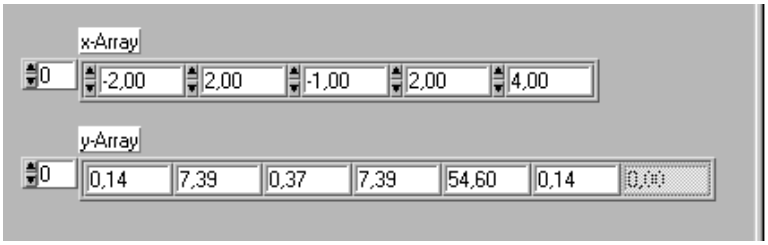
Hier ein kleines Programm, das die negativen und nicht negativen Werte in einem Input-Array findet und in getrennte Arrays (Negative Array, Positive Array) schreibt:



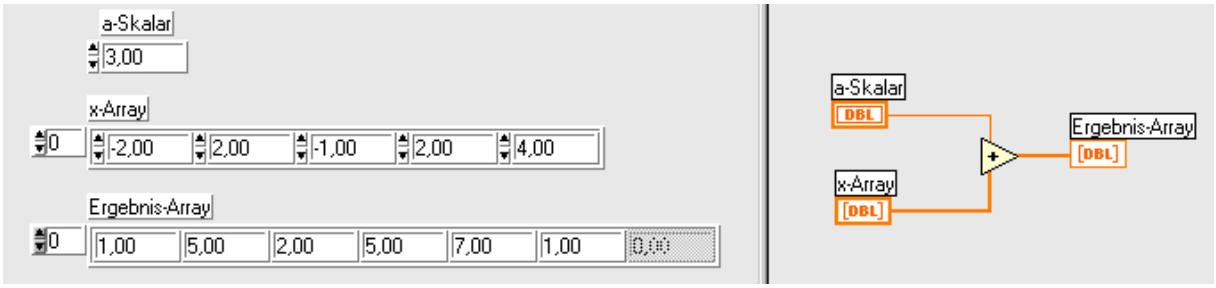
Aufgabe: Gegeben sei die gewöhnliche Differentialgleichung dy/dx = y, deren allgemeine Lösung Sie ja sicher kennen. Bestimmen Sie im Intervall [0,1] eine numerische Lösung, die durch den Punkt (0,1) verläuft. (Hinweis: Euler-Cauchy) Variieren Sie die Schrittweite. Vergleichen Sie Ihre Werte mit der exakten Lösung. Schreiben Sie nun ein VI mit einem Formelknoten im Innern der Iterationsschleife zur Lösung beliebiger gewöhnlicher DGLen y‘ = g(x,y)

Polymorphismus

Viele LabVIEW-Grundfunktionen haben eine weitere angenehme Eigenschaft, die mit „Polymorphismus“ bezeichnet wird, was „Vielgestaltigkeit“ bedeutet. In unserem Zusammenhang ist damit gemeint, daß die Eingänge der arithmetischen Funktionen ADD, MULTIPLY, DIVIDE, SUBTRACT und auch alle anderen Funktionen aus dem „Arithmetic“-Menüpunkt unterschiedliche Datentypen wie „numerics“ oder „arrays“ akzeptieren, und daß sie (die Funktionen) sich dann trotzdem „vernünftig“ verhalten, wie folgende Beispiele zeigen:



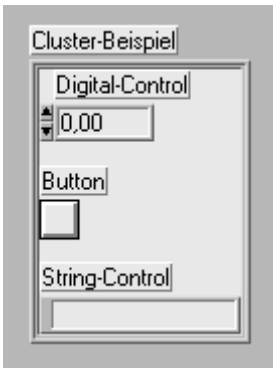
Beim folgenden Beispiel wird der eine Eingang der ADD-Funktion mit einem Skalar belegt, während am anderen Eingang ein Array liegt:



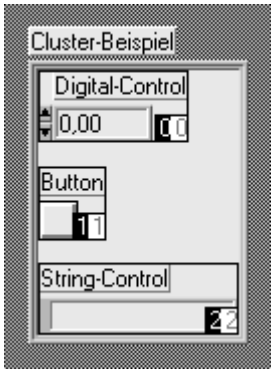
Ob gewisse Funktionen, die man verwenden will, sich polymorph im gewünschten Sinne verhalten, probiert man am besten aus.

Cluster

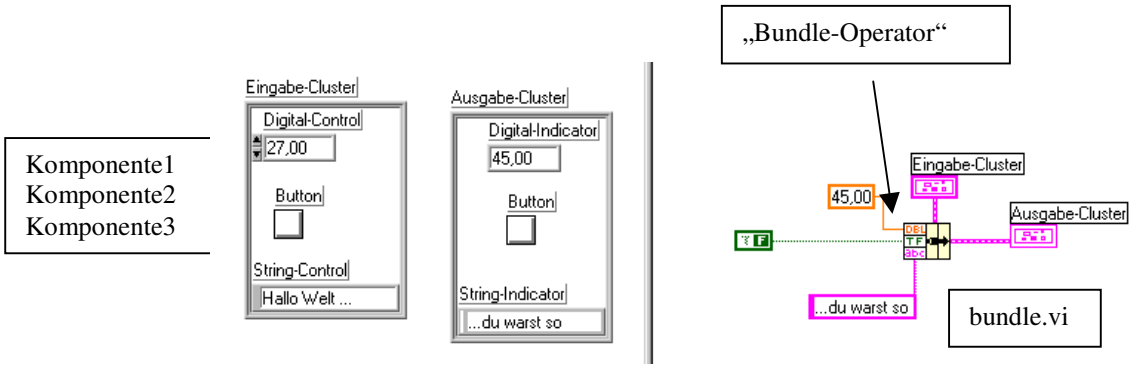
In Clustern werden unterschiedliche Datentypen zu einem neuen Datentyp zusammen gefaßt. Wie Arrays, wo es immer nur um einen Datentyp geht, werden Cluster dadurch erzeugt, daß verschiedenartige Objekte innerhalb des Clusters abgelegt werden. Dabei gilt die Regel, daß es sich dabei ausschließlich um „Controls“ oder „Indicators“ handeln darf, daß also in einem Cluster nicht „Controls“ und „Indicators“ nebeneinander plaziert werden können! Ein Cluster nimmt die Datenflußrichtung des ersten in ihm plazierten Objektes an.



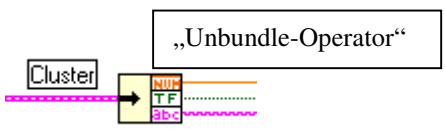
Clusterelemente haben eine logische Ordnung, die bei der Erzeugung des Clusters definiert wird: das erste eingefügte Element ist Element 0, das zweite eingefügte Element ist Element 1, usw. Elemente des Clusters können gelöscht werden, die Ordnung der Elemente kann geändert werden, wenn man mit der rechten Maustaste auf den Rand des Clusters klickt (s. rechts!). Auf die Elemente des Clusters wird über ihre Ordnung, nicht über ihren Namen zugegriffen.



Cluster werden u.a. dann eingesetzt, wenn es darum geht, eine große Zahl von Parametern an ein VI zu übergeben. Es ist zwar möglich ein VI mit bis zu 20 Eingängen zu versehen, es ist aber ganz offensichtlich, daß dann schnell sehr schwer zu lokalisierende Fehler auftreten können, weil die Drähte falsch angeschlossen werden.
Die Bündelung der Daten erfolgt diagrammseitig unter Verwendung des „Bundle-Operators“ (im Menü „Cluster“). Mit diesem Operator können entweder Daten zu einem neuen Cluster zusammengefaßt oder die Komponenten eines existierenden Bündels können mit neuen Werten belegt werden, wie im folgenden Beispiel vorgeführt wird.



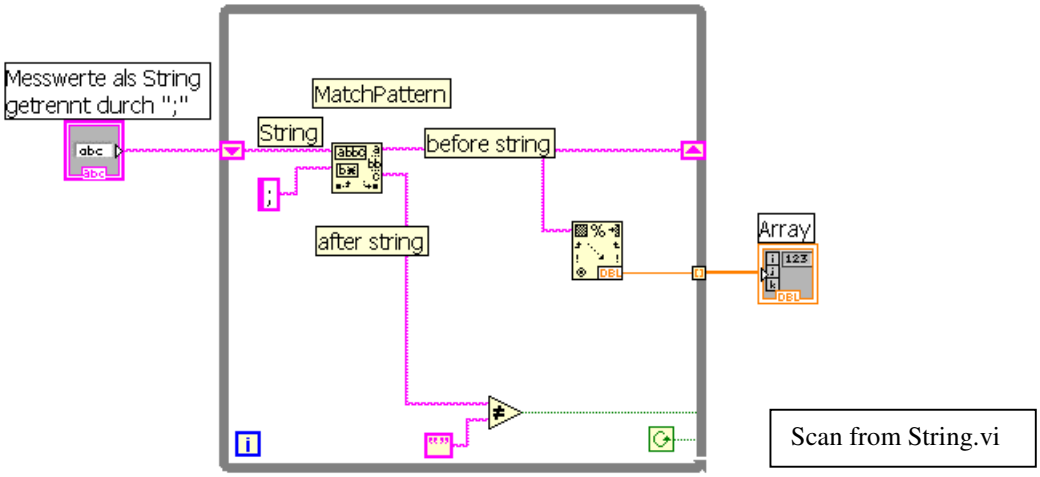
Die Zerlegung, die „Entbündelung“ eines Clusters spaltet den Cluster wieder in seine einzelnen Komponenten auf. Die Ausgangskomponenten werden entsprechend der Ordnung der Komponenten aufgeführt:



Aufgabe: Schreiben Sie ein Programm, das einen eindimensionalen Array umfüllt nach dem Prinzip: „Die letzten werden die ersten sein“.

Aufgabe: Schreiben Sie ein Programm, das 100 Würfe mit dem Würfel simuliert und dabei mitzählt, wie häufig jeder der 6 Werte gewürfelt wurde. Hinweis: Verwenden Sie ein Shift-Register für jeden der 6 Werte.

Aufgabe: Angenommen Sie erhalten eine Serie von Messwerten als String, in dem die einzelnen Messwerte durch Semikolon getrennt sind. Schreiben Sie ein Programm, das diese Messwerte in einem Array von reellen Zahlen ablegt. Überlegen Sie sich zunächst Ihr eigenes Programm und dann überlegen Sie, ob Sie etwas noch einfacheres, als das unten dargestellte Programm finden können.



Eine graphische Darstellung einer physikalischen Grösse erfordert nicht nur die eigentlichen Messwerte, sondern auch Angaben über den Beginn der Messung (den Startpunkt) und den Abstand zwischen zwei aufeinanderfolgenden Messungen (Schrittweite, Samplingrate, Abtastrate). D.h. dem graphischen Indicator-Element müssen diese Werte bekannt sein und die Hilfe zeigt an, wie diese Werte übergeben werden, nämlich als Cluster von Startwert x_0, Schrittweite Δx und Messwerte „y array“. Dies ist der Kontext in dem man zuerst auf Cluster stösst.

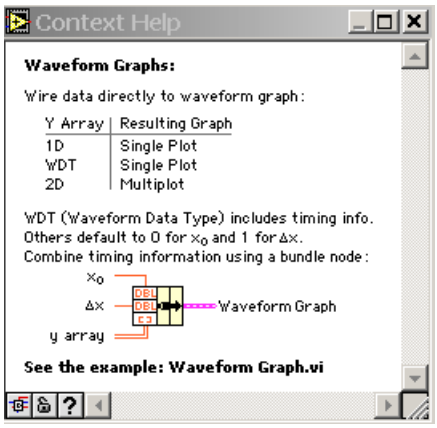
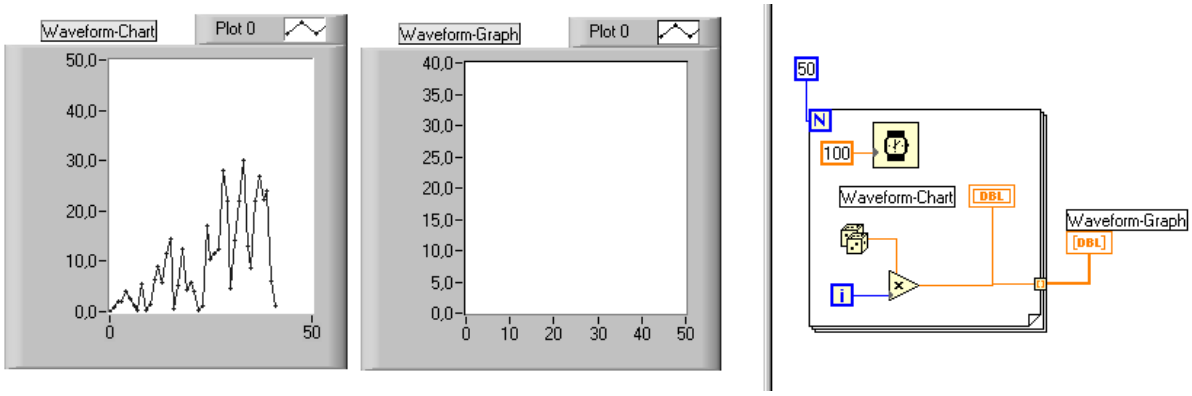
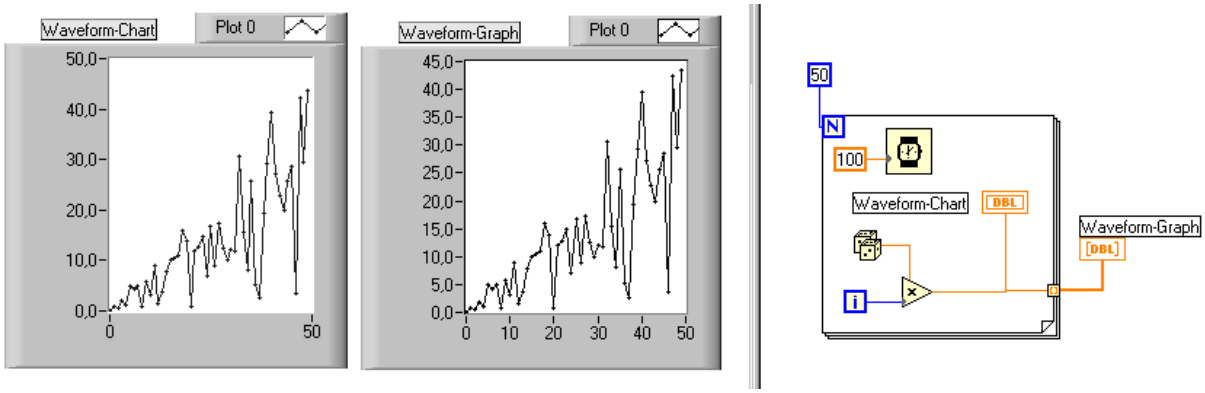


Diagramme und Kurven

Mit LabVIEW können Sie die bei einer Messung erfaßten oder die bei einer Simulation berechneten Werte in sehr bequemer Weise in graphischer Form in Form von Diagrammen oder Graphen darstellen. Der Unterschied dabei ist, daß Diagramme (Charts) jeden neu erfaßten oder berechneten Meßwert sofort darstellen, während bei Graphen zunächst die Gesamtheit aller Werte erfaßt oder berechnet werden und dann erst dargestellt werden.

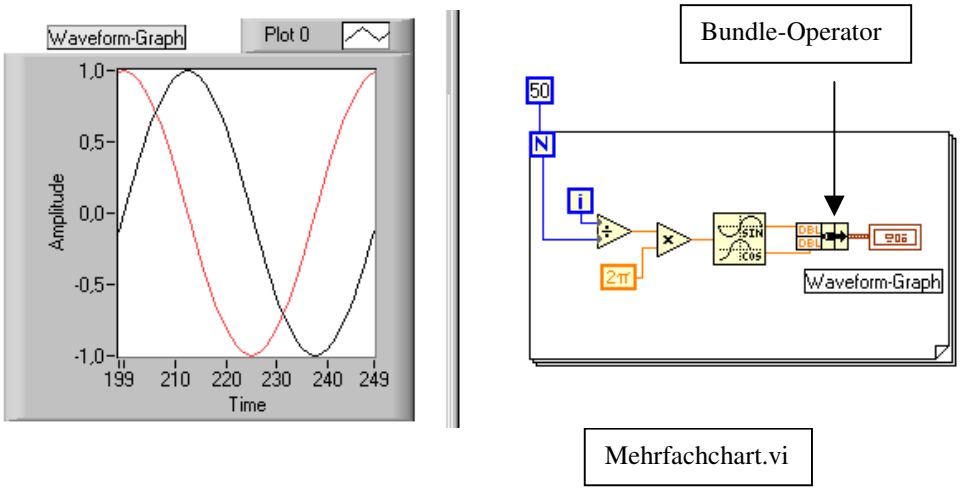


Oben wurde das Programm abgebrochen, so daß nur im Chart Punkte eingezeichnet sind und der Graph leer bleibt. Unten wurde abgewartet, bis das Programm zuende gelaufen ist: sowohl Chart als auch Graph zeigen (die gleichen) Werte an.



Charts haben drei verschiedene „Update-Modes“. Was darunter zu verstehen ist kriegen Sie am besten raus, wenn Sie mal mit dem Programm arbeiten.

In Charts, wie in Graphen können mehrere verschiedene Kurven parallel gezeichnet werden.
Bei Charts müssen dazu die Werte mit dem Bundle-Operator gebündelt werden.



Aufgabe: Schreiben Sie ein Programm, das die beiden Funktionen f(x) =1-x*x und g(x) = exp(-x*x) in einem Graphen darstellt. Für x soll gelten: -1 <= x <=2

Skalierung von Charts und Graphen

Um ein möglichst großes Bild zu erhalten skaliert LabVIEW die x-Achse und die y-Achse automatisch. Dies ist eine Option, die ausgeschaltet werden kann, indem Sie mit der rechten Maustaste auf den Graphen klicken.
Mit der „Loose Fit"-Option können Sie die Achsen mit „schöneren" Zahlen beschriften.
Mit der „Formatting..."-Option können Sie Gitter einzeichnen, den Skalenstil verändern, logarithmisch skalieren, u.v.a.m.

Die Legende

Hier können Sie den Punktstil des Graphen (Pünktchen, Kreuze, u.s.w.), Linienstil (gestrichelt, u.s.w.), Farbe, und den Interpolationsstil für jede darzustellende Kurve einzelnen festlegen. Eventuell müssen Sie zunächst erst mal wieder mit einem rechten Mausklick auf den Graphen klicken um diese überhaupt anzuzeigen.

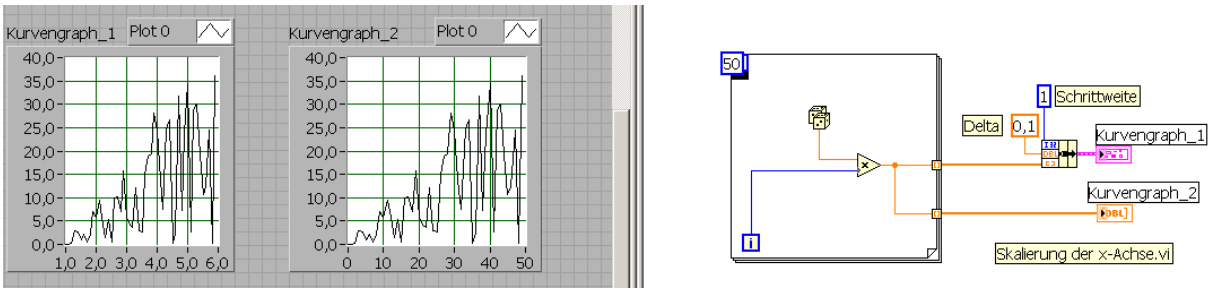
Die Palette

Damit können Sie Bildteile vergrößern („zoomen").
Aufgabe: Erzeugen Sie mit einem Programm einen Funktionsverlauf und üben Sie dann damit die verschiedenen „features" von LabVIEW, die eben beschrieben wurden.

Einfach- und Mehrfach-Plot-Graphen

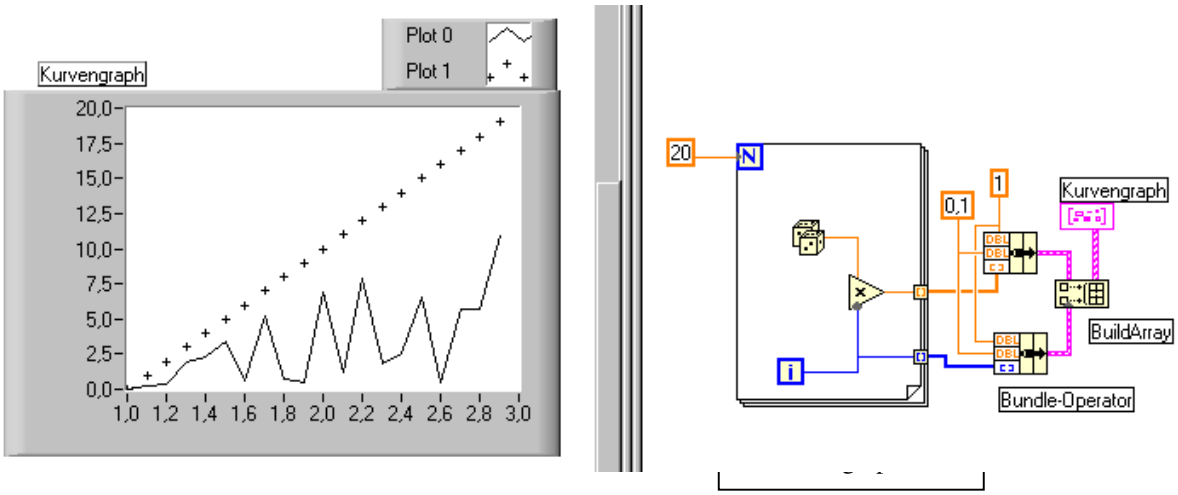
Wie schon gesagt: bei Graphen wird der Plot erst erstellt, wenn alle Meßpunkte erfaßt, bzw. berechnet sind. Im folgenden sehen Sie zwei Beispiele. Das zweite Beispiel stellt einfach eine Folge von 50 y-Werten dar: (y_n). Bei der Erstellung des Graphen wird dabei angenommen, daß zwei aufeinanderfolgende Abszissen immer den Abstand 1 haben und daß die erste Abszisse Wert 0 hat.

Im Beispiel unten wird zweimal dieselbe Folge von Meßpunkten erzeugt. Bei der Erstellung des Graphen wird aber bei Kurvengraph1 berücksichtigt, daß die Abszissen alle den Abstand „Delta“ haben und daß die erste Abszisse den Wert „Anfangswert“ besitzt.



Der obere Graph läuft in 0,1-er-Schritten von 1 bis 5,9, weil als Anfangswert 1 und als Schrittweite Delta = 0.1 übergeben wurden, während im unteren Graphen, der x-Bereich von 0 an in 1-er-Schritten bis 49 läuft.

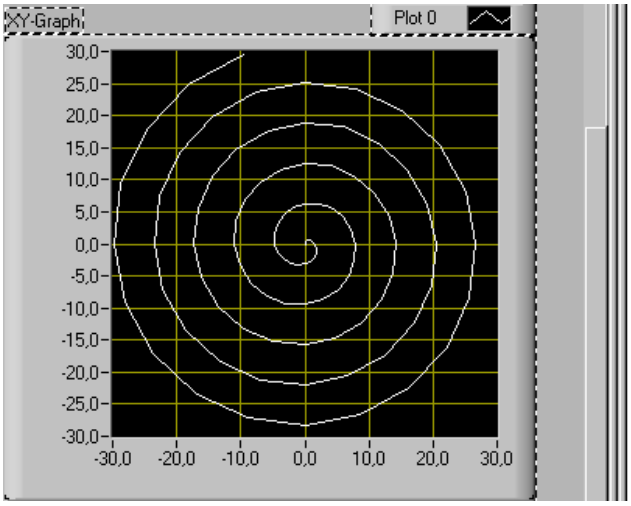
Einen „Mehrfach-Waveform-Plot“ erhalten Sie indem Sie ein Array der verschiedenen Plots aufbauen, dabei müssen wieder die beiden Fälle von oben unterschieden werden:

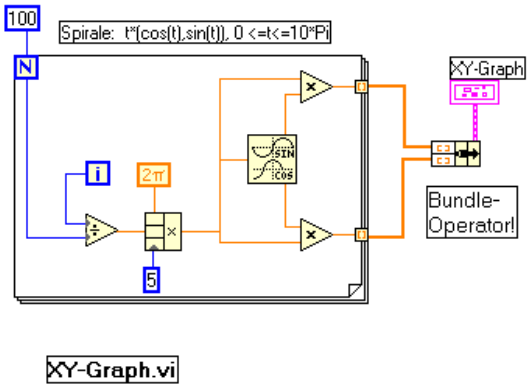


Im oberen Beispiel wird nur ein Array der y-Werte gebildet. Im unteren Beispiel müssen zuerst die beiden Strukturen (Cluster) bestehend aus {x0, delta, y-Werte} gebildet werden. In einem zweiten Schritt wird daraus ein Array gebildet, das dann an den Graphen übergeben wird. Analog geht man dann vor, wenn drei und mehr Graphen dargestellt werden sollen.

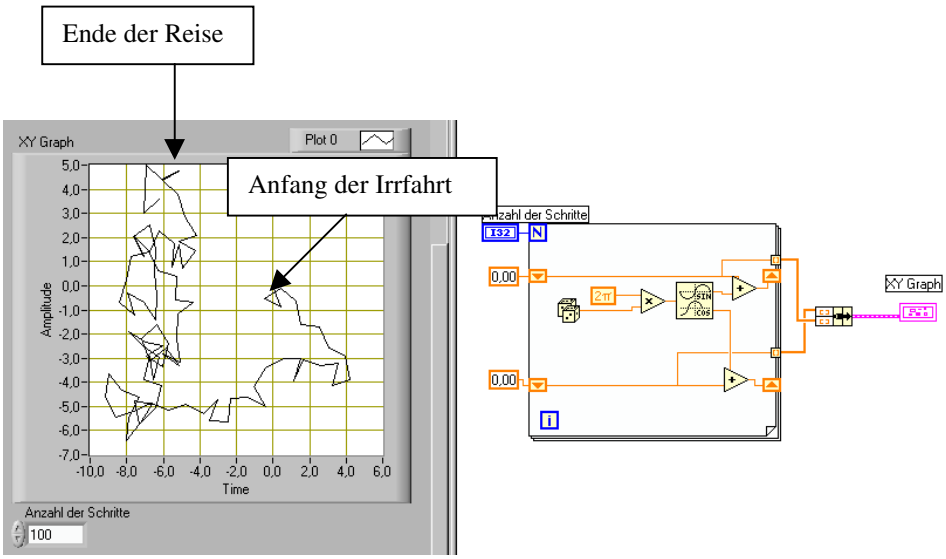
Einfach- und Mehrfach-Plot-XY-Graphen

Bei XY-Graphen müssen die x-Werte nicht äquidistant sein. Die x-Werte und die y-Werte müssen in diesem Fall zu einer Struktur gebündelt und dann an den Graphen übergeben werden: Bei Mehrfachplots muß für jeden Plot (mit „bundle“ die Struktur erzeugt werden und dann (mit Build Array) zu einem Array zusammengefasst werden, bevor sie zur Ausgabe übergeben werden. XY-Graphen eignen sich insbesondere zur Erzeugung von Parameterdarstellungen von Kurven in der Ebene.

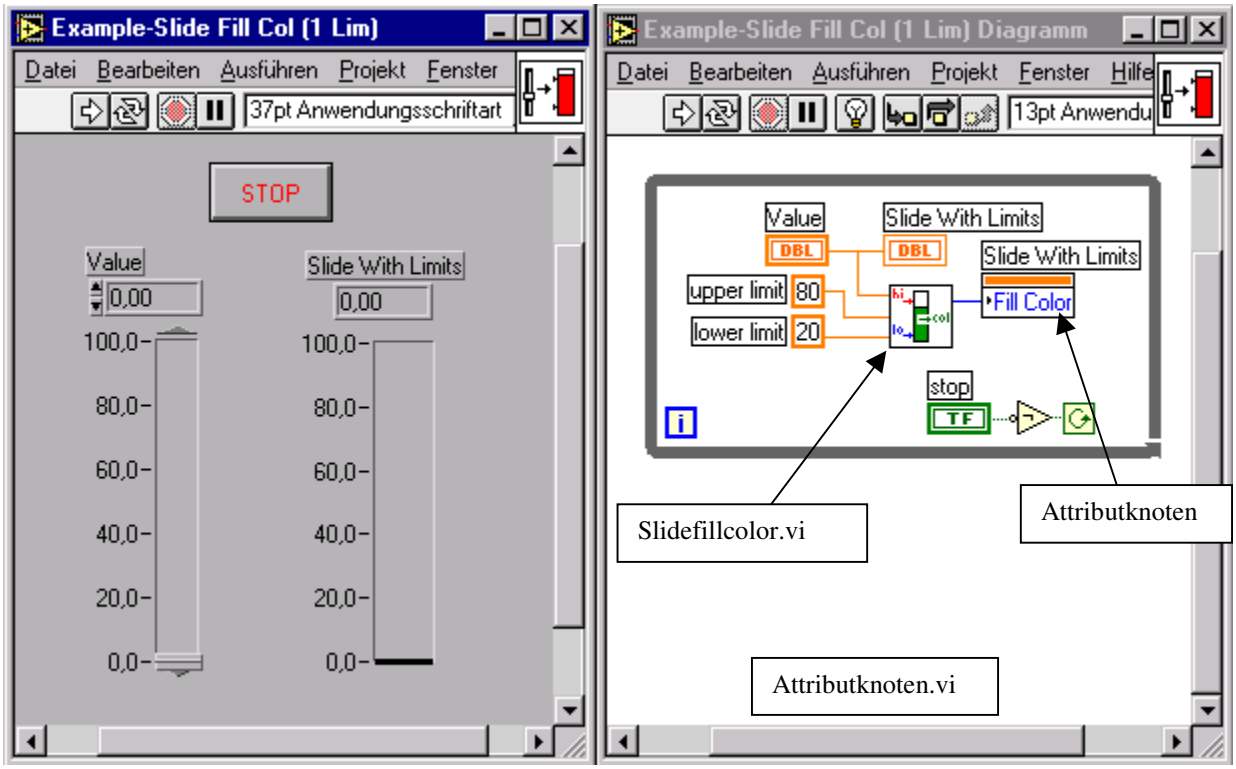




Aufgabe: Programmieren Sie eine Irrfahrt: Sie würfeln jede Sekunde eine Richtung (oben, unten, links, rechts) ,dann gehen Sie einen Schritt der Länge 1 in diese Richtung und würfeln dann eine neue Richtung, usw.
Unten sehen Sie ein Beispiel für eine Irrfahrt, in der die gewürfelte Richtung ein beliebiger Wert zwischen 0 und 2*π ist. Wie wäre das Programm, wenn der Irrfahrer nur die Möglichkeiten „Norden, Süden, Osten, Westen“ würfeln könnte?



Attributknoten zum Steuern von Bedienelementen und Diagrammen



Die in LabVIEW eingesetzten Anzeige- und Bedienelemente haben noch spezielle Attribute (Farbe, Position, Blinkmodus, Sichtbarkeit, usw.), die vom Programm selbst aus gesteuert werden können. Im obigen Beispiel sehen Sie zwei Schieberegler, einer im Eingabe- der andere im Anzeigemodus („Value“ und „Slide with limits“). Je nach Wert von „Value“, soll sich die Anzeigefarbe des rechten Reglers von blau nach grün nach rot ändern (denken Sie an die Kühlwassertemperaturanzeige in einem Auto). Auf der Diagrammseite sehen Sie den Attributknoten „Slide With Limits“ bei dem das Attribut „Fill Color“ ausgewählt wurde. Der Wert von „Fill Color“ wird vom VI Slidefillcolor.vi in Abhängigkeit vom aktuellen Wert und der oberen und unteren Grenze bestimmt und an das Attribut übergeben. Einen Attributknoten zu einem bestimmten Objekt erhält man über Rechten Mausklick/Erstelle/Attributknoten. Auf den im Diagramm erschienen Attributknoten geht man dann wieder über Rechter Mausklick/Objekt wählen zu dem konkreten Attribut, das beeinflusst werden soll.

Aufgabe: Ändern Sie das obige Programm so ab, daß der rechte Regler blinkend oder nicht blinkend dargestellt werden kann.

Aufgabe: Farben werden oft als Mischung von drei Grundfarben Rot, Grün, Blau dargestellt. Ein Farbwert einer Farbe setzt sich wie folgt aus drei Bytes zusammen. Das heisst man hat für jede Grundfarbe 256 Werte. (Wieviel Farbwerte insgesamt ergibt das?)

R	**R**	**R**	**R**	**R**	**R**	**R**	**R**	**G**	**G**	**G**	**G**	**G**	**G**	**G**	**G**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**

Wie kommt nun das Byte mit dem roten Farbwert an die richtige Stelle? Man muss es um 16 Bit nach links verschieben! Wie verschiebt man um 1 Bit nach links? Man multipliziert mit 2! Wie verschiebt man um 2 Bit? Wie verschiebt man also um 16 Bit? Wir finden also Farbwert = Rotwert*2^8 *2^8+ Grünwert*2^8+ Blauwert

Probieren Sie es aus mit LabVIEW. Sie benötigen Frontpanelseitig dazu eine Farbbox.

Erzeugung und Veränderung von Zeichenketten (Strings)

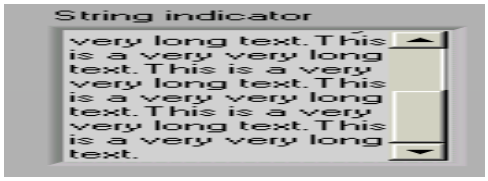
Viele computergesteuerte Geräte in der Messtechnik erwarten Steueranweisungen vom Rechner in Form von Zeichenfolgen („Set Range 10 mV“) und ebenso geben sie ihre Messergebnisse in Form von Zeichenfolgen an den Computer zurück. Daher steht man oft vor der Aufgabe Zahlen in Zeichenketten und umgekehrt Zeichenketten in Zahlen umzuwandeln.

Die Formatanweisung bei der Konvertierung von Zahlen in Strings ist denjenigen, die C kennen, sicher geläufig. Für diejenigen unter ihnen, die sie noch nicht kennen, hier das Wesentliche, das Sie besser verstehen, wenn Sie gleichzeitig, das nachfolgende Programm („formatundsoweiter.vi“) erzeugen um die jeweiligen Angaben gleich nachzuprüfen. Eine Formatanweisung in LabVIEW hat folgende Syntax:

[String]%[-][0][Stringbreite][.Stringgenauigkeit]Umwandlungskennung[String]

Ausdrücke in eckigen Klammern [ ] sind dabei optional, d.h. Sie können, müssen aber nicht, Bestandteil einer Formatanweisung sein. Da fast alles (bis auf „%“ und „Umwandlungskennung“ optional ist, heißt das zunächst: eine Formatanweisung muß mindesten ein %-Zeichen und eine Umwandlungskennung enthalten! Die folgende Tabelle erklärt Ihnen die einzelnen Elemente der obigen Syntaxregel:

Syntaxelement	Beschreibung
String	Zeichenfolge, die gewisse der unten beschrieben Zeichen enthalten kann
%	Zeichen, das die Formatspezifizierung einleitet
- (Bindestrich)(optional!)	Erzwingt Ausrichtung an der linken Kante
0 (zero)(optional)	Zeichen, das freien Raum links von der Zahl mit Nullen, statt mit Abständen („spaces) auffüllt
Stringbreite (optional)	Mindestbreite des Felds, das die konvertierte Zahl enthalten soll. Mehr Platz wird verwendet, falls nötig. Freier Platz wird von LabVIEW mit Spaces aufgefüllt und zwar links oder rechts der Zahl, je nach Ausrichtung. Bei fehlender Stringbreite wird so viel Platz wie nötig bereit gestellt.
.(Dezimalpunkt – Komma-)	Zeichen, das die Stringbreite von der Stringgenauigkeit trennt
Stringgenauigkeit	Zahl, die die Anzahl der Stellen rechts vom Dezimalpunkt festlegt, wenn die zu konvertierende Zahl eine Floating-Point-Zahl ist. Wenn keine Angabe zur Genauigkeit erfolgt, werden 6 Stellen ausgegeben.
Umwandlungskennung	Einzelnes Zeichen, das festlegt, wie die Zahl zu konvertieren ist: d Dezimal Integer x Hexadezimal Integer o Oktal Integer f Floating-Point-Zahl normal e,g Floating-Point-Zahl in wiss. Schreibweise Bemerkung: Diese Umwandlungskennungen können groß oder klein geschrieben werden.



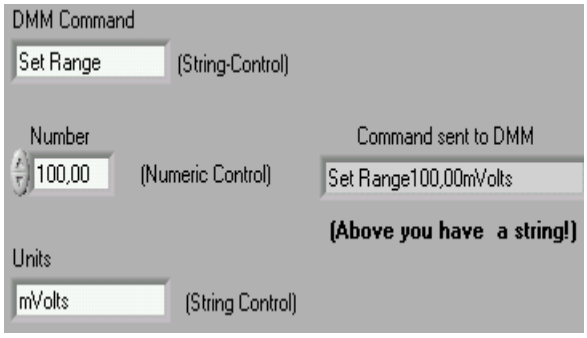
Ein String-Indicator oder ein String-Control kann übrigens auf die übliche Art und Weise vergrössert oder durch Klick auf den Rand auch mit einem Scroll-Balken versehen werden. Auch die Darstellungsform der Zeichen kann verändert werden (normal, Hexdarstellung, \Code und Passwortmodus)

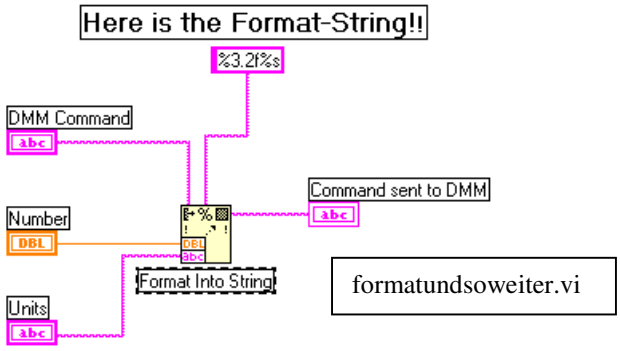
Aufgabe: Schreiben Sie ein kleines Programm, das in einem ersten Schritt die Eingabe von Username und Passwort erfordert.

Im Zusammenhang mit Zeichenfolgen gibt es drei wichtige Aufgaben:

- **Format Into String**: Aneinandersetzen und Formatierung von Strings und Zahlen zu einem Gesamtstring.
- **Scan From String**: Durchsuchen einer Zeichenfolge nach gültigen Ziffernfolgen (0...9, A...F, a...f und, nicht vergessen: das dezimale Trennungszeichen Komma oder Punkt, je nach dem, in welcher Welt (USA vs. Old Europe) Sie leben. Und diese Zeichenfolgen müssen dann noch in Zahlen (natürliche oder Floatingpointzahlen) konvertiert werden.
- **Match Pattern**: Darunter versteht man das Durchsuchen eines Strings, beginnend bei einem bestimmten Offset und die Aufteilung dieses Strings in drei Sub-Strings: den Substring vor dem gesuchten Muster, das Muster selbst und den Reststring danach.

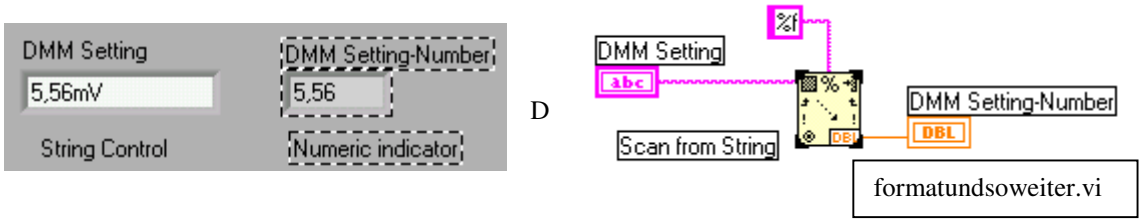
Unten sehen Sie ein Programmfragment, das zur Kommunikation mit einem digitalen Multimeter (DMM) dient. Um etwa den Messbereich auf 100 mV einzustellen erwartet das DMM einen String der Form: „SetRange100mVolts“. In LabVIEW bastelt man sich so einen String wie folgt zusammen:





Die wesentliche Routine im Diagramm ist dabei das Format Into String.vi, das Sie im String Sub-Menü finden. Mit einem Rechtsklick auf das VI kommen Sie in einen Edit-Modus, der Ihnen hilft den richtigen Format-String zu finden – damit können Sie dann die vorige Seite über die Syntax von Format-Anweisungen vergessen.

Nun geht es in die entgegengesetzte Richtung. Ihr DMM liefert Ihnen, nachdem die Messung durchgeführt worden ist, den Messwert in Form einer Zeichenfolge und Sie müssen daraus wieder eine Zahl machen, mit der man rechnen kann. Das für diesen Zweck zu verwendende VI heisst: Scan from String.vi. Experimentieren Sie damit!

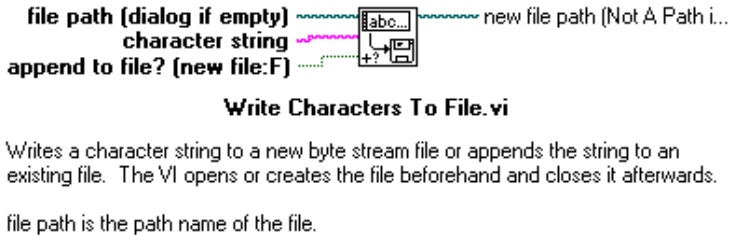


Ein- Ausgabe in Dateien

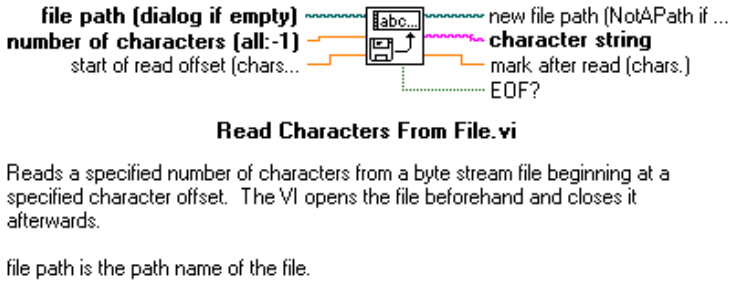
Die Funktionen, die benötigt werden, um Dateien auf die Festplatte zu schreiben oder von der Festplatte zu lesen finden Sie unter dem Menüpunkt „File I/O“. Wenn diese Funktionen keinen Pfadnamen zur betreffenden Datei erhalten wird dieser Pfad abgefragt. Die erzeugten Dateien sind gewöhnliche Textdateien, die mit jedem Editor einsehbar und bearbeitbar sind. Eine weit verbreitete Anwendung besteht darin, daß Dateien so abgespeichert werden, daß sie von einem Tabellenkalkulationsprogramm geöffnet werden können. Meistens werden in solchen Programmen Spalten durch Tabulatoren getrennt und Zeilen durch EOL’s („End of Lines“). Die beiden Funktionen „Write To Spreadsheet File“ und „Read from Spreadsheet File“ behandeln diesen Fall.

Aufgabe: Experimentieren Sie mit den im Folgenden beschriebenen VI’s. Vielleicht haben Sie Meßwerte aus dem AP oder dem PL, die Sie mit Hilfe eines Editors eingeben könnten, um Sie anschließend mit LabVIEW weiterzuverarbeiten?

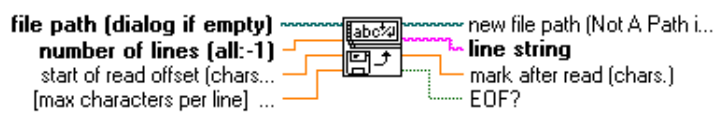
Die „Write Characters to File“-Funktion hat folgende Parameter:



Die „Read Characters from File“-Funktion hat folgende Parameter:



Die „Read Lines from File“-Funktion hat folgende Parameter:

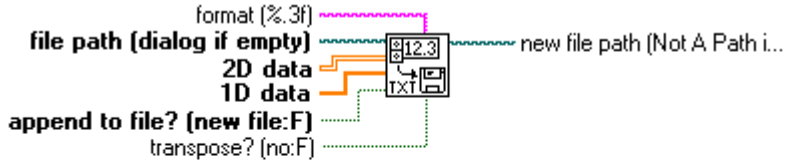


Read Lines From File.vi

Reads a specified number of lines from a byte stream file beginning at a specified character offset. The VI opens the file beforehand and closes it afterwards.

file path is the path name of the file.

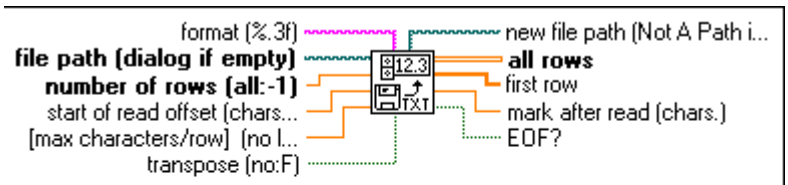
Die „Write to Spreadsheet File“-Funktion hat folgende Parameter:



Write To Spreadsheet File.vi

Converts a 2D or 1D array of single-precision numbers to a text string and writes the string to a new byte stream file or appends the string to an existing file. You can optionally transpose the data. This VI opens or creates the file beforehand and closes it afterwards.

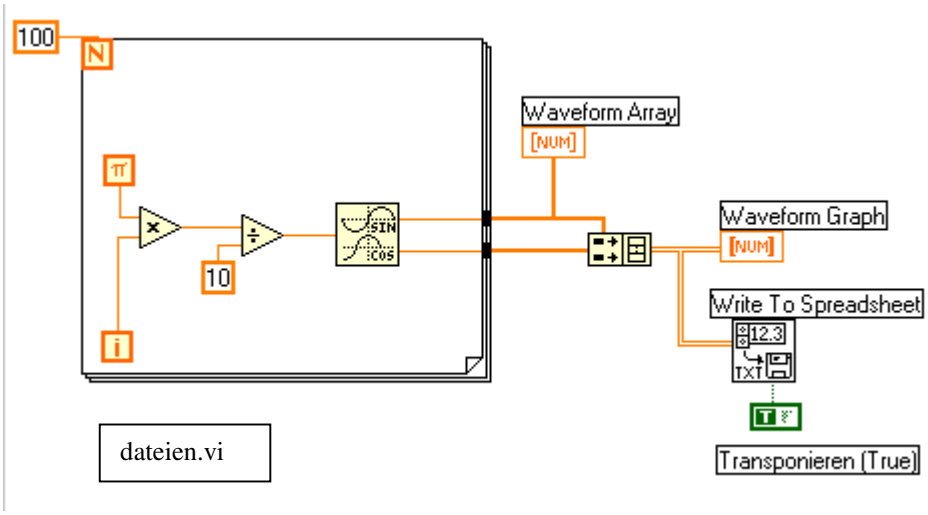
Die „Read from Spreadsheet File“-Funktion hat folgende Parameter:



Read From Spreadsheet File.vi

Reads a specified number of lines or rows from a numeric text file beginning at a specified character offset and converts the data to a 2D single-precision array of numbers. You can optionally transpose the array. The VI opens the file beforehand and closes it afterwards.

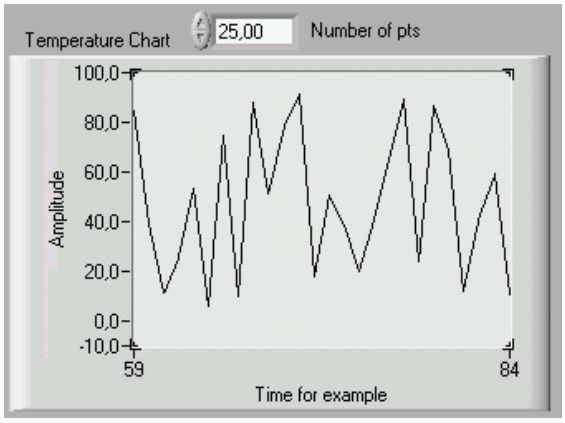
Aufgabe: Untersuchen Sie das und experimentieren Sie mit dem folgende(n) Programm

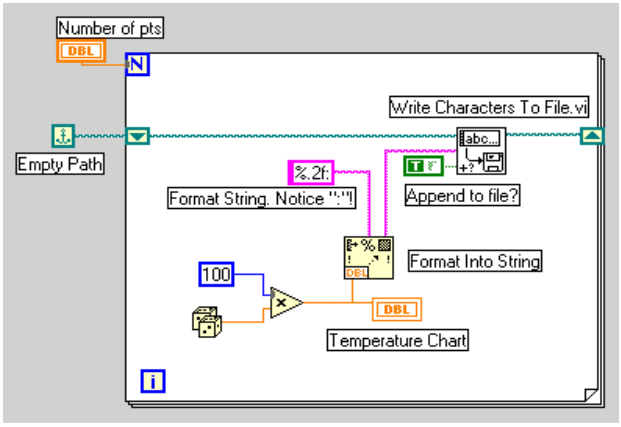


Öffnen Sie die dabei erzeugte Datei mit einem Tabellenkalkulationsprogramm und sehen Sie nach, was Sie erhalten haben. Was ändert sich, wenn die logische Konstante („Transponieren“) auf „false“ gesetzt wird? Schreiben Sie ein neues Programm, in dem Sie diese eben erzeugten Werte aus der Datei zurück in das Programm einlesen und graphisch darstellen. Ändern Sie das Programm so ab, dass zu jedem Messwert die laufende Nummer und die Einheit (z.B. „MegaWatt“) mit abgespeichert wird.

Schreiben von Messdaten in eine Datei

Angenommen Sie machen Temperaturmessungen und wollen eine Protokolldatei erzeugen, in der alle Ihre Messwerte enthalten sind. Das folgende einfache LabVIEW-Programm erledigt diese Aufgabe (nur die Temperaturwerterfassung darin wird mit dem Zufallsgenerator simuliert).



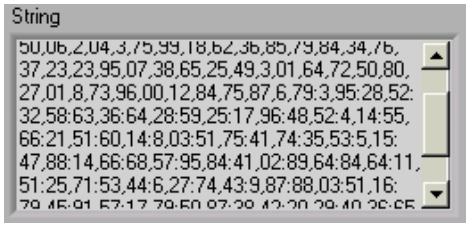


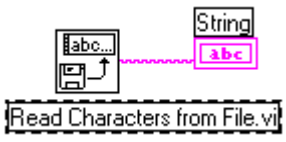
Verstehen Sie das Programm? Wozu dient der Aufwand mit den Shiftregistern? Stellen Sie sich vor, die Messwerte würden sehr oft gescannt (z. B. 10000 mal pro Sekunde), verschluckt sich dann der Rechner, bei der Aufgabe, die Messwerte auf die Platte zu schreiben?

Aufgabe: Zu jedem der oben erzeugten Messwerte soll auch noch Datum und Uhrzeit der Messung abgespeichert werden. Hinweis: Format Date/Time String im Time&Dialog Menüpunkt.

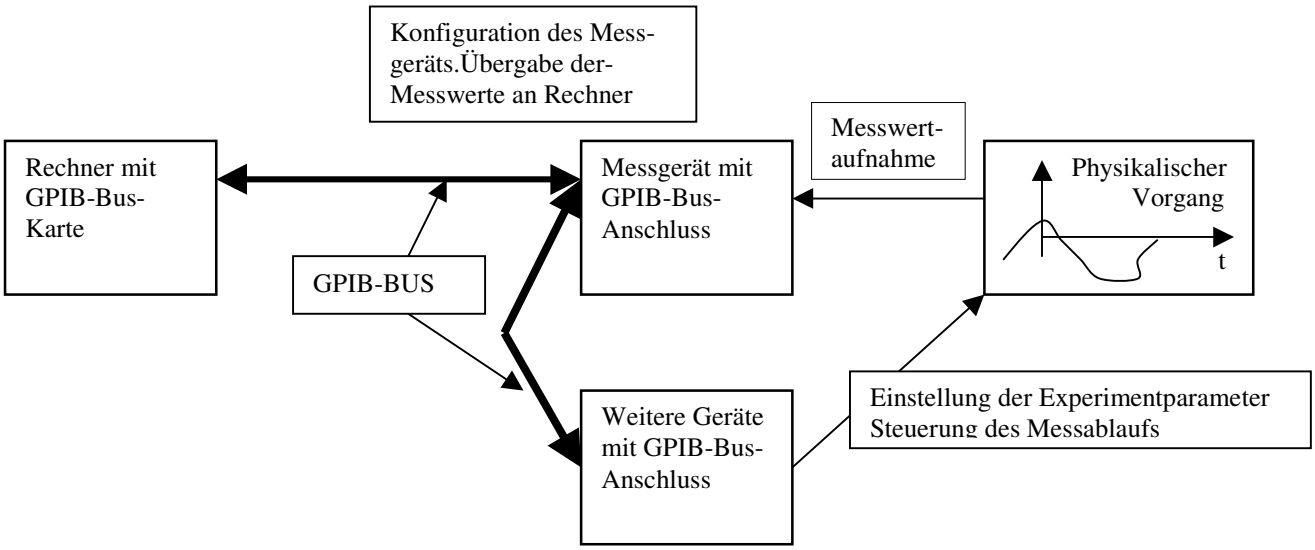
Lesen von Zeichen aus einer Text-Datei (ASCII-Datei)

Sie können das Read Characters from a File.vi verwenden. Das ganze Programm besteht aus einem einzigen VI!



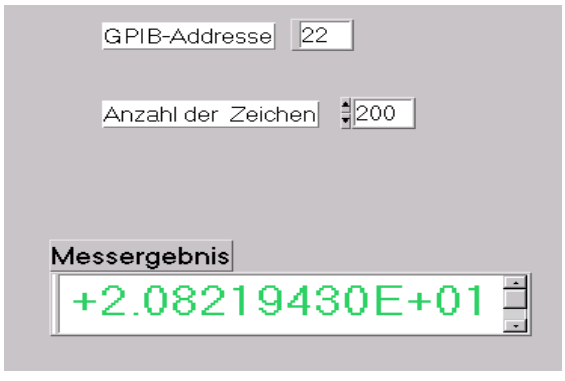


Der GPIB-Bus



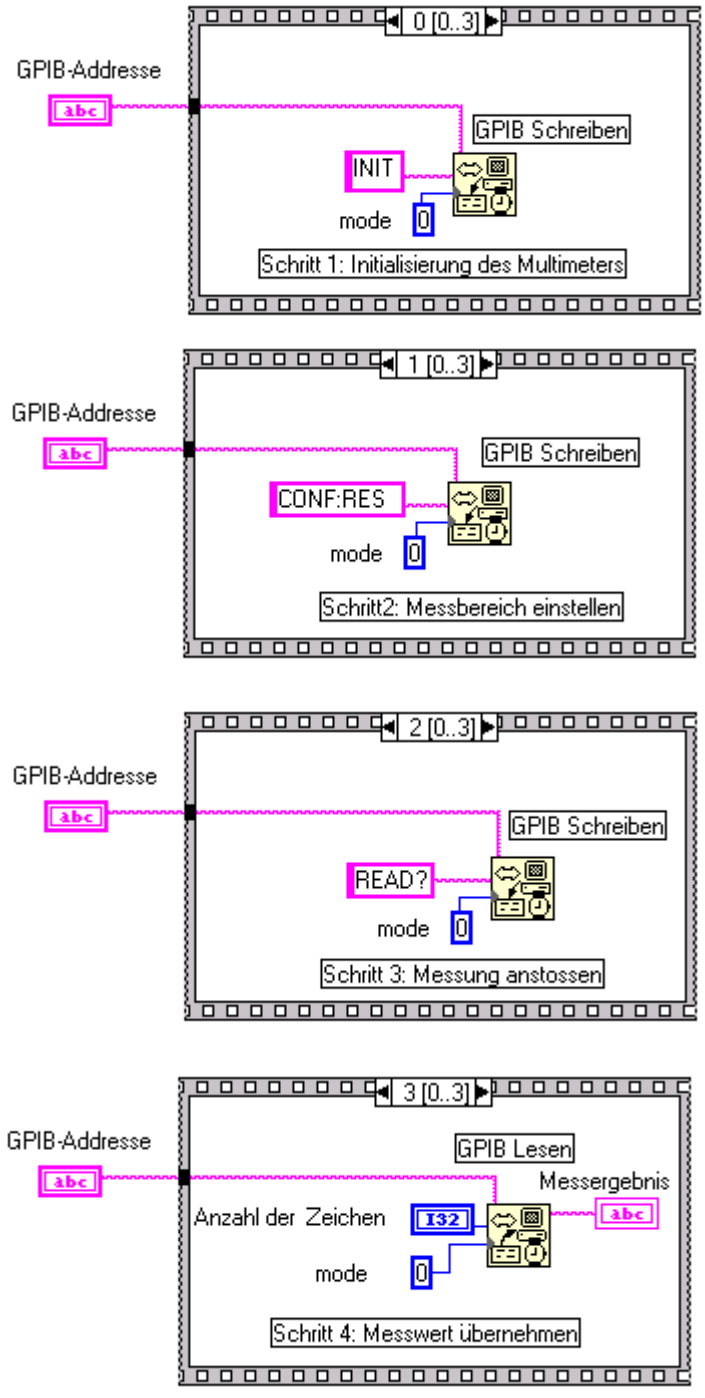
Zu Beginn der Messung muss das Messgerät konfiguriert werden. Die Experimentparameter müssen eingestellt und der Ablauf des Experiments muss kontrolliert werden. Die Messwerte des (zeitlich veränderlichen) physikalische Vorgangs, der erfasst werden soll, werden durch das Messgerät aufgenommen und über den GPIB-Bus an den Rechner übergeben. Jedes der angeschlossenen Geräte hat eine eindeutige GPIB-Adresse. GPIB (**G**eneral **P**urpose **I**nterface **B**us) ist ein von der Firma HP entwickelter Kommunikationsstandard, der für viele Messgeräte realisiert ist. Je nach Gerät müssen gewisse Kommandos zur Steuerung des betreffenden Geräts vom Rechner übergeben werden. Wie diese Kommandos heissen und was genau sie bewirken, muss dem zugehörigen Handbuch des Geräts entnommen werden. Dort gibt es auch oft Programmbeispiele für erste eigene Tests, aus denen die Bedeutung der Kommandos ebenfalls erschlossen werden kann.
Hier in diesem Skript soll nun in einem sehr einfachen Beispiel die Ansteuerung eines Multimeters (34401A) zur Widerstandsmessung beschrieben werden. Dem Kapitel „Externe Programmierung“ des Handbuchs kann entnommen werden, daß das Meßgerät dazu folgende Kommandos erwartet:

- INIT (Bedeutung: das Gerät wird in einen eindeutigen Grundzustand versetzt)
- CONF:RES (Bedeutung: der Widerstandsmessbereich wird ausgewählt)
- READ? (Bedeutung: die Messung wird angestossen)
- in einem letzten Schritt wird der Messwert über den GPIB-Bus an den Rechner übergeben.



Dieser Ablauf wird in LabVIEW in einer vierstufigen Sequenz realisiert. Links sehen Sie das Frontpanel des Programms. Es erlaubt, die GPIB-Adresse des zu bedienenden Geräts einzugeben. Durch die „Anzahl der Zeichen“ wird festgelegt, wieviele Zeichen maximal vom Gerät an den Rechner übergeben werden sollen. In der String-Control-Anzeige unten wird das Messergebnis dargestellt.

Das Programm besteht wie schon gesagt aus vier Schritten, die als Sequenz realisiert werden. Es ist sehr einfach und es lohnt sich nicht darüber Worte zu verlieren, denn klarer kann ein Programm nicht beschrieben werden.



DAQ - Data AcQuisition – Datenerfassung per Einbaukarte

Das Thema dieses Abschnitts ist Datenerfassung per Einbaukarte, (entweder eine der bekannten Erweiterungskarten für PC's oder eine PCMCIA-Karte für Laptops) mit Funktionen wie: Analog/Digitalwandlung, Digital Input/Output, Zählung und Triggerung. Wenn im Zusammenhang mit Messdatenerfassung von „DAQ" gesprochen wird, dann ist damit die eben erläuterte Vorgehensweise gemeint. Ein DAQ-System besteht aus vier Teilen:

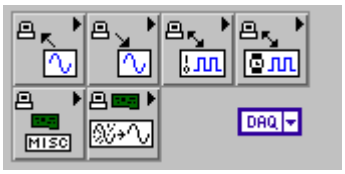
- Rechner
- Einbaukarte
- Sensoren, Transducer: Vorrichtungen, die physikalische Grössen (Kraft, Druck, Geschwindigkeit, Feldstärke, Temperatur, u.v.a.m. in eine elektrische Spannung übertragen
- Software

Eine DAQ-Einbaukarte kann folgende Funktionen haben:

- ADC (**A**nalog-to-**D**igital **C**onversion, konvertiert elektrische Spannung in eine Zahl (Input)
- DAC (**D**igital-to-Analog Conversion konvertiert eine Zahl in eine elektrische Spannung (Output)
- Digital Input (sieht nach, ob ein Schalter geschlossen oder offen, eine Lampe an oder aus ist)
- Digital Output (setzt einen Schalter, Relais, LED)
- Timing (legt die Erfassungsrate (sampling rate) fest, die Anzahl der Messungen pro Sekunde)
- Triggering (triggert ein Ereignis, eine Messung)

Es gibt noch eine grosse Anzahl weiterer Probleme, wie Auflösung, Rauschen, Störspannungen, Aliasing, Abtasttheorem von Shannon, usw., die bei der Messdatenerfassung per DAQ überlegt werden müssen, aber das sind keine spezifischen LabVIEW-Probleme, trotzdem müssen Sie sich auch darum kümmern. Wir konzentrieren uns nun auf die LabVIEW-spezifischen Vorgehensweisen und die Vis, die LabVIEW für DAQ zur Verfügung stellt. Sie finden sie in der Subpalette **Data Acquisition** in der **Functions-Palette**. Dieser Teil ist in sechs Unterpunkte gegliedert, deren Bedeutung an den Symbolen schon erkennbar ist:

- Analog Input
- Analog Output
- Digital I/O (**I**nput/**O**utput)
- Counter
- Calibration and Configuration
- Signal Conditioning, Signalaufbereitung



In diesem Kurs beschränken wir uns auf die drei ersten Punkte. Für jeden dieser Punkte gibt es wieder ein ganzes Spektrum von Unterfunktionen: Easy I/O, Intermediate (Mittelschwer), Tools (Hilfsmittel) und Advanced (Fortgeschrittene).

Der ideale Einstiegspunkt in DAQ per LabVIEW sind die **Easy I/O VIs** mit deren Hilfe grundlegende Ein-/Ausgabefunktionen in LabVIEW studiert werden können.

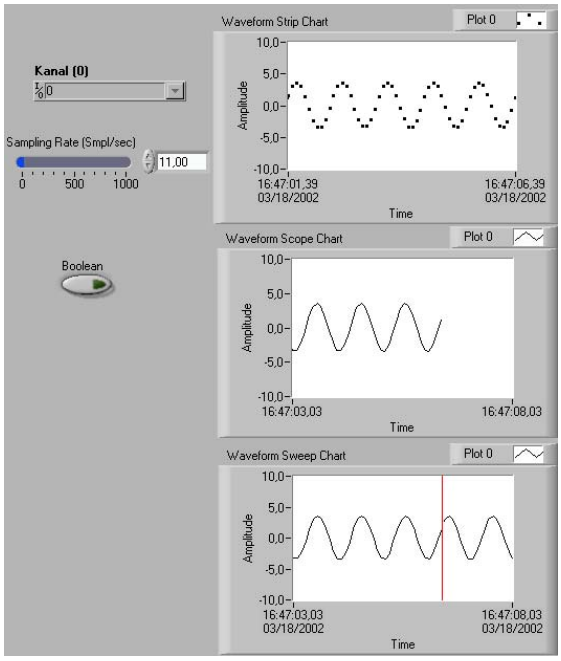
Die **Intermediate VIs sind flexibler und stellen mehr Funktionen bereit** (etwa externe Zeitvorgabe und externes Triggern von Vorgängen).

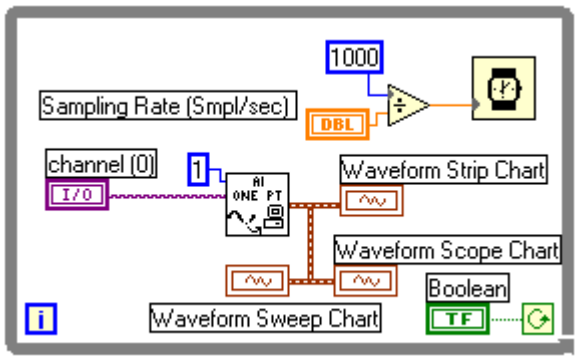
Die **Advanced VIs** bewegen sich auf sehr niedrigem (daher schwierigem) Niveau. Sie werden nur in sehr seltenen Fällen benötigt.

Analog Input

Single-Point Acquisition

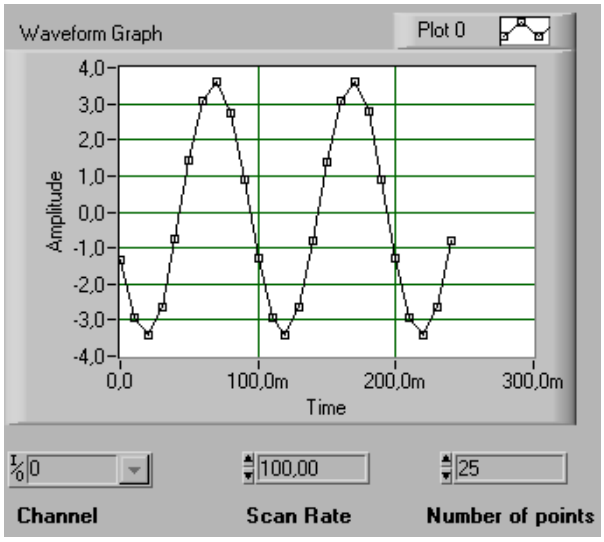
Es wird nur ein einziger Wert auf dem bezeichneten Kanal gelesen und sofort an das aufrufende VI zurück gegeben. Diese Art der Erfassung ist sinnvoll, wenn man ein „fast konstantes“ Signal erfassen will, z. B. wenn man die Temperatur an einer gewissen Stelle überwachen will.

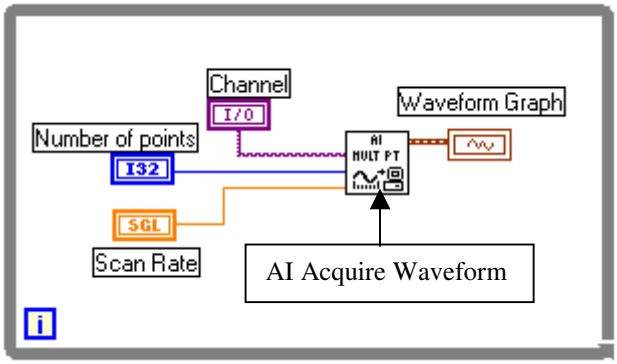




Waveform Acquisition – Erfassung eines ganzen Kurvenverlaufs

Wenn Sie gleich einen ganzen Array von Messwerten erfassen wollen, die zu gleichmässig verteilten Zeitpunkten gemessen werden, können Sie das AI Acquire Waveform.vi verwenden. Es verlangt als Eingangsparameter die Abtastrate und die Anzahl der zu messenden Werte. Der Ausgang ist das Feld der gemessenen Werte.

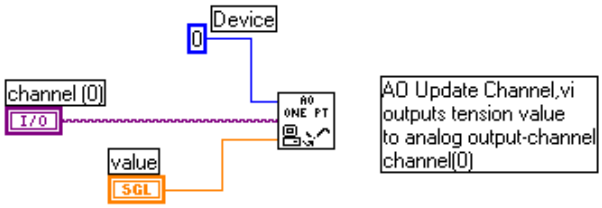




Analog Output

Wie beim Input gibt es hier wieder generell zwei Methoden: entweder wird ein einzelner Punkt/Wert zu einem gewissen Zeitpunkt ausgegeben, oder ein ganzer Array von Werten, ein Wert nach dem anderen. Sie stutzen? Das ist gut so, denn natürlich muss die Hardware auch noch wissen, wie gross der Zeitabstand zwischen zwei aufeinanderfolgenden Werten sein muss! Die zweite Methode sollte auf jeden Fall angewendet werden, wenn die Zeitabstände gering sind (d.h. viele Werte pro Sekunde), und wenn es auf die Genauigkeit der Zeitabstände ankommt, denn diese Genauigkeit wird in diesem Fall von einer Hardware-Uhr kontrolliert.

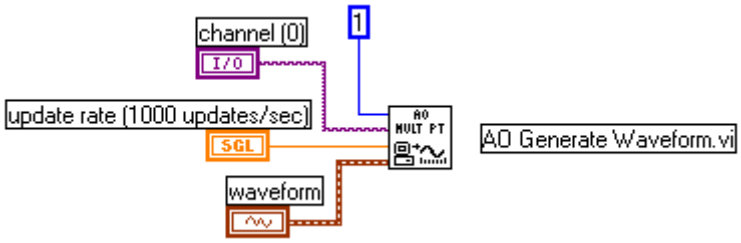
Single-Point Generation



Das **AO Update Channel.vi** schreibt einen vorgegebenen Wert auf einen analogen Ausgabekanal. Dieses VI hat zwei Eingangsparameter: Channel – ein String, der den Ausgabekanal festlegt und value, der Wert der Spannung, die ausgegeben werden soll.

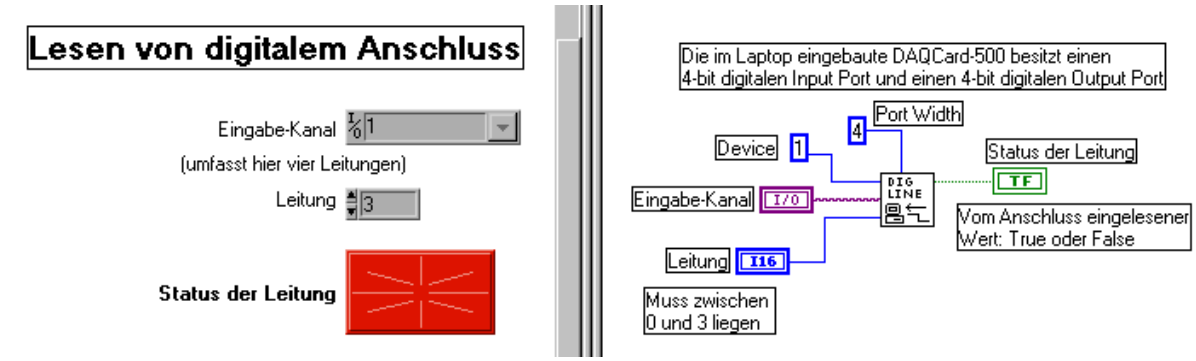
Waveform Generation

Wenn Ihr DAQ-System als Signalgenerator arbeiten soll, müssen Sie genau die Ausgaberate kontrollieren können, d.h. die Anzahl der Werte, die pro Sekunde ausgegeben werden sollen, bzw. der Zeitabstand zwischen zwei aufeinanderfolgenden Werten. Dies muss hardwaremässig geschehen, denn es ist damit zu rechnen, dass Ihr Betriebssystem die Schleifen in Ihrem Programm, die die Ausgabe erledigen sollen stört, weil ausser der Messung auch noch andere Aufgaben erledigt werden müssen. Sie sollten also auf jeden Fall das VI **AO Generate Waveform.vi** verwenden!



Digital I/O

Digitale I/O-Komponenten erzeugen, bzw. überwachen boolesche An-Aus-Signale zur Kontrolle, bzw. Steuerung peripherer Geräte (Lampen, Relais: eben alles, was entweder an- oder ausgeschaltet sein/werden kann). Diese einzelnen Leitungen, werden üblicherweise zu Ports, bestehend aus 4 oder 8 Anschlüssen, zusammen gefasst. In einem Port sind alle Anschlüsse entweder Ein- oder Ausgabe-Anschlüsse. Im folgenden Beispiel wird eine einzelne Leitung eines Ports nach ihrem Wert abgefragt.



Hier wird eine digitale Leitung gesetzt (z.B. eine LED (Leuchtdiode) angeschaltet):