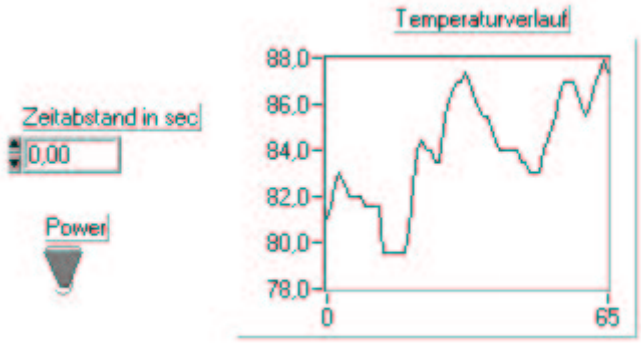
LabVIEW für Anfänger[1]

Inhaltsverzeichnis

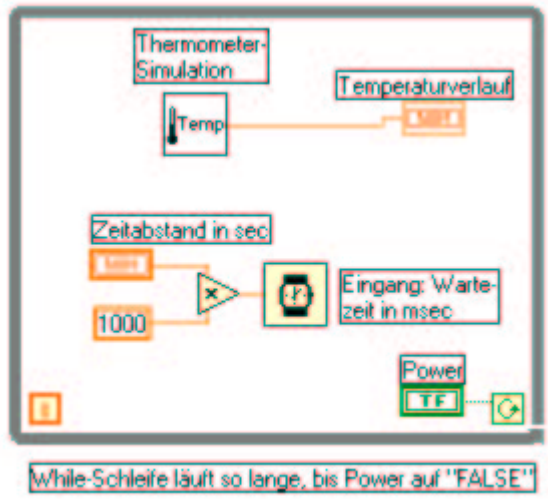
[1] Skript für den PC−Pool−Ferienkurs "LabVIEW für Anfänger" Version 5 (02/02, Blersch)

Was ist LabVIEW?

LabVIEW ist eine Programmierumgebung, mit der Sie effizient und ökonomisch Programme zur Steuerung von Geräten und zur Meßdatenverarbeitung erstellen können. Eine große Zahl vorgefertigter, in der professionellen Version enthaltener 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. (Auf jeden Fall ist es erheblich einfacher, LabVIEW−Programme, die andere Leute geschrieben haben, zu verstehen, zu verbessern und zu erweitern).
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 Temperatur 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:

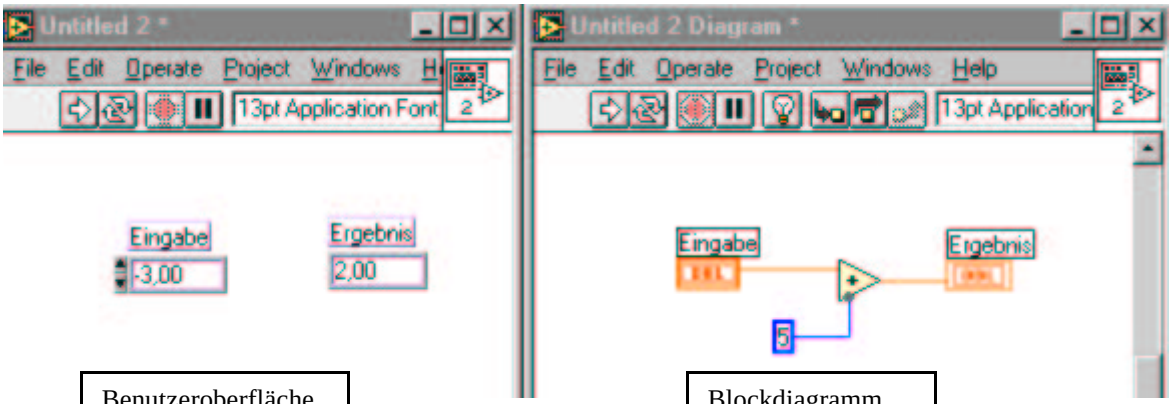


LabVIEW enthält eine große Anzahl von Funktionen, die speziell im Bereich der Meßdatenerfassung und Meßdatenverarbeitung benötigt werden, d.h. in Bereichen, die insbesondere in der experimentellen Physik von großer Bedeutung sind. Natürlich können Sie mit diesem Programm auch Daten weiterverarbeiten, die nicht

mit LabVIEW erfaßt wurden. LabVIEW kann weiterhin sehr gut bei der Simulation gewisser physikalischer Phänomene eingesetzt werden.
Bemerkung zu diesem Skript: es gibt 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.

Benutzeroberfläche und Blockdiagramm

Nach dem Öffnen von LabVIEW erhält man zwei Fenster: Benutzeroberfläche (Front Panel) und Blockdiagramm. Die Benutzeroberfläche ist das Fenster, in dem der Benutzer die Daten eingibt, die an das Programm und eventuell an die Geräte weiter gegeben werden und wo auch die Meßdaten (z.B. in Form von Tabellen oder Graphen) erscheinen. Hinter dem "Front−Panel" versteckt sich das Programm, das in der LabVIEW−Sprache "Blockdiagramm" genannt wird, darstellt. Dieses Blockdiagramm − obwohl es wie eine Grafik aussieht − **ist** das Programm. Die Zusammenfassung von Panel und Blockdiagramm heißt "VI", d.h. "Virtuelles Instrument". Im Normalfall wird ein Blockdiagramm selbst wieder aus einer Reihe von Sub−Programmen oder Sub−Vi's aufgebaut sein. Im Unterschied zu einem herkömmlichen Programm erfolgt der Datenfluß in LabVIEW über Drähte, die die Ein− und Ausgänge der SubVI's auf der Diagrammseite miteinander verbinden. (Bei den klassischen Programmiersprachen erfolgt dieser Datenfluß über die Parameterübergabe.)
Aufgabe: "Tippen" Sie das folgende Programm ab und bringen Sie es zum Laufen:



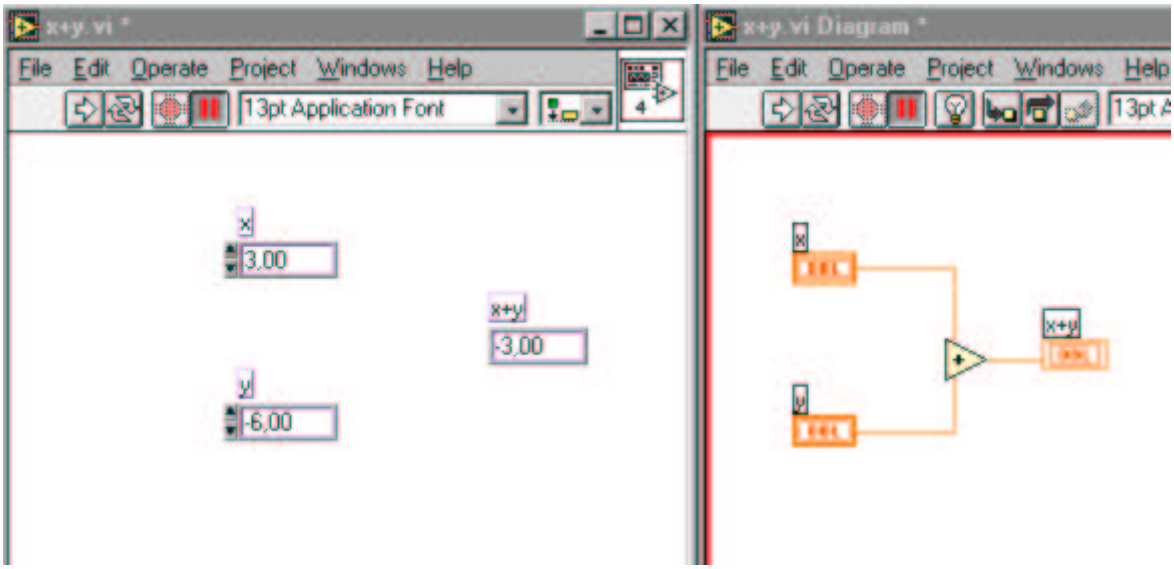
Benutzeroberfläche

Blockdiagramm

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.

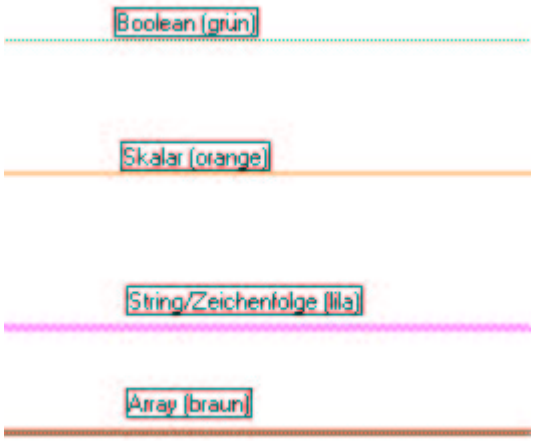
Terminals, Knoten und Drähte

Immer wenn Sie auf der Panelseite ein Ein− oder Ausgabeobjekt plazieren, 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 Datenkugeln aus den Datenquellen herauskommend, die Drähte entlang laufen, um schließlich in den Senken am Drahtende zu verschwinden.



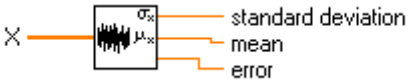
Ein *"Knoten"* ist ein "Programmausführungselement" − er entspricht in klassischen Programmen einem Operator oder einer Subroutine. Wie im klassischen Fall werden Werte/Parameter übergeben und andere Werte zurückgegeben. Einfaches Beispiel: der Plus−Operator im obigen Beispiel ist ein Knoten mit zwei Eingabeparametern und einem Rückgabeparameter.

Ein *"Draht"* ist ein Datenpfad zwischen einem Eingabe−Terminal und einem Ausgabe−Terminal. Drähte sehen verschieden aus, 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

Wie jedes anständige (d.h. gut strukturierte) Programm aus Subroutines aufgebaut wird, wird jedes VI aus SubVI's aufgebaut. Ein SubVI wird als Icon dargestellt, einer graphischen Darstellung mit Ein− und Ausgängen, sogenannten "Connectors", zu deutsch "Anschlüssen", mit deren Hilfe die Ein− und Ausgabeparameter übergeben werden können. LabVIEW stellt dem Benutzer eine große Anzahl SubVI's bereit. So muß man natürlich zur Berechnung der Standardabweichung kein eigenes VI mehr programmieren, sondern kann auf das entsprechende VI von LabVIEW zurückgreifen:



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

Aufgabe: a) In LabVIEW gibt es im Menü Arithmetik einen einfachen Zufallsgenerator. Plazieren Sie eine digitale Anzeige auf dem Panel und lassen Sie sich einige Zufallswerte anzeigen.
b) Nun soll ein ganzer Strom von Zufallszahlen erzeugt und grafisch dargestellt werden. Verwenden Sie dazu panelseitig einen sogenannten "Waveform−Chart" und einen Schalter und im Diagramm eine "while−Schleife" aus dem Menüpunkt "Structs & Constants" und den in a) studierten Zufallsgenerator.

Aufgabe: Gegeben sind zwei numerische Eingaben "a" und "b". Je nachdem, ob gilt: "a > b", "a = b" oder "a < b" soll eine runde rote Lampe, eine große ovale grüne Lampe oder eine viereckige blaue Lampe aufleuchten.

Aufgabe: Mit einem Textring (Menü Numeric) können Texten Zahlen zugeordnet werden. Schreiben Sie ein kleines Testprogramm mit einer Textringeingabe und einer numerischen(!) Ausgabe.

Nach einiger Zeit wird es störend, wenn man für 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> − 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 erstellten Testprogramm.

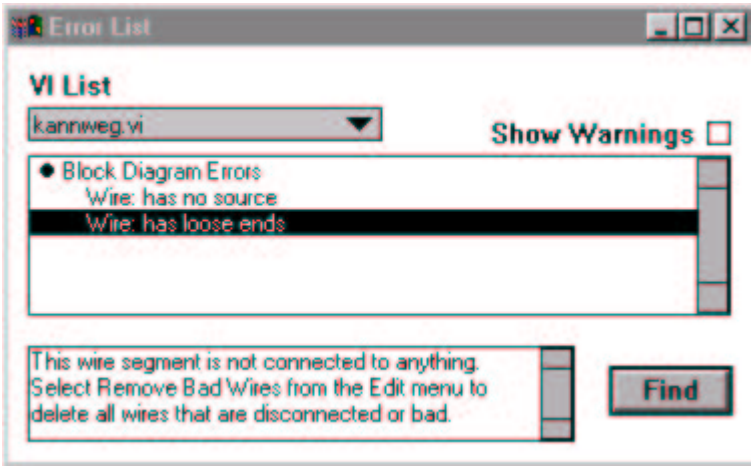
Abspeichern und Einlesen von Programmen

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".

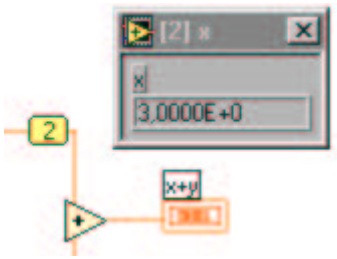
Fehlersuche, Debugging

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", d.h. im "Schritt−für−Schritt−Verfahren" 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 sogenannte Breakpoints zu setzen. Das sind Punkte an denen das Programm anhält, so daß man in Ruhe die zum Zeitpunkt des Breakpoints berechnet Werte untersuchen kann. Einzelheiten dazu finden Sie im Handbuch! Ein Fehler der mich bei der Erstellung dieses Skripts einen halben Tag gekostet hat, bestand darin, daß ich ein SubVI (das "Curve Fit VI) in meinem Diagramm ohne Anschlüsse auf einem nicht sofort einsehbaren Bereich des Diagramms vergessen hatte. Darauf kamen andauernd mysteriöse Fehlermeldungen, solange, bis ich aus Versehen entdeckte, daß es da noch dieses vergessene VI gab.

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 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 dann 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!

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.

Dritter Schritt: Dokumentation. Im Windows−Menü gibt es einen Unterpunkt "Show VI Info ..." mit dessen Hilfe Sie den Dokumentationstext (kurz und prägnant!) eintragen können.

Aufgabe: Erstellen Sie ein SubVI für die Berechnung von

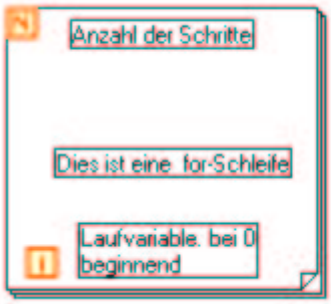
$$\sqrt[3]{x_1 + x_2 + x_3}$$

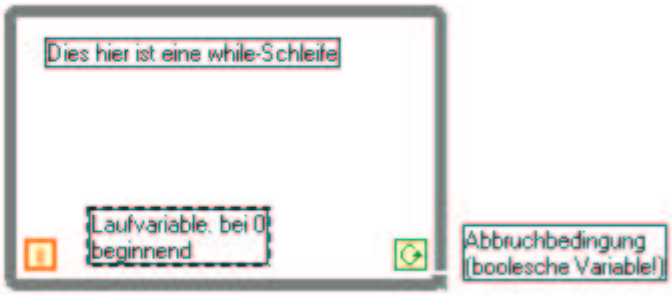
Aufgabe: Erstellen Sie ein SubVI, das zwei beliebige Zahlen dividiert! Wenn der Nenner den Wert 0 hat, soll eine rote Lampe aufleuchten. Dazu benötigen Sie eine Case−Abfrage, die Sie im Menü Strukturen finden können.

Programmstrukturen

Wie bei klassischen textbasierenden Programmiersprachen benötigen wir auch in LabVIEW Kontrollstrukturen, die den Ablauf des Programms 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 enthält, genau N−mal aus. Die Variable N muß gesetzt werden, indem eine numerische 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 also äquivalent zu folgendem C−Code:

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

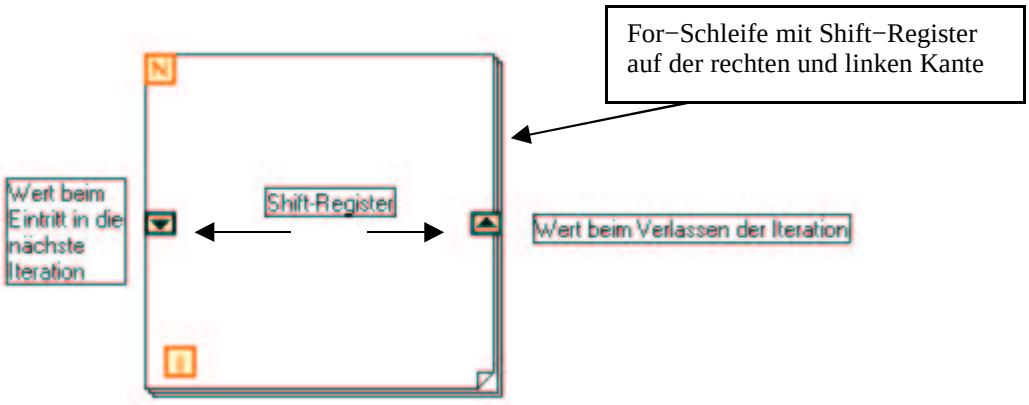
Die while−Schleife führt die in ihr enthaltenen Programmteile so lange durch, wie die Abbruchbedingung wahr ist. An die Abbruchbedingung muß also ein logischer Ausdruck angeschlossen werden. Wenn nichts angeschlossen wird, wird der logische Wert "false" angenommen, was zur Folge hat, daß die Schleife dann genau einmal ausgeführt wird.

Die while−Schleife ist äquivalent zu folgendem C−Code:

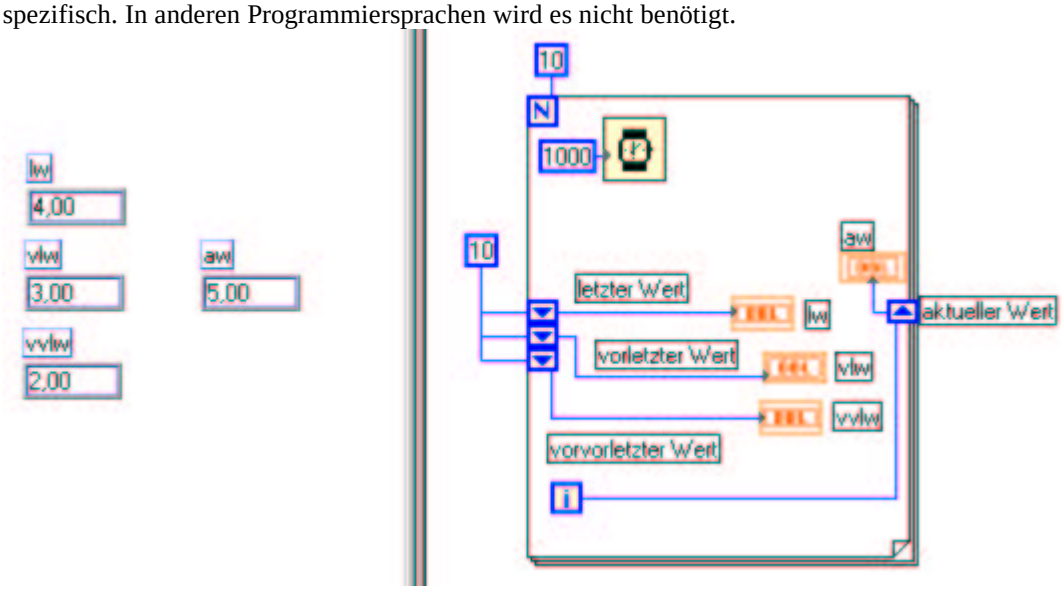
```
do {statement1;statement2; ....} while (condition == TRUE )
```

Aufgabe: Schreiben Sie ein Programm, das bei 10 beginnend, in Dreierschritten die 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.

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

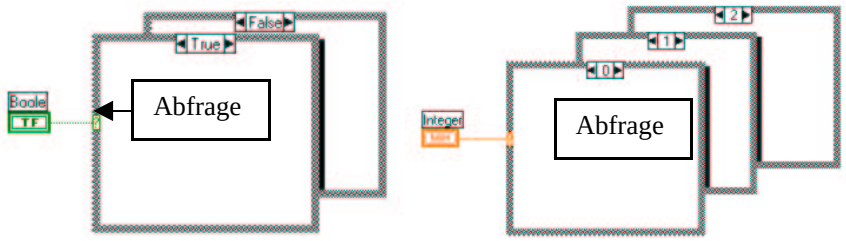
Ein Programmablauf muß sich verzweigen können. Solche Verzweigungen werden in herkömmlichen Programmen mit den Sprachelementen "if {Bedingung == TRUE} then {Anweisung1} else {Anweisung2} erzeugt. Eine weitere Verzweigung, die eher noch häufiger benötigt wird, ist die "switch−case−Verzweigung". In C lautet 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



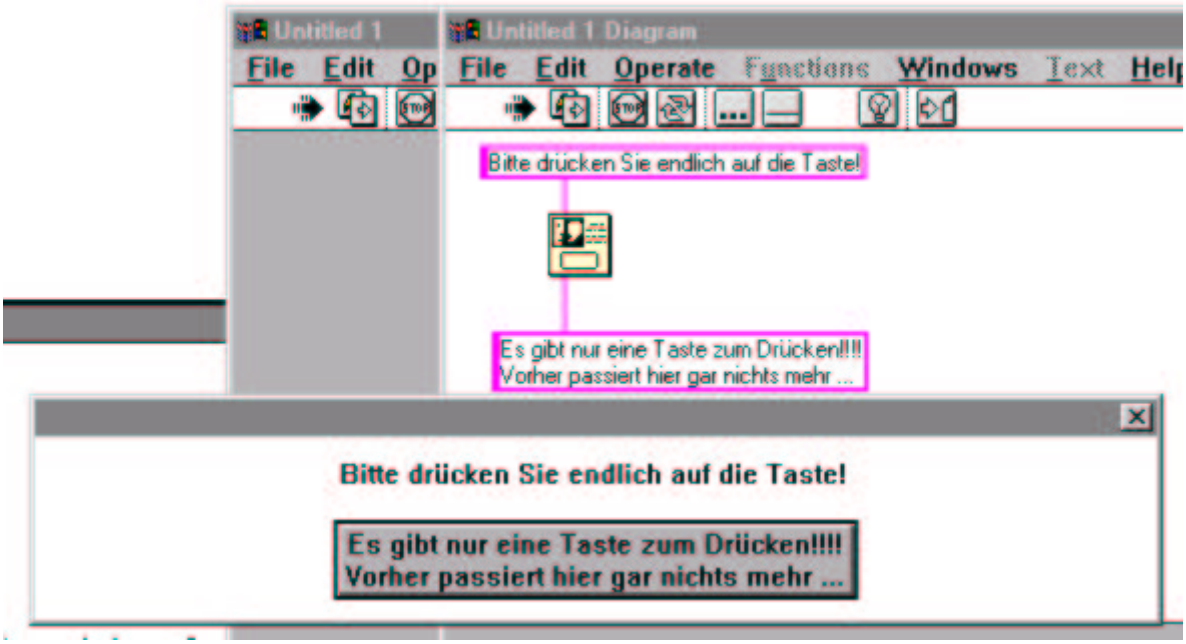
wird.

Aufgabe: Schreiben Sie ein kleines Test−Programm, mit dem Sie lernen, mit dieser Struktur umzugehen, insbesondere wie man Fälle hinzufügt, verdoppelt, entfernt und verschiebt, und wie man von Fall zu Fall kommt.

Aufgabe: Schreiben Sie ein Programm zur Steuerermittlung, mit einer Einstellung, ob der Nutzer Steuern zahlt oder nicht und einer weiteren Einstellung zum Familienstand (ledig, verheiratet, geschieden).

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 / Sequenz

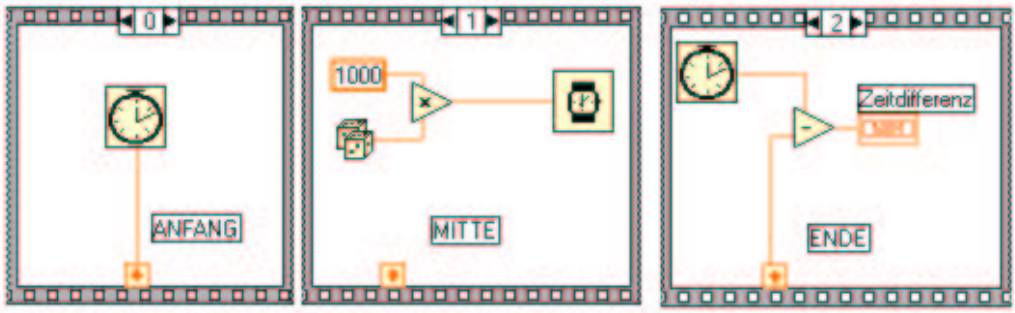
Alle klassischen Programmiersprachen enthalten einen inhärent durch die zeilenweise Abfolge der Programm−anweisungen festgelegten Ablauf. Bei LabVIEW ist das anders, denn LabVIEW selbst legt diese Abfolge fest. Wenn Sie dies verhindern müssen, weil Sie auf einen fest (von Ihnen, vom Problem, vom Lauf 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!

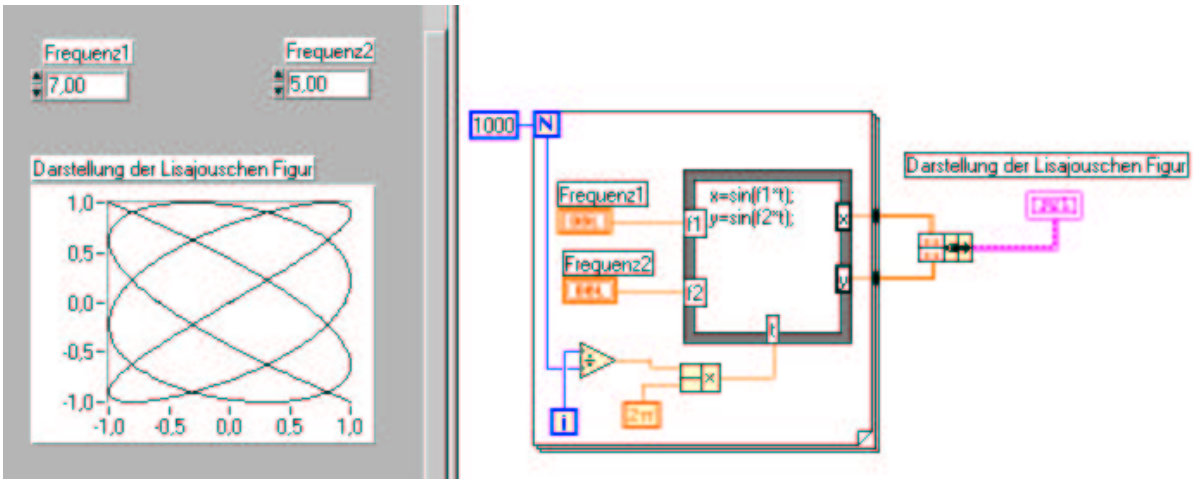


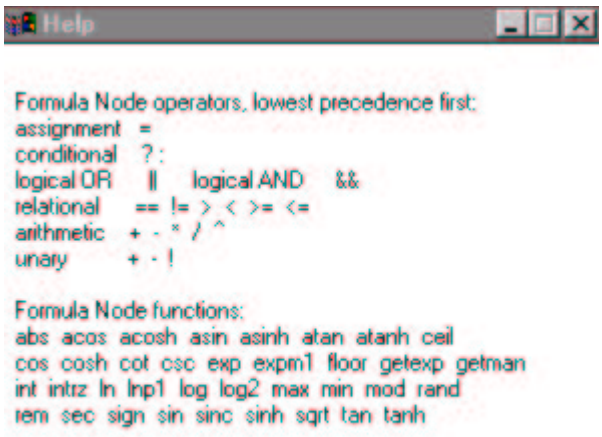
Beachten Sie am unteren Filmrand die sogenannten "lokalen Sequenzen". Sie dienen der Übergabe eines Wertes aus einem Bild in ein anderes Bild. Man erhält sie durch einen rechten Mausklick auf den Filmrand. Am Anfang sind sie ohne Pfeil, erst wenn sie verdrahtet werden, erkennt LabVIEW, ob hier was raus oder reingehen soll und trägt die entsprechende Pfeilrichtung ein. Achtung: einer lokalen Variablen 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.

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.

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.





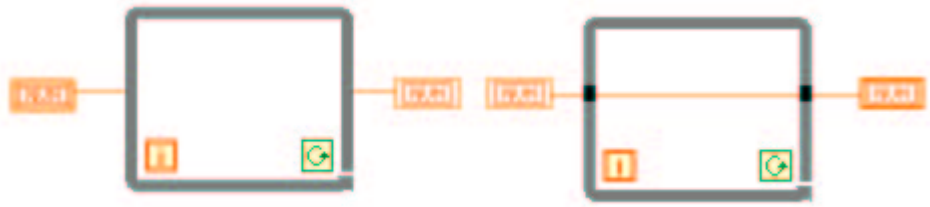
Aufgabe: Schreiben Sie ein Programm, das die Kapitalentwicklung bei einem festen Zinssatz berechnet. Anzahl der Jahre und der Zinssatz sollen als Variable eingegeben werden können. Am besten verwenden Sie dabei einen sogenannten Formelknoten − der wurde grade eben erklärt.

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 liefert der "TRUE−Case" keinen Wert nach draußen!

- 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.



Sie sehen: auch LabVIEW macht Fehler, oder vorsichtiger gesagt: auch LabVIEW hindert Sie nicht am Fehler machen. Und, zum Trost: Selbst sehr erfahrene Programmierer (egal in welcher Sprache) verbringen den größten Teil ihrer Arbeitszeit mit der Fehlersuche.

Arrays und Cluster − alias: Datenfelder und Datenstrukturen

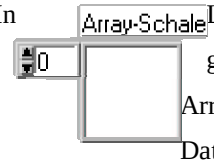
Datenfelder kurz Felder, häufiger Arrays, sind Listen von Daten vom gleichen Datentyp. Beispiel: eine Tabelle von Meßwerten − alle Elemente der Tabelle sind vom gleichen Typ: Zahlen, nichts als Zahlen, noch genauer: floating−point Zahlen.

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 in einer Personaldatei abzubilden.

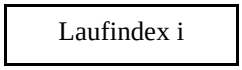
Felder

Zurück zu den Feldern. Felder erzeugt man z.B. bei der Aufnahme einer Meßreihe: wenn Sie 20 verschiedene Temperaturen messen, tragen Sie diese Werte in eine Liste ein. So etwa 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!



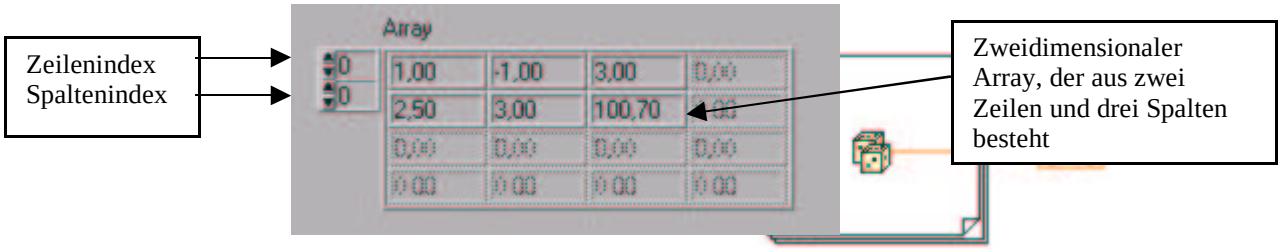
In LabVIEW erzeugt man einen Array in zwei Schritten und zwar auf der Panelseite. Dort gibt es zunächst einen Menüpunkt Array & Cluster / Array mit dem ein leeres Array, eine Array−Schale, erzeugt werden kann. Im zweiten Schritt muß gesagt werden, mit welchem Datentyp das Feld gefüllt wird. Angenommen 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.



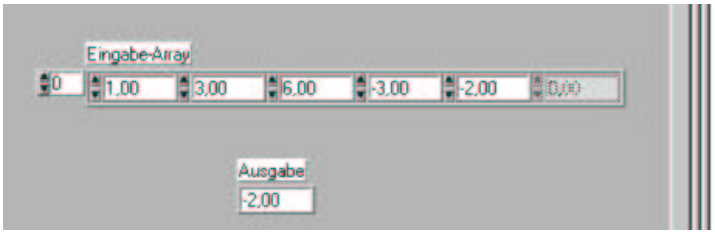
Wenn Sie den Laufindex durchklicken, können Sie die Array−Werte ablesen, beim linken Beispiel können Sie diese Werte auch noch abändern.

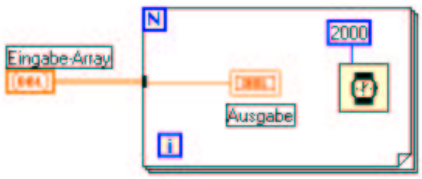
Ein zwei− oder höherdimensionales Feld erhält man mit einem rechten Mausklick auf den Indexbereich, wenn man dann "Add dimension" wählt:



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.

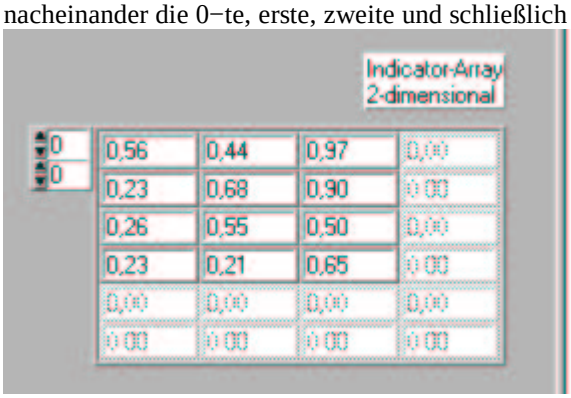
LabVIEW erkennt also selbständig, wieviele Werte in einem Array abgelegt sind. Im nächsten Beispiel können Sie das Feld von Hand z. B. mit 10 Werten füllen. Beim Programmablauf muß die Endebedingung der for−Schleife (N) nicht mit der Zahl 10 belegt werde. LabView erkennt selbst, daß das Feld mit 10 Werten belegt worden ist. Diese Möglichkeit des Indexierens kann verändert werden. For−Schleifen indexieren standardmäßig, während while−Schleifen standardmäßig nicht indexieren.

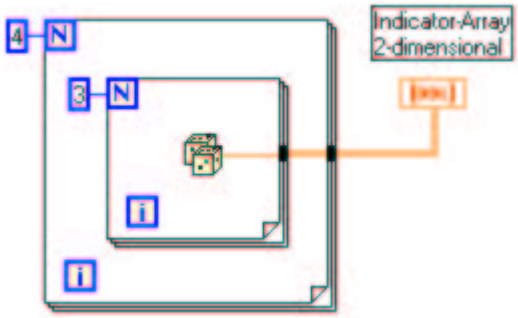




Array anzeigen.vi

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.



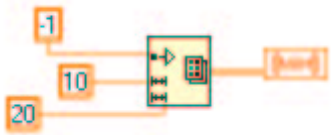


Array fuellen.vi

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.

Funktionen zur Bearbeitung von Feldern

LabVIEW bietet Ihnen 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.

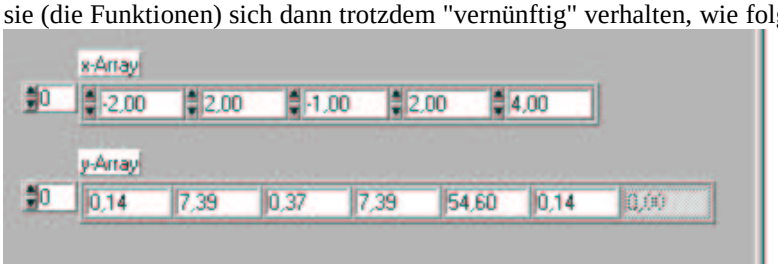


Array−Größe − size(s) gibt einen 1−dimensionalen Array zurück, bei dem jedes Element die Größe in einer Dimenion darstellt.

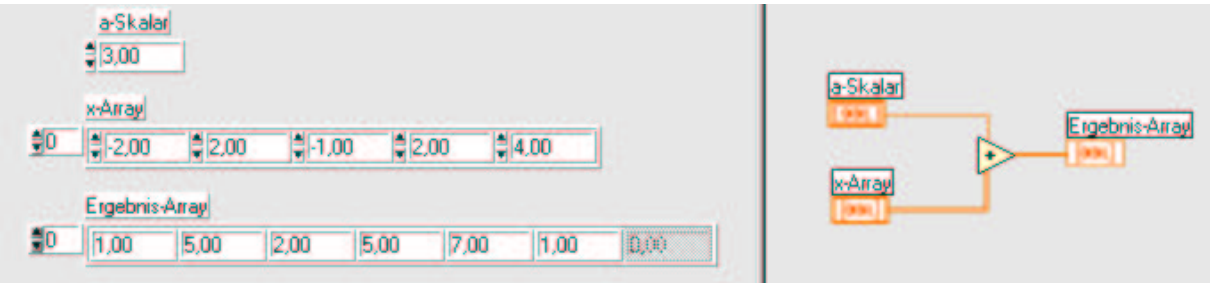
Aufgabe: Experimentieren Sie mit den Funktionen "Build Array", "Array Subset", "Index Array". Bei Anfangsschwierigkeiten hilft oft die Hilfefunktion!

Polymorphismus

LabVIEW hat eine weitere hübsche Eigenschaft, die mit "Polymorphismus" bezeichnet wird, was im Grunde "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:



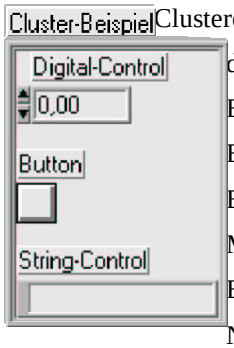
Beim folgenden Beispiel wird der eine Eingang der ADD−Funktion mit einem Skalar belegt, während am anderen Eingang ein Array liegt:



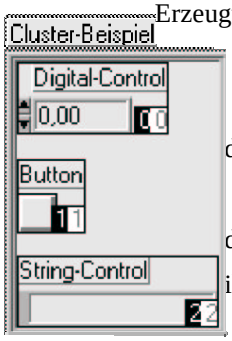
Details zum Polymorphismus finden Sie entweder im Handbuch oder durch Rumprobieren.

Cluster

Wie schon gesagt, werden mit Clustern verschiedene Datentypen zu einem neuen Datentyp zusammen gefaßt. Wie bei den 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.

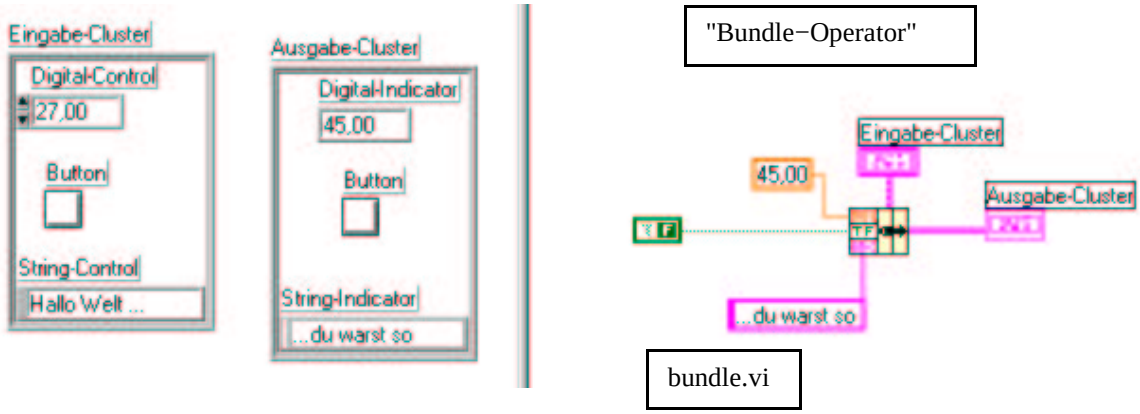


Clusterelemente haben eine logische Ordnung, die bei der Erzeugung des Clusters definiert wird: das erste eingefügte Element ist Element 0, das zweite eingefügte Element ist Element 1, usw. Elemente des Clusters können gelöscht werden, die Ordnung der Elemente kann geändert werden, wenn man mit der rechten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

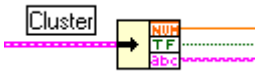


Cluster werden u.a. dann eingesetzt, wenn es darum geht, eine große Zahl von Parametern an ein VI zu übergeben. Es ist nämlich zwar theoretisch möglich ein VI mit bis zu 20 Eingängen zu versehen, es ist aber offensichtlich, daß dann sehr leicht sehr schwer zu lokalisierende Fehler auftreten können, weil die Drähte falsch angeschlossen werden.

Die Bündelung der Daten erfolgt diagrammseitig unter Verwendung des "Bundle−Operators" (rechts unten im Menü Array&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:

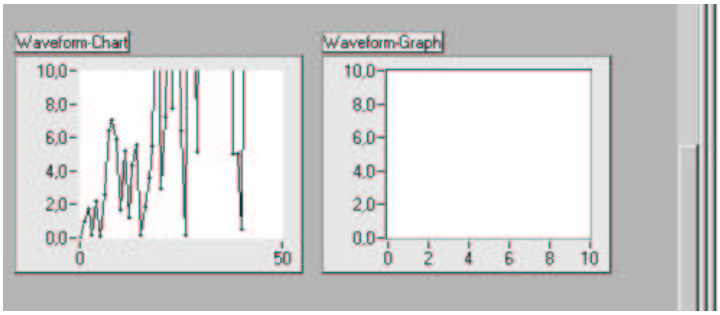


Aufgabe: Schreiben Sie ein Programm, das ein Array umfüllt nach dem Prinzip: "Die letzten werden die ersten sein".

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 Shift−Register!

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.



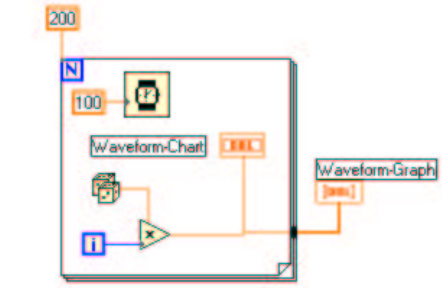
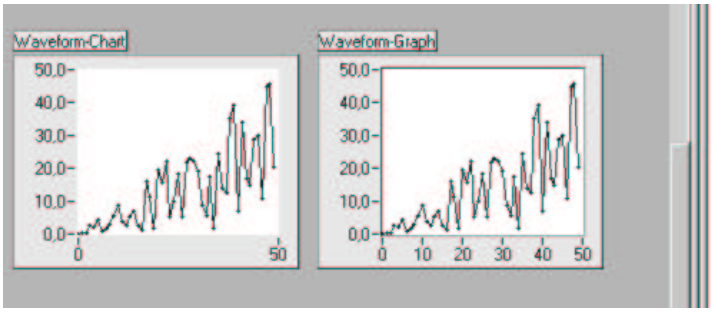


Diagramm.vi

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.



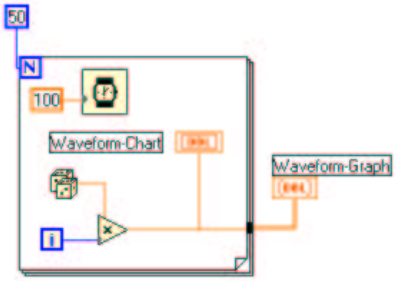
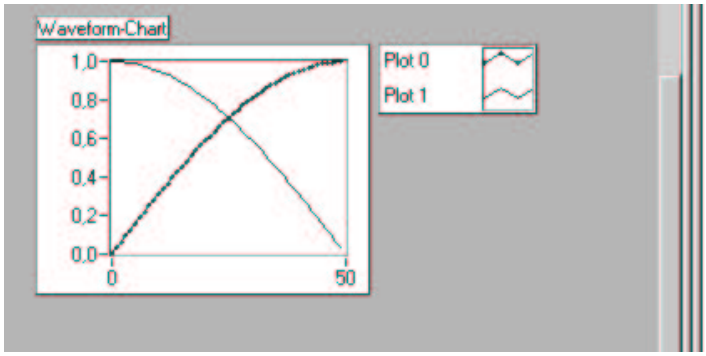


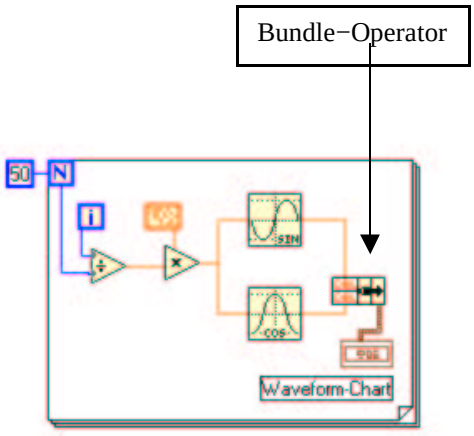
Diagramm.vi

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.





Mehrfachchart.vi

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

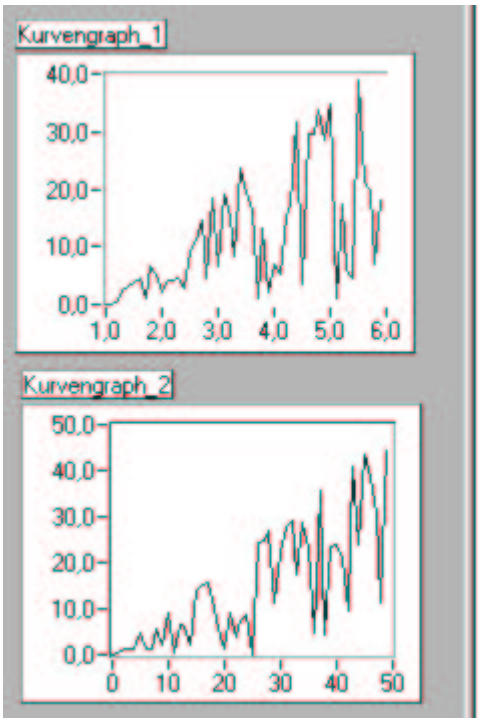
Hier können Sie den Punktstil des Graphen (Pünktchen, Kreuze, u.s.w.), Linienstil (gestrichelt, u.s.w.), Farbe, und den Interpolationsstil für jede darzustellende Kurve einzelnen festlegen. Eventuell müssen Sie zunächst erst mal wieder mit einem rechten Mausklick auf den Graphen klicken um diese überhaupt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

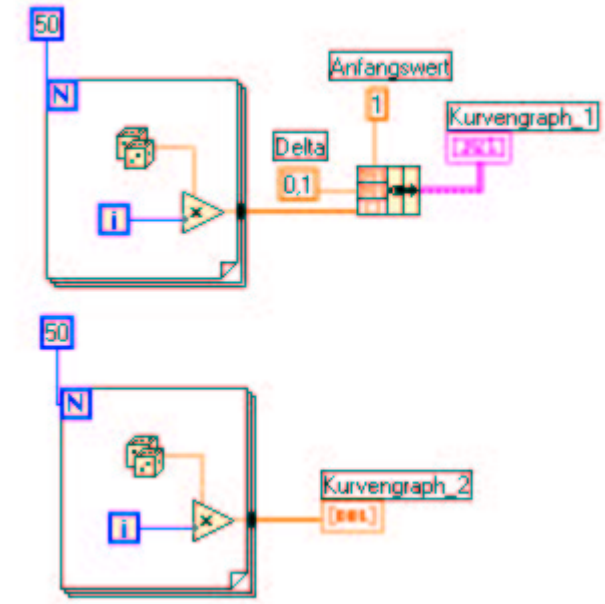
Die Palette

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 ersten Beispiel wird dieselbe Folge von Meßpunkten dargestellt. Bei der Erstellung des Graphen wird aber davon ausgegangen, daß die Abszissen alle den Abstand "Delta" haben und daß die erste Abszisse den Wert "Anfangswert" besitzt.

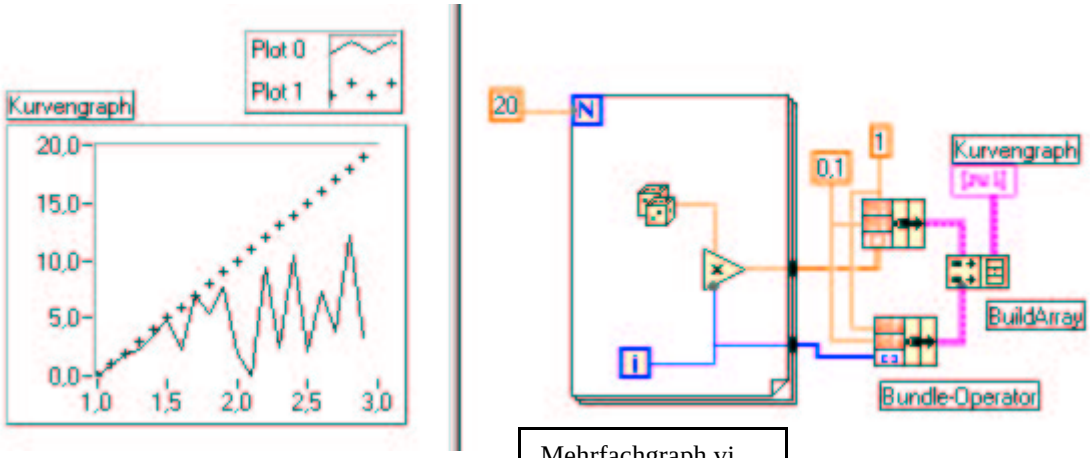




Skalierung der x−Achse.vi

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:

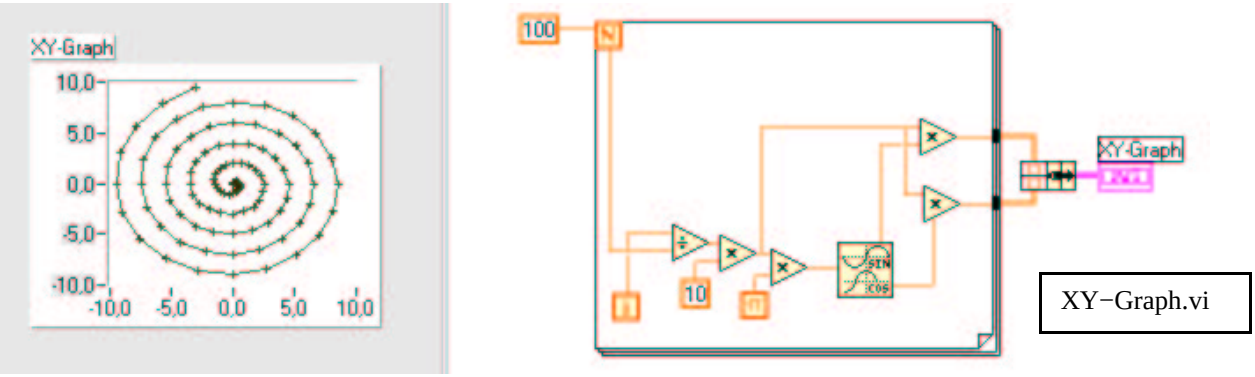


Mehrfachgraph.vi

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.

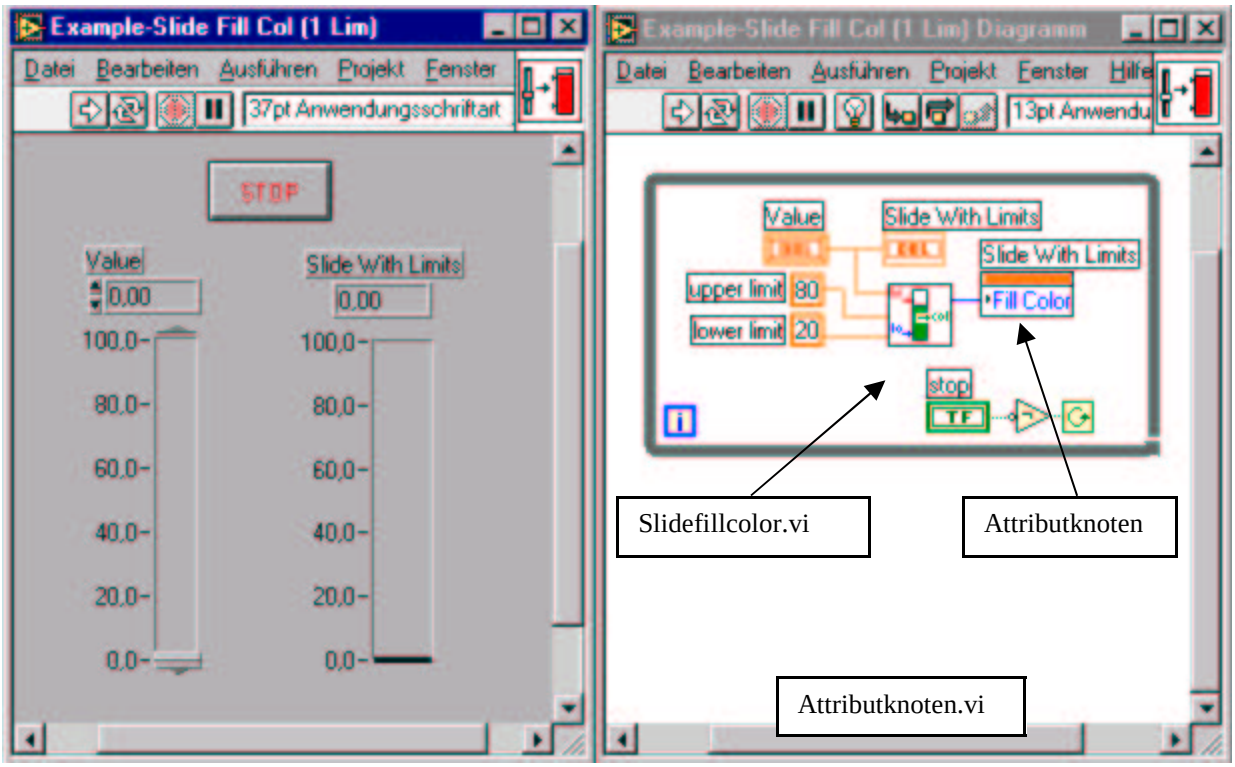


XY−Graph.vi

Aufgabe: Programmieren Sie ein VI, das die Temperaturerfassung in einer Wetterstation simuliert. Der Temperaturbereich soll von −50 Grad bis +50 Grad gehen. Jede Sekunde soll die Temperatur gemessen werden. Insgesamt 100 Messungen sollen erfaßt werden. Am Ende der Messung sollen Minimum, Maximum und −Mittelwert ausgegeben werden. Der Temperaturverlauf soll direkt dargestellt werden. Außerdem soll der Temperaturverlauf mit einem Polynom dritter Ordnung gefittet werden. Hinweis: Es gibt SubVi’s, die die meisten dieser Aufgaben für Sie erledigen: "Mean&Standard Deviation", "Array Max&Min", "Curve Fit".

Attributknoten zum Steuern von Bedienelementen und Diagramm

Attributknoten



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.

Manipulationen an Zeichenketten

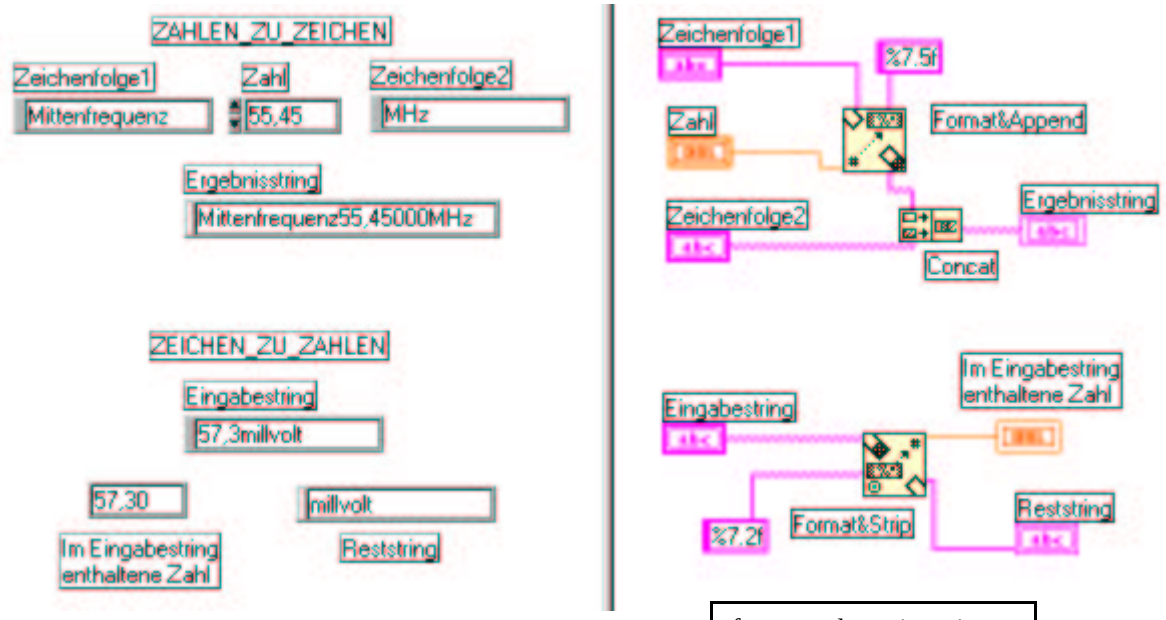
Im Zusammenhang mit Geräten, die per serieller Schnittstelle oder über den IEC−Bus angesteuert werden, ist es oft wichtig zu wissen, wie man Zeichenketten zusammensetzt und zerlegt. Außerdem müssen öfter Zahlen in Zeichenketten oder umgekehrt: Zeichenketten in Zahlen umgewandelt werden.
Die Formatanweisung 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:
Was jetzt kommt ist etwas trocken! Sie können alles besser verstehen und merken, wenn Sie gleichzeitig, das nachfolgende Programm erzeugen und die jeweiligen Aussagen nachprüfen.
Eine Formatanweisung hat folgende Syntax:

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

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

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 Umwandlungkennungen können groß oder klein geschrieben werden.

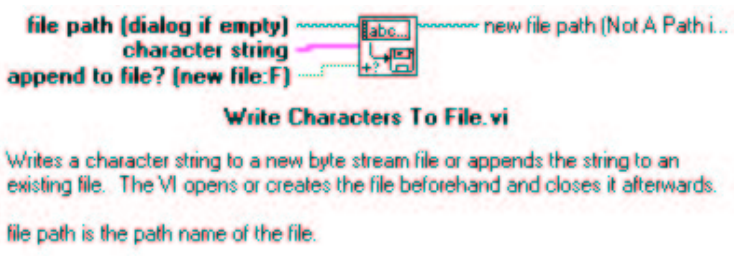
Die wichtigsten Operationen beim Umgang mit Zeichenfolgen sind "Concatenate Strings", "Format&Append" sowie "Format&Strip". Das folgende Beispiel zeigt Ihnen diese Funktionen, die Sie unter dem Menüpunkt Functions/String oder Functions/String/Additional String to Number Functions finden können.



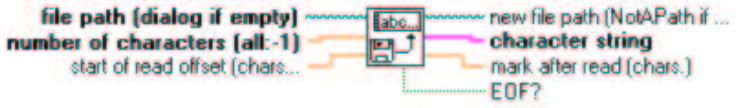
formatundsoweiter.vi

Ein− Ausgabe in Dateien

Die Funktionen, die benötigt werden, um Dateien auf die Festplatte zu schreiben oder von Festplatte zu lesen finden Sie unter dem Menüpunkt "File&Error". 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 auf der folgenden Seite 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:



Read Characters From File.vi

Reads a specified number of characters 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.

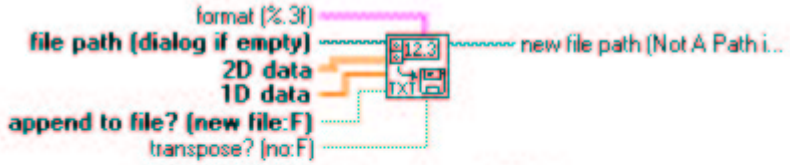
Die "Read Lines from File"−Funktion hat folgende Parameter:

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.

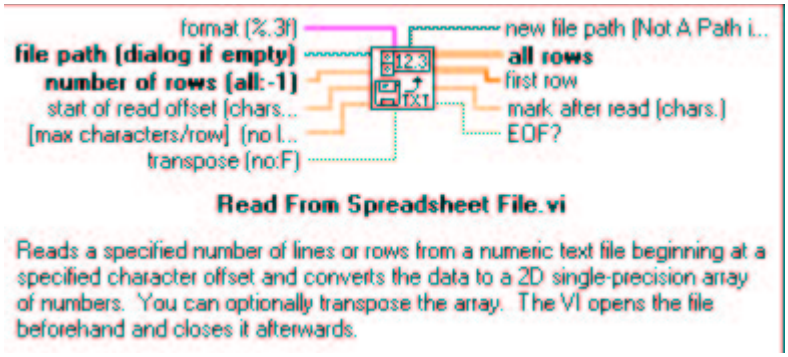
Die "Write to Spreadsheet File"−Funktion hat folgende Parameter:



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.

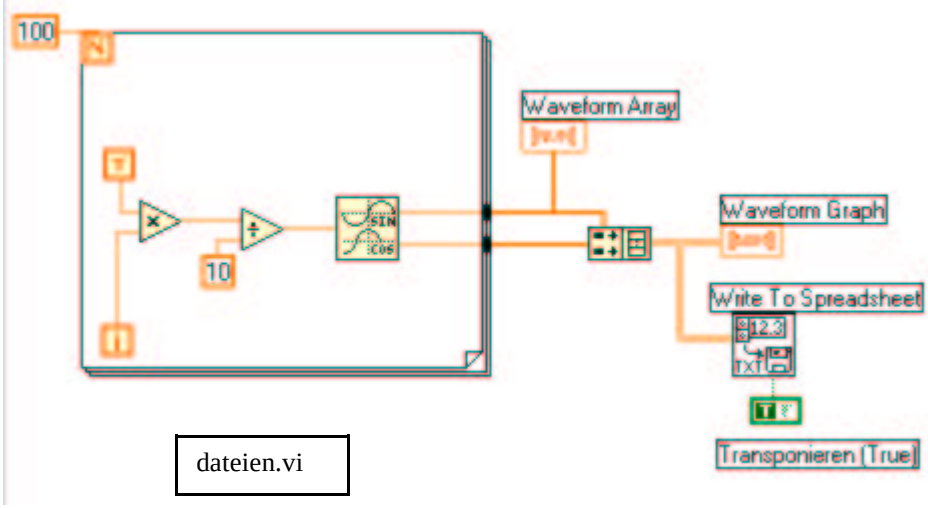
Die "Read from Spreadsheet File"−Funktion hat folgende Parameter:



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.

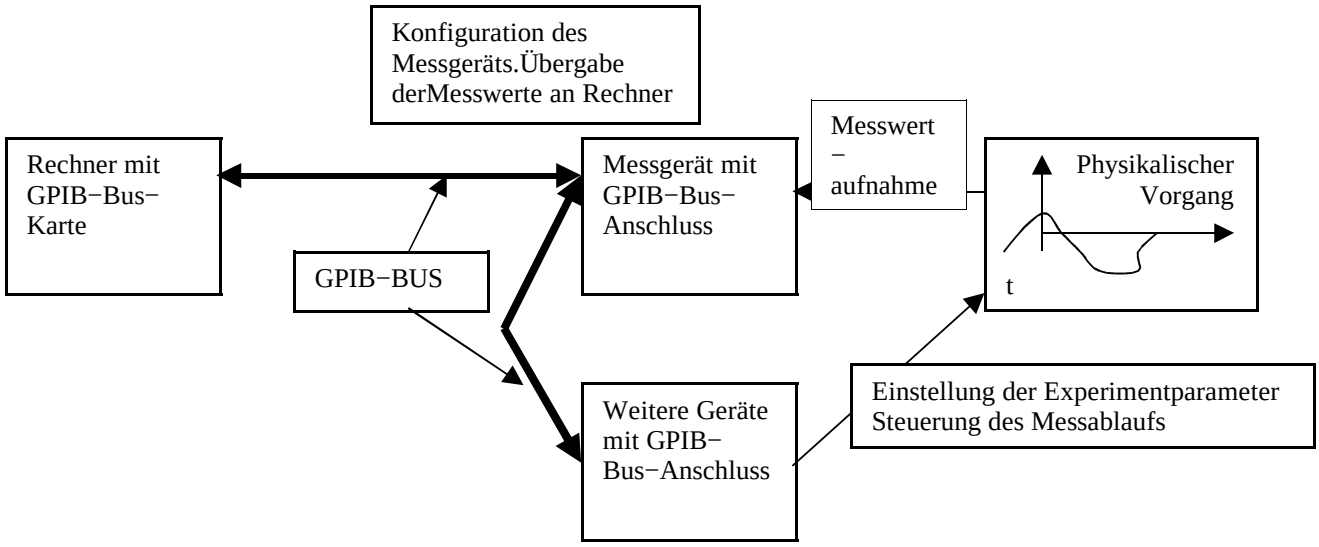
Aufgabe: Untersuchen Sie das und experimentieren Sie mit dem folgende(n) Programm



dateien.vi

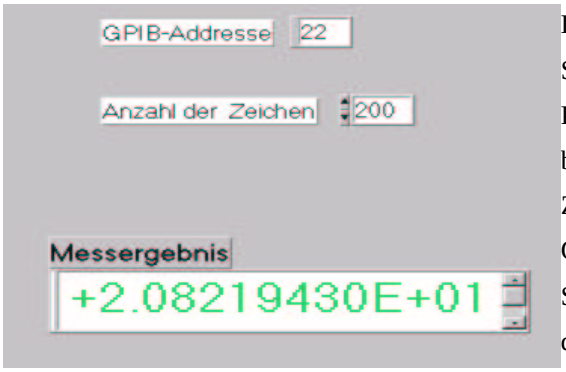
Öffnen Sie die dabei erzeugte Datei mit einem Tabellenkalkulationsprogramm und sehen Sie nach, was Sie erhalten haben. Was ändert sich, wenn die logische Konstante ("Transponieren") auf "false" gesetzt wird? Schreiben Sie ein neues Programm, in dem Sie diese eben erzeugten Werte aus der Datei zurück in das Programm einlesen und graphisch darstellen.

Messen mit LabVIEW



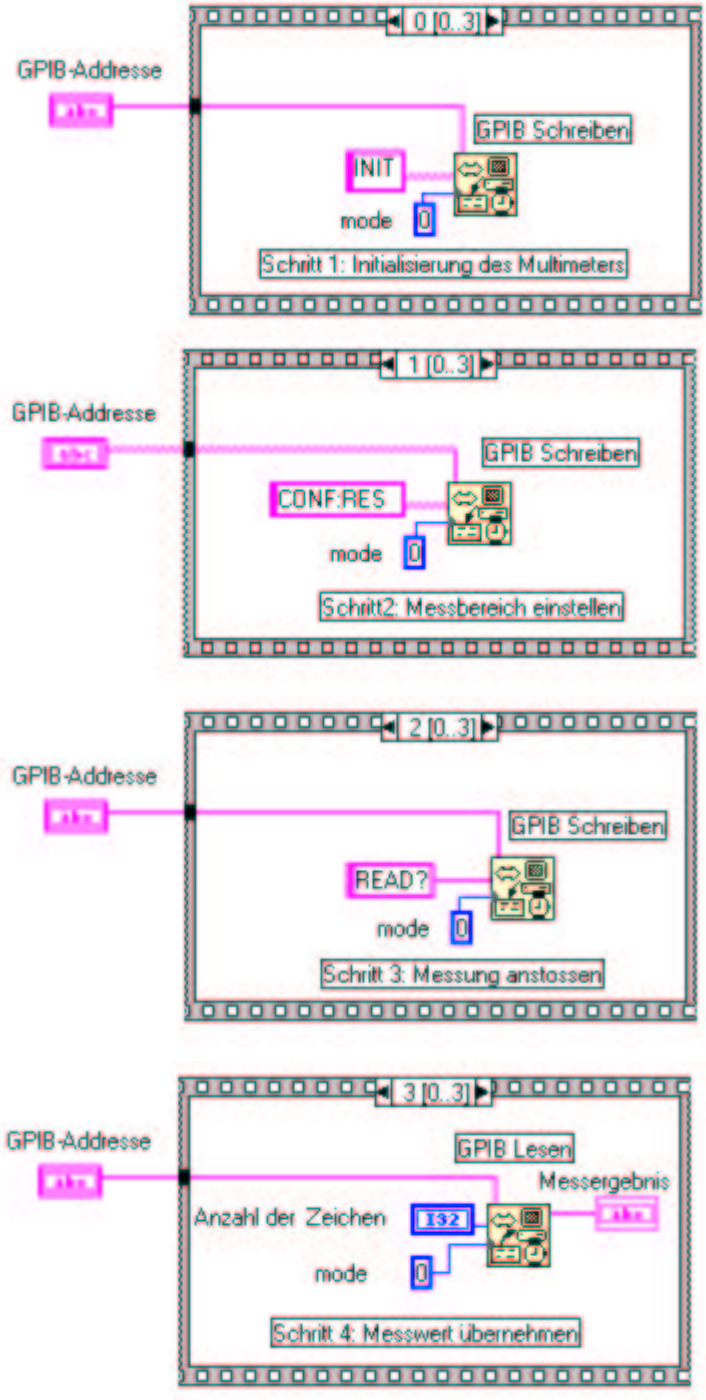
Zu Beginn der Messung muss das Messgerät konfiguriert werden. Die Experimentparameter müssen eingestellt und der Ablauf des Experiments muss kontrolliert werden. Die Messwerte des (zeitlich veränderlichen) physikalische Vorgangs, der erfasst werden soll, werden durch das Messgerät aufgenommen und über 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.

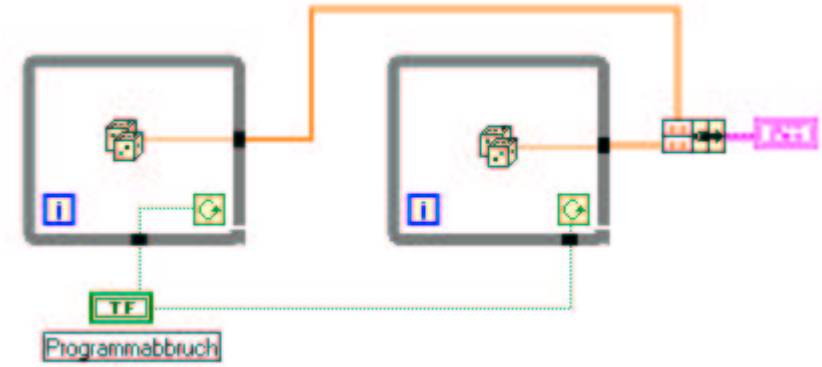


Lokale und globale Variable

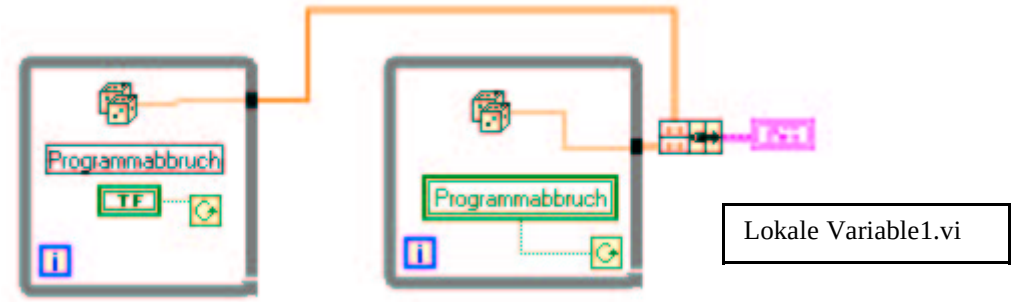
Wenn Sie schon mit klassischen Programmiersprachen gearbeitet haben, sind Ihnen diese beiden Begriffe sicher geläufig. Lokale Variable sind nur in einem begrenzten Programmbereich gültig, während auf globale Variable aus dem gesamten Programm (lesend und schreibend) zugegriffen werden kann. Das kann zu verwirrenden Situationen und schwer zu findenden 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 und globale Variable und auch in LabVIEW gibt es die diesbezüglichen Fehlermöglichkeiten, insbesondere bei der Nutzung globaler Variablen!

Lokale Variable

Stellen Sie sich vor, daß Sie in einem VI den Wert einer Variablen an vielen verschiedenen Stellen benötigen. Mit vielen Drahtmetern könnten Sie dieses Problem zwar lösen. Aber Sie erhalten bei dieser Vorgehensweise ein verwirrendes Programm! Außerdem kann man sich Fälle vorstellen, wo man selbst mit 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.

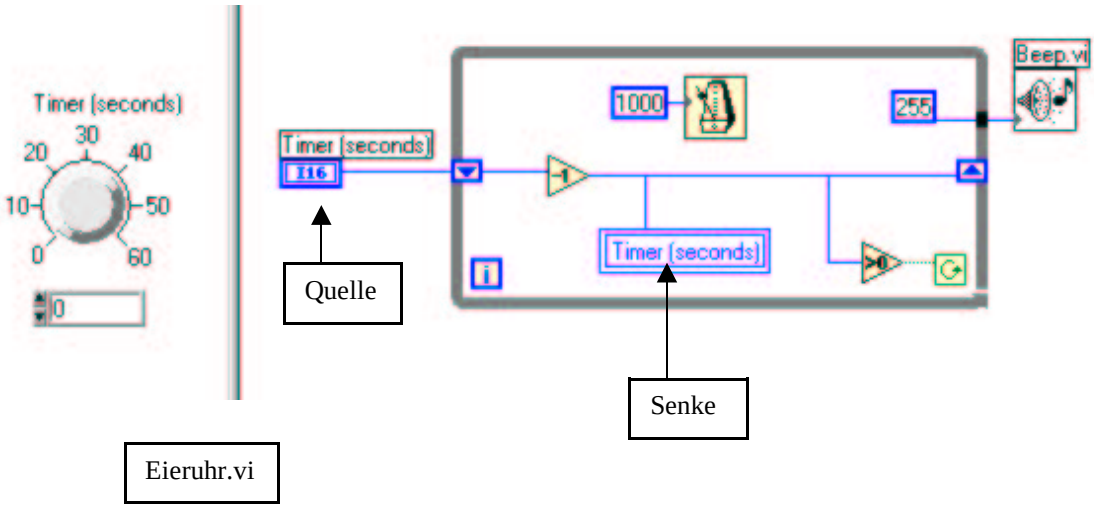


Egal wohin Sie die Programmabbbruch−Variable schieben: Sie werden es nicht schaffen, das Programm vernünftig zum Laufen zu bringen. (Schalten Sie in den "High−Lighting−Modus" um zu verstehen warum es nicht gehen kann).



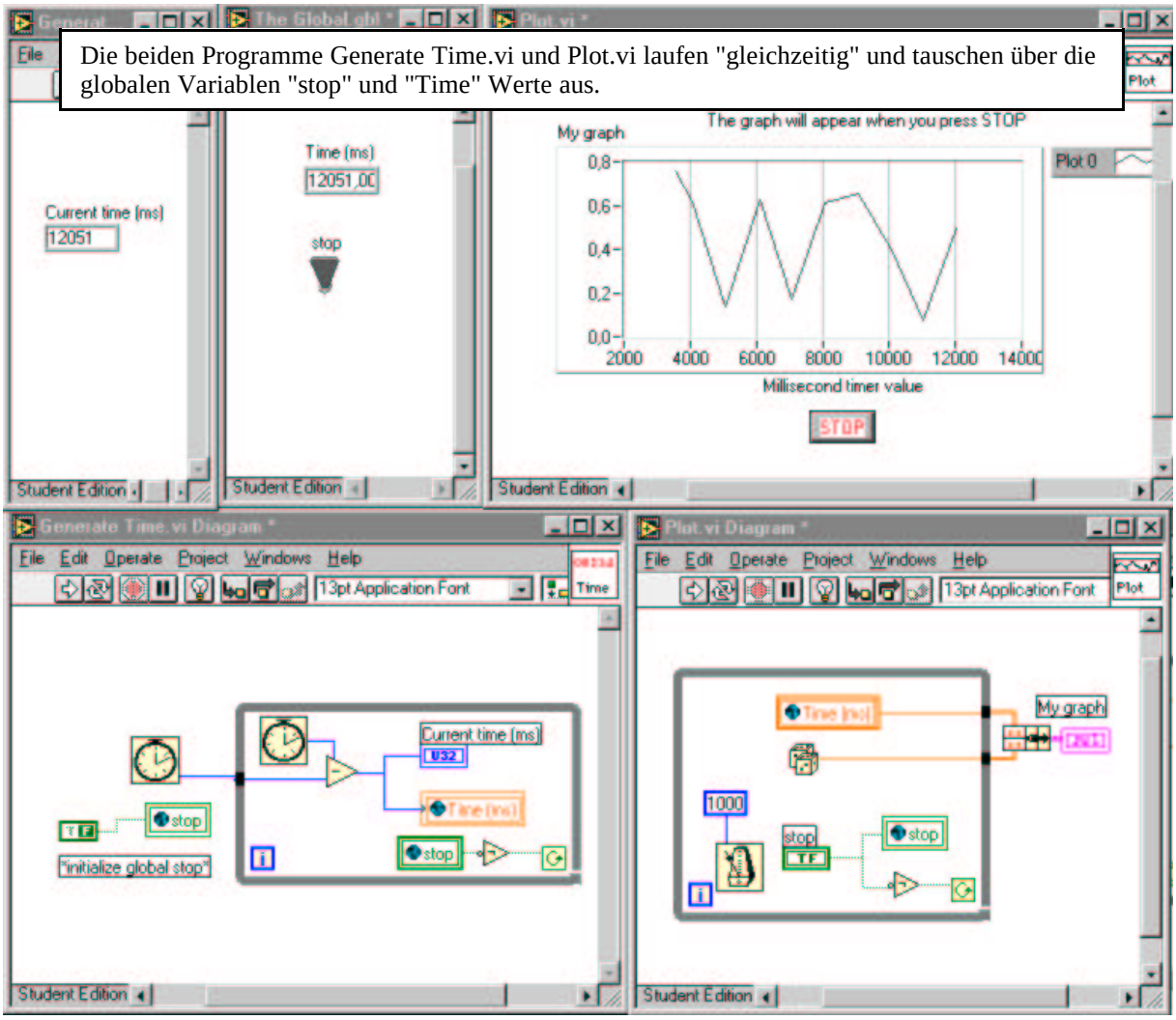
Das Problem können Sie lösen, wenn Sie in der rechten Schleife eine lokale Variable, die eine Kopie der Programmabbruchvariablen ist, erzeugen. Lokale Variable finden Sie unter dem Programmpunkt "Structures", dort wo auch die "while−Schleife" zu finden ist. Nach dem Anklicken müssen Sie diese lokale 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

hneinzutransferieren. Lokale Variable haben noch eine wichtige Eigenschaft: Sie können als Datenquelle oder als Datensenke definiert werden − beachten Sie diesen Punkt, wenn Sie Verdrahtungsprobleme bekommen. Darüber hinaus kann eine lokale Variable an der einen Stelle 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)



Globale Variable

Globale Variable sollten aus den oben genannten. Gründen nur sehr sorgfältig und überlegt eingesetzt werden. Der Geltungsbereich von lokalen Variablen überspannt nur das VI, in dem sie definiert wurde. Angenommen, Sie müssen mehrere VI’s betreiben, die "quasi−parallel" arbeiten sollen. Wenn dann gewisse Informationen in den verschiedenen VI’s gemeinsam benötigt werden, dann müssen Sie globale Variable einsetzen. Die Datenstruktur "globale Variable" finden Sie im gleichen Menüpunkt, der auch die anderen Programmstrukturen zur Verfügung stellt. In diesem Kurs ist nicht der Platz und nicht die Zeit um die vielfältigen Probleme im Zusammenhang mit globalen Variablen zu behandeln. Deshalb gibt es zum Thema "globale Variable" nur ein Beispiel, an dem Ihnen einige Aspekte klar werden können. Unser Beispiel umfaßt zwei VI’s, die parallel arbeiten: "Generate Time.vi" mißt die Zeit in ms, von dem Moment an, in dem dieses VI gestartet wurde und "Plot.vi" erzeugt vom Moment seines Anlaufens an jede Sekunde einen Zufallswert, der in Abhängigkeit von der mit "Generate Time.vi" gestarteten Zeitmessung geplottet werden soll, sobald die Stop−Taste von "Plot.vi" gedrückt wurde. Unten finden Sie die Panels und die Diagramme der beiden eben beschriebenen VI’s und zusätzlich noch ein Panel "The Global.gbl" für das kein Diagramm existiert, das aber zwei Variable − Time und stop − enthält, auf die in den anderen beiden VI’s in der Weise zugegriffen wird, daß sich das oben beschriebene Verhalten ergibt.
Beachten Sie die Initialisierung der globalen Variablen "stop" und wie die Variable Time, die in "Generate Time.vi" gesetzt wird und jede Sekunde in "Plot.vi" ausgelesen wird und auf dem Rand der while−Schleife angesammelt wird.



Abschlußaufgabe: Programmieren Sie eine Simulation einer Verkehrsampelanlage. Achtung! Die Aufgabe kann umfangreich werden! Gehen Sie schrittweise vor! Zuerst sollten Sie eine einzige Ampel programmieren, überlegen Sie sich dazu ein geeignetes VI. Dann folgt eine Anlage, die aus zwei Ampeln besteht und den Verkehr immer nur in einer Richtung durchläßt. In einem dritten Schritt könnten Sie sich dann eine Straßenkreuzung vornehmen und wenn Sie dann immer noch nicht genug haben, können Sie sich eine Anlage überlegen, bei der die Linksabbieger gesondert behandelt werden.