LabVIEW

Rudolf Lehn

Schülerforschungszentrum
Bad Saulgau

21. Mai 2007

Inhaltsverzeichnis

Das folgende Skript gibt eine grundlegende Einführung in die LabVIEW Programmierung und MSR (messen, steuern, regeln) mti LabVIEW Literatur:

- Rahman Jamal: **LabVIEW für Studenten**
- Wolfgang Georgi: **Einführung in LabVIEW**

Ein Update des Skripts wird auf der URL **www.sfz-bw.de** veröffentlicht.

Hinweise und Korrekturen werden erbeten an **sfz@rlehn.com**.

1. Einführung

1.1. Erste Schritte mit LabVIEW

LabVIEW ist eine Programmierumgebung, mit der man effizient Programme zur Steuerung von Geräten und zur Meßdatenverarbeitung erstellen kann. Bei LabVIEW wird eine „graphische Programmiersprache G“ eingesetzt, die es erlaubt, Programme zu schreiben, die sehr gut strukturiert und dokumentiert sind.

LabVIEW ist die Abkürzung für „Laboratory Virtual Instrument Engineering Workbench“. Im Folgenden wird LabVIEW 8.0 auf den Plattformen Windows XP sowie OS X und Linux eingesetzt. Der LabVIEW-Kurs soll als praktischer Einstieg in die Messdatenerfassung, aber auch in die Programmierung mit LabVIEW dienen. Die Messdatenerfassung sowie die Steuerung und Regelung wird mit dem einfachen USB-6008 Device erfolgen.

Ein LabVIEW-Programm wird als virtuelles Instrument (VI) bezeichnet. Ein VI hat zwei Hauptbestandteile

- Das Frontpanel ist die Benutzerschnittstelle des VI. Es simuliert die jeweiligen Messgeräte mit Drehknöpfen, Schaubildern, ...
- Das Blockschaubild enthält die Programmierlogik des VI. Das Blockdiagramm, welches mit der LabVIEW eigenen Programmiersprache G entworfen wird, enthält das tatsächlich ausführbare Programm.

Ein VI, das innerhalb eines anderen VI zum Einsatz kommt, wird SubVI genannt, d. h. es ist ein Unterprogramm. In LabVIEW wird somit die modulare Programmierung unterstützt. Man teilt eine Anwendung in immer einfachere Teilaufgaben. Dabei kann jedes SubVI einzeln ausgeführt werden und so die Fehlersuche erleichtert werden.

Anwendungsbeispiel

Dieses Programm stellt Sinuskurven beliebiger Frequenz in einem `Waveform Graph` dar.

Erstellung des LabVIEW-Programms:

- LabVIEW - File - New VI
- Im Frontpanel (vgl. 1.2.1) die Control-Palette öffnen und dort unter Modern oder Classic `Waveform Graph` auf das Frontpanel ziehen. Im Folgenden wird - soweit möglich - Modern verwendet.
- Schalte auf das Blockdiagramm (vgl. 1.2.2) um. Menüleiste - Windows - Show Block Diagram. Im Blockdiagramm ist bereits das Schaltbild von `Waveform Graph` aus dem Frontpanel übertragen worden.
- Öffne im Blockdiagramm die Functions-Palette und öffne dort `Signal Processing` und `Waveform Generation`. Ziehe `Sine Waveform.vi` ins Blockschaltbild.
- Am Eingang `frequency` kann mit der rechten Maustaste ein `Control`-Objekt zur Eingabe der Frequenz angefügt werden. Dies erscheint dann auch im Frontpanel. Mit einem Doppelklick auf den Text `frequency` kann der Hinweistext verändert werden.
- Am Ausgang `signal out` des `Signal Processing` Objekts kann mit der Maus eine Verbindung zum `Waveform Graph` gezogen werden. Dies wird durch ein Drahtspulen-Symbol gekennzeichnet.

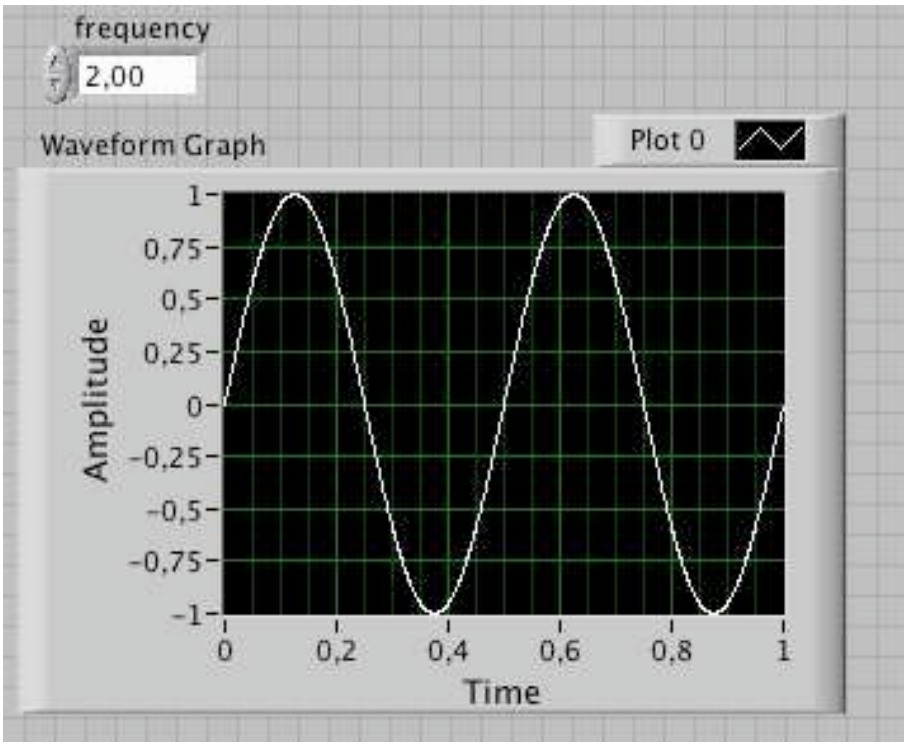


ABB.1.1

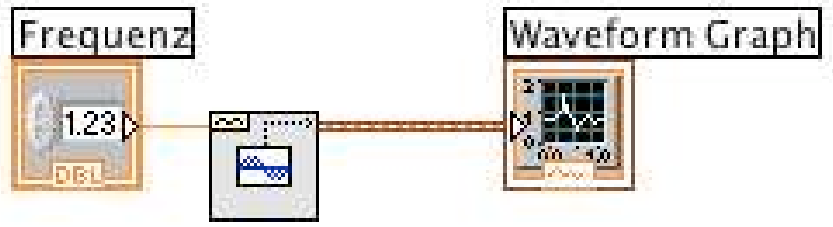


ABB.1.2

Die Schaltfläche `Run` auf Frontpanel oder im Blockdiagramm, die das Aussehen eines Pfeils hat, startet nach dem Anklicken das VI.

1.2. LabVIEW Entwicklungsumgebung

In diesem Kapitel soll die LabVIEW-Umgebung und das Zusammenwirken von Frontpanel und Blockdiagramm erläutert werden.

1.2.1. Das Frontpanel

Der LabVIEW-Anwender interagiert über das Frontpanel mit dem Programm. Das Frontpanel besteht aus Bedien- und Anzeigeelementen. Bedienelemente sind Eingabeobjekte (Datenquelle) wie Drehknöpfe, Schalter, ..., welche Daten an das Blockdiagramm des VI übermitteln. Anzeigeelemente zeigen vom Programm erzeugte Werte an (Datensenke).

Bedien- und Anzeigeelemente werden auf dem Frontpanel eingefügt, indem man sie aus der `Controls`-Palette auswählt und an den gewünschten Platz positioniert oder im Blockdiagramm (vgl. 1.2.2) am entsprechenden Anschluss mithilfe der rechten Maustaste `Create` aktiviert und eine der Möglichkeiten `Constant`, `Control`, `Indicator` auswählt.

ABB.1.3 Controls

1.2.2. Das Blockdiagramm

Das Blockdiagramm stellt das Steuerprogramm eines LabVIEW-VI dar. Man erstellt das Blockdiagramm, indem man Objekte mithilfe der „Drahtspule“ miteinander verbindet. Ein Blockdiagramm besteht aus Anschlüssen, Knoten und Verbindungen (vgl. 1.2.4).

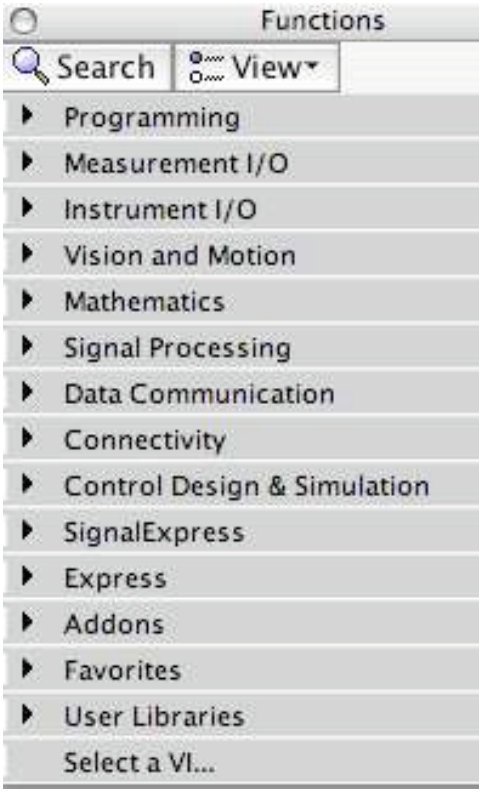


ABB. 1.4 Functions

1.2.3. Demonstrationsbeispiel

Berechnung der Summe zweier Zahlen

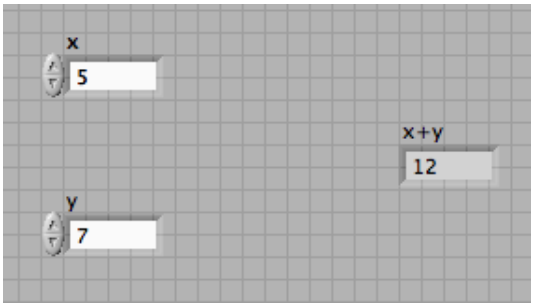


ABB. 1.5 Frontpanel

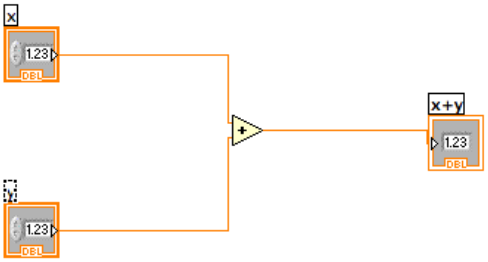


ABB. 1.6 Blockdiagramm

- Starte LabVIEW und erstelle ein neues (leeres) VI (Blank VI)
- Wechsle zum Frontpanel und wähle `Controls - System`

- Ziehe zwei `System Spin Control` Elemente zur Werteingabe und ein `Classic Numeric Indicator` als Wertanzeige in das Frontpanel. Einfacher geht dies mithilfe der rechten Maustaste am Ausgang des Add-Objekts im Blockdiagramm (`Create - Indicator`).
- Ziehe eine Addition zwischen Ein- und Ausgabe (`Functions - Programming - Numeric - Add`) ins Blockdiagramm
- Lege im Blockdiagramm mit der Drahtrolle die Verbindungen.
- Bezeichne die Eingabefelder mit x und y und das Anzeigefeld mit x+y (Doppelklick auf die Textfelder)

Hinweis

Statt `Controls - System` kann auch `Modern - Numeric` mit den Elementen `Numeric Control` und `Numeric Indicator` verwendet werden.

Mit Run wird dann das VI gestartet. Die Control-Elemente auf dem Front-Panel oder im Blockdiagramm können mit der Option „Align Objects“ ausgerichtet werden, die neben `Text Settings` zu finden ist.

1.2.4. Anschlüsse - Knoten - Verbindungen

- Sobald ein Bedien- oder Anzeigeelement auf dem Frontpanel platziert wird, erstellt LabVIEW im Blockdiagramm einen entsprechenden Anschluss. Anschlüsse von Bedienelementen haben einen dicken Rahmen und Anzeigeanschlüsse haben einen dünnen Rahmen. Man kann bei Anschlüssen auch von Datenquellen und Datensenken sprechen.
- Ein Knoten ist definiert als programmausführendes Element (z. B. Addition, Wurzelberechnung, Funktionen, Sub VI, . . .).
- Zwischen Knoten und Anschlüssen sind die Verbindungen. Sie liefern Daten von einer Datenquelle an eine oder mehrere Datensenken. Jede Verbindung ist entsprechend ihrem Datentyp eingefärbt.

1.2.5. Wichtige Paletten

Die Palette `Tools`, die Palette `Controls` beim Frontpanel und die Palette `Functions` beim Blockdiagramm sind die am häufigsten eingesetzten Paletten in LabVIEW.

In der Tools-Palette ist `Operate Value` (Hand) zur Änderung der Werte der Eingaben und Anzeigen auf dem Frontpanel wichtig. Das `Position/Size/Select` wählt Objekte aus, verschiebt sie und ändert ihre Größe. `Edit Text` erstellt und bearbeitet Textfelder. `Connect Wire` zieht Verbindungen zwischen Objekten im Blockdiagramm.

ABB.1.7 Tools

1.2.6. Die Symbolleiste

ABB.1.8

- Die Schaltfläche `Run`, die das Aussehen eines Pfeils hat, startet nach dem Anklicken ein VI. Der Pfeil wechselt sein Aussehen.
- Die Schaltfläche `Run Continuously` lässt die Ausführung immer wieder von neuem beginnen, bis die Schaltfläche stop angeklickt wird.
- Die Schaltfläche `Abort Execution` stoppt ein VI sofort.
- Die Schaltfläche `Pause` hält das VI an. Damit kann z. B. Einzelschritt-Debugging ausgenutzt werden.
- Die Schaltfläche `Highlight Execution` im Blockdiagramm bewirkt eine Illustration des Datenflusses.
- Wird um die auszurichtenden Objekte ein Rahmen gezogen, dann können sie mit den Schaltflächen **Align Objects**, **Distribute Objects**, **Resize Objects** angeordnet werden.

1.2.7. Das Hilfefenster

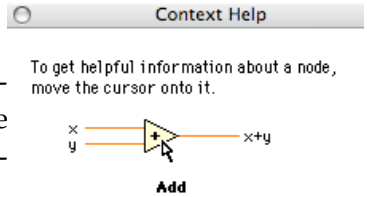


ABB. 1.9 Context Help

Das LabVIEW Hilfefenster bietet viele Hilfsinformationen für Funktionen, Konstanten, SubVI sowie Eingaben und Anzeigen. Um das Hilfefenster zu aktivieren, wählt man`Show Context Help`.

1.3. Vertiefung der VI-Techniken am Beispiel

1.3.1. VI-Erstellung

In der Regel wird damit begonnen, Eingaben und Anzeigen auf dem Frontpanel zu positionieren. Wenn ein Element auf dem Frontpanel abgelegt wird, erscheint der dazugehörige Anschluss im Blockdiagramm.

Soll die Beschriftung geändert werden, klickt man das Textfeld über der digitalen Eingabe doppelt an, sofern automatisches Tool in der `Tools`-Palette eingestellt ist (grün). Mit `Text Settings` in der Symbolleiste lasten sich die Textattribute ändern.

Mit `Edit Text` aus `Tools` kann an einer beliebigen freien Stelle ein Text erstellt werden. Erstelle eine freie Beschriftung auf dem Frontpanel mit dem Text „SFZ - Ort der Ideen“. Die Texteingabe wird mit einem Klick an einer Stelle außerhalb der Beschriftung abgeschlossen.

Nach den Festlegungen im Frontpanel sind im Blockdiagramm nur noch die Funktionen, SubVI und Strukturen je nach Programmanforderung zu ergänzen. Öffne dazu die Palette *Functions* und wähle die gewünschten Elemente aus; z. B. die `Add` aus `Mathematics`, `Numeric` im Blockdiagramm ablegen.

LabVIEW unterscheidet im Wesentlichen zwischen drei einfachen Eingabe- und Anzeigetypen: numerisch, Boolscher Typ, String.

In LabVIEW gibt es eine Menge Möglichkeiten, numerische Objekte einzugeben bzw. auszugeben: Drehknöpfe, Schieberegler, Thermometer, Digitalanzeige, ...

Übung

Erstelle ein VI, in dem die Eingabe über einen Drehknopf ausgeführt wird. Zu dieser Eingabe solle eine Konstante addiert werden. Die Ausgabe soll sowohl digital wie auch analog sein.

Die Erscheinung numerischer Anschlüsse im Blockdiagramm hängt von der Darstellung der Daten ab. Blockdiagrammanschlüsse sind blau für ganze Zahlen, orange für Fließkommazahlen. Die Anschlüsse enthalten auch Buchstaben zur Beschreibung des Datentyps, z. B. DBL für Fließkommazahlen doppelter Genauigkeit.

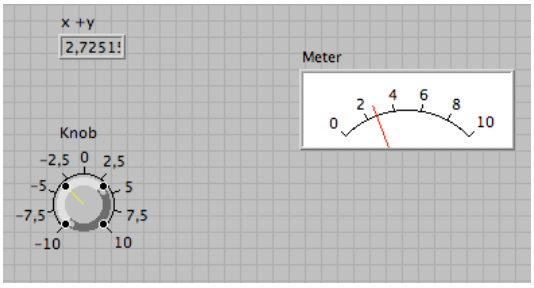


ABB.1.10

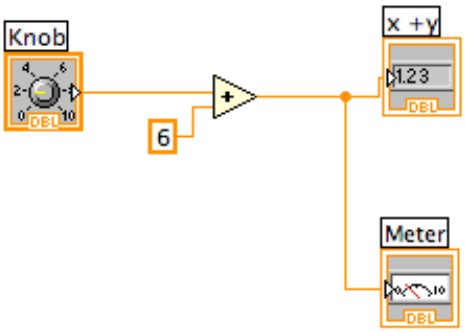


ABB.1.11

Die Front Panel Objekte findet man in `Controls` - `Classic` - `Classic Numeric` - `Knob` - `Meter` - `Numeric Indicator`.

Die Darstellung numerischer Werte kann mit der rechten Maustaste über *Properties-Data Range* gewählt werden.

LabVIEW bietet die Möglichkeit, einen bestimmten gültigen Bereich von Werten zu erzwingen. Wenn man z. B. nur Eingaben zwischen 0 und 100 mit einer Schrittweite von 5 zulassen möchte, kann man dies über Data Range festlegen.

Verbindungstechniken

Die logische Verbindung der unterschiedlichen Elemente im Blockdiagramm wird mit dem Verbindungs-Werkzeug (Connect Wire), welches die Form einer Drahtspule besitzt, hergestellt. Fehlerhafte Verbindungen erscheinen als gebrochene Verbindungen anstatt der üblichen farbigen Verbindungen. Eine falsche Verbindung kann man entfernen, indem man sie auswählt und löscht. Alle defekten Verbindungen lassen sich mit Strg-B (Windows) oder Apfel-B (OS X) entfernen.

Ausführen der VI

Mit dem Ausführen-Befehl aus dem Menü `Operate - Run` kann man ein VI starten. Alternativ kann man das entsprechende Tastenkürzel (z. B. Ctrl R) oder die Start-Schaltfläche `Run` in der Symbolleiste verwenden. Mit der Schaltfläche `Run Continuously` kann man ein VI im Dauerlauf betreiben.

2. Grundlagen der LabVIEW-Programmierung

2.1. SubVI

Ein SubVI entspricht einem Unterprogrammaufruf in einer Programmiersprache. Jedes VI kann als SubVI im Blockdiagramm eines anderen VI verwendet werden. Hierzu muss in der Functions-Palette `Select a VI` angeklickt werden. Damit kann ein beliebiges VI im System gewählt werden. Bevor man ein VI als SubVI einsetzt, muss man festlegen, wie es Daten vom aufrufenden VI erhält und an das VI zurückgeben kann.

Jedes SubVI muss ein Symbol besitzen, das es in einem übergeordneten VI darstellt. Mit `Icon Editor` kann das Symbol bearbeitet werden. Es kann mit Drag&Drop jede beliebige Bitmap in den Symbol-Editor gebracht werden.

Bevor man ein VI als SubVI verwendet, muss man dem Anschlussblock im Frontpanel die notwendigen Anschlüsse zuweisen. Der Anschlussblock eines VI weist die Eingaben und Anzeigen des VI den Ein- und Ausgabeanschlüssen zu. Um den Anschlussblock zu definieren, öffnet man das Popup-Menü des Symbolfeldes und wählt *Anschluss anzeigen* (Show Connector). Soll einem Bedien- oder Anzeigeelement ein Anschluss zugewiesen werden, kann man folgendermaßen vorgehen.

1. Klick auf einen Anschluss des Anschlussblocks. Der Cursor wird automatisch zum Verbindungswerkzeug und der Anschluss wird schwarz.
2. Klick auf ein Bedien- oder Anzeigeelement an, das mit dem Anschluss verknüpft werden soll.
3. Klick in einen freien Bereich des Frontpanels. Die gepunktete Linie verschwindet und erhält die Farbe des zugehörigen Datentyps, um anzuzeigen, dass die Zuweisung des Bedien- oder Anzeigeelements zu dem Anschluss notwendig war. Im Hilfefenster werden die Anschlüsse darsgestellt.

Beispiel Erstellen eines SubVI für Wechselspannungsmessungen

Das AC+Graph-Voltage.vi soll in ein SubVI umgewandelt werden, so dass es im Diagramm eines anderen VI eingesetzt werden kann.

- Öffne das AC+Graph-Voltage.vi.
- Erstelle ein Symbol für das VI. Öffne das zugehörige Popup-Menü und wähle *Symbol bearbeiten ...* (Edit Icon ...). Hierzu kann in Google-Bilder evtl. eine Vorlage gefunden werden (z. B. ein Symbol für das Voltmeter)
- Erstelle den Anschlussblock, indem du im Front Panel im Symbol *Anschluss anzeigen* (Show Connector) auswählst. Man kann dann mehrere kleine Rechtecke im Symbol erkennen. Wähle den rechten Anschluss und weise ihm die Analoganzeige zu. Verwende dazu das Verbindungs-Werkzeug. Klicke auf den Anschluss im Anschlussblock (er wird schwarz) und dann auf die Analoganzeige. Zum Schluss klicke auf einen freien Bereich im Frontpanel. Der gewählte Anschluss färbt sich orange.
- Wähle aus dem Popup-Menü des Symbols mit einem rechten Mausklick *Symbol anzeigen* (Show Icon), um zum Symbol zurückzukehren.
- Speichere die Änderungen.
- Verwende das Positionier-Werkzeug, um einen Bereich des Blockdiagramms zu markieren, der als SubVI dienen soll. LabVIEW ist ziemlich intelligent, um den User soweit zu unterstützen, dass korrekte Eingaben und Ausgaben realisiert werden können. Am besten versucht man erst mal das ganze Block-Diagramm zu markieren. Dann wählt man den Befehl *Bearbeiten » SubVI erstellen* (Edit » Create SubVI), um das SubVI automatisch zu erstellen.
- Es wird beim Sichern das bearbeitete VI als SUB-VI abgelegt und susätzlich ein Modul, das man sinnvollerweise als SUB-VI-Modul bezeichnet.

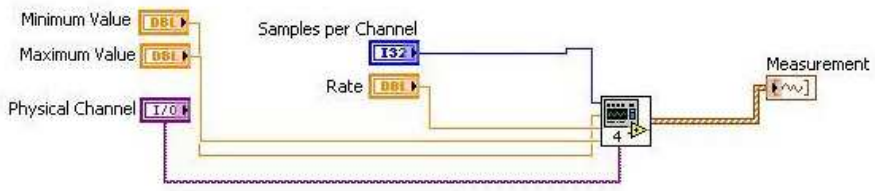


ABB.2.1

Jetzt kann in ein neues VI über das Blockdiagramm mit *All Functions - Select a VI ...* das erstellte SubVI als Modul geladen und verdrahtet werden.

Übung Erstelle ein Wechselspannungsmessung unter Verwendung des AC+Graph-Voltage SubVIs.

3. Fehlerbeseitigung

Sobald man eine gebrochen dargestellte *Start* (Run)-Schaltfläche erkennt, bedeutet das, dass das Programm nicht ausgeführt werden kann. Klickt man auf dieses gebrochene Symbol, werden alle Fehler angezeigt.

3.1. Einzelschrittmodus

Um einen Fehler aufzuspüren kann man das Blockdiagramm Schritt für Schritt ausführen. Dazu wird das VI durch eine *Einzelschritt*-Schaltfläche (Start Single Stepping) gestartet. Die erste Taste ist zum Einsprung ins VI, die zweite Taste zum Durchführen einer Sequenz und die dritte Taste zum Beenden der Struktur. Die erste und zweite Taste haben oft die gleiche Funktion beim Einzelschrittverfahren. Wird noch die *Highlight Funktion* (Highlight Execusion) angeklickt, dann lassen sich die Bewegungen der Daten durch Blasen (Bubbles) in den Verbindungen anzeigen.

3.2. Zwischenergebnisse

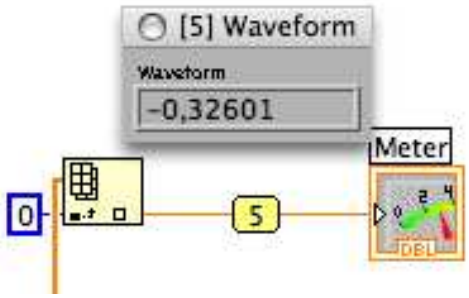


ABB.3.1

Manchmal wundert man sich über die Resultate und hätte gerne die Zwischenergebnisse in einem VI überprüft. Wähle dazu mit der rechten Maustaste die *Sonde* (Probe) aus.

Hilfreich sind auch sogenannte Haltepunkte (Set Breakpoint), an denen das Programm stoppt und die Pause-schaltfläche aktiviert. Danach kannst du das Programm im Einzelschrittmodus ablaufen lassen, Verbindungen wieder mit einer Sonde abtasten oder Werte in Frontpanelobjekten ändern.

4. Ablaufstrukturen

In LabVIEWgibt es vier verschiedene Ablaufstrukturen: die While-Schleife, die For-Schleife, die Case-Anweisung und die Sequenz-Struktur.

4.1. Schleifenstrukturen

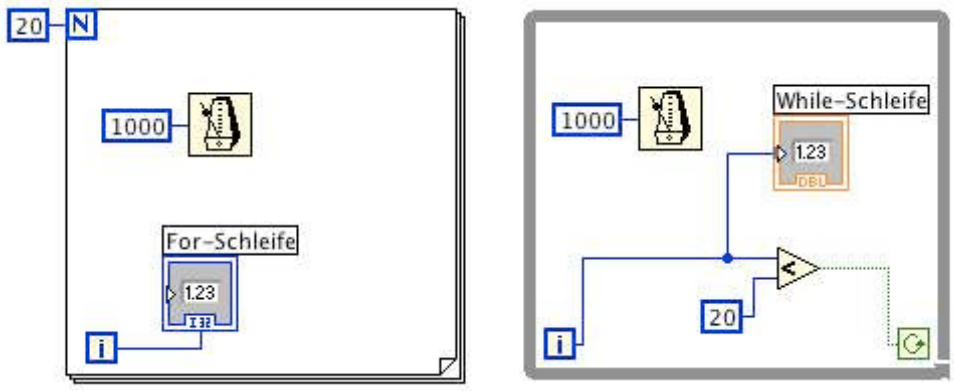


ABB.4.1

ABB.4.2

4.1.1. While-Schleife

Eine While-Schleife besteht in LabVIEWaus einem Rahmen. Das Blockdiagramm im Rahmen wird einmal ausgeführt. Danach wird die Schleifenbedingung, die im Rah-

men unten rechts einen booleschen Wert als Eingang erwartet überprüft. Die Stopp-Bedingung kann auf True oder False eingestellt werden. Mit Rechtsklick auf die Stopp-Bedingung lässt *Stop if True* oder *Continue if True* auswählen. Da die Schleifenbedingung erst am Ende einer Iteration überprüft wird, wird eine While-Schleife wenigstens einmal durchlaufen.

Der zweite Anschluss innerhalb der While-Schleife befindet sich links unten im Rahmen: der Iterationszähler i. Er entspricht der bisherigen Anzahl der Iterationen. Der Zähler startet bei 0.

4.1.2. For-Schleife

Eine For-Schleife führt alle Anweisungen, die innerhalb des Rahmens enthalten sind, so oft aus, wie das Zählerterminal *N* angibt, das von außen mit einer Zahl versorgt wird. Wie die While-Schleife hat auch die For-Schleife einen Iterationszähler. Wird eine 0 an das Zählerterminal angelegt, wird die Schleife nicht ausgeführt.

Um den Schleifenablauf verfolgen zu können, zieht man aus *All Functions - Time & Dialog* den Befehl *Wait Unit Next ms Multiple*. Auf dem linken Ausgang *millisecond multiple* kann eine ganzzahlige Konstante angegeben werden, welche die Anzahl von Millisekunden angibt, die verstreichen sollen, ehe weitere Aktionen ablaufen. Den Schleifenablauf kann man aber auch mit der *Highlight Execution* durchgeführt werden.

4.1.3. Schieberegister

Manchmal ist es notwendig, innerhalb einer Schleife Daten aus einem vorausgegangenen Schleifendurchlauf zu verwenden. Dies kann mit sogenannten Schieberegister realisiert werden, welche sowohl für While- wie auch für For-Schleifen zur Verfügung stehen. Ein Schieberegister besteht aus einem Anschlusspaar, das sich direkt gegenüber an den senkrechten Schleifenbegrenzungen befindet. Der rechte Anschluss speichert die Daten, sobald der Schleifendurchlauf beendet ist. Diese Daten werden dann zum linken Schieberegisteranschluss geschoben, damit sie beim nächsten Durchlauf der Schleife wieder zur Verfügung stehen. Das Schieberegister auf der linken Seite wird zu Beginn initialisiert. Klickt man auf den linken Schieberegistereingang, dann kann man weiter Eingänge (*Add Element*) hinzufügen.

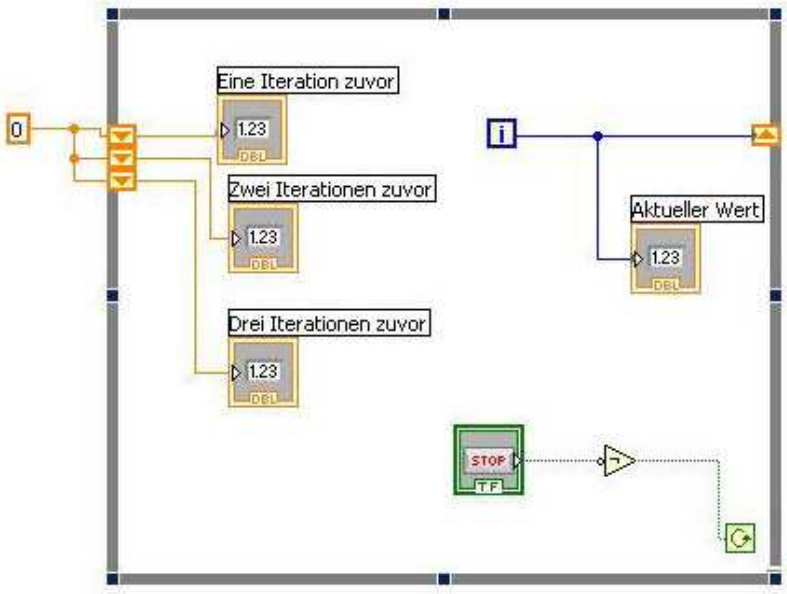


ABB.4.3

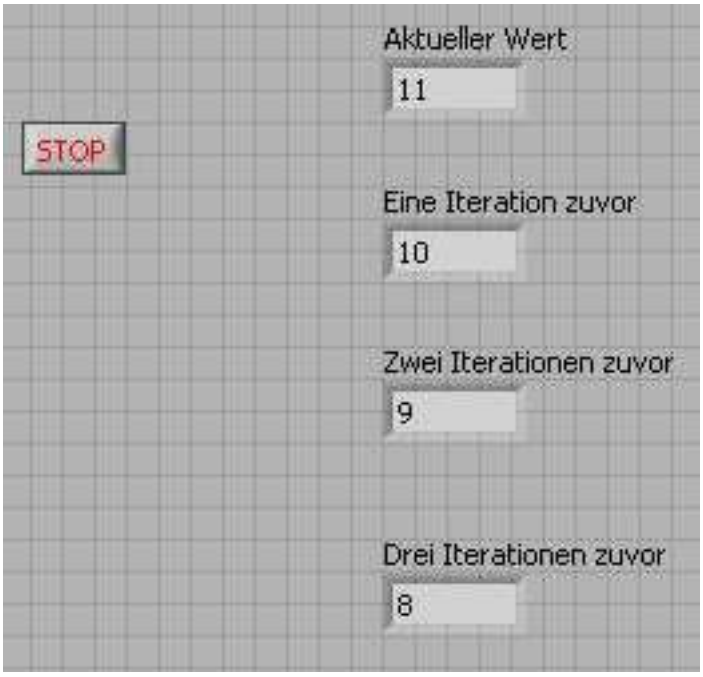


ABB.4.4

Werte des Schieberegisters auf der rechten Seite werden dann durch die linken Register nach dem FIFO-Prinzip (First in, First out) geschleift. Jeder aktuelle Wert am rechten Ausgang wird mit dem Beginn der nächsten Wiederholung an den linken Anschluss (Eine Iteration zuvor) geschoben. Die Werte am linken Anschluss rutschen zuvor um

ein Register nach unten. Damit ist der aktuelle Wert an oberster Stelle.

4.1.4. Case-Struktur

Die Case-Anweisung ermöglicht die Ausführung des Datenflusses in Abhängigkeit vom Ergebnis einer Berechnung oder vom Zustand eines Eingangswertes. Die Case-Anweisung ist im Blockdiagramm durch einen Rahmen mit einem Auswahlmenü am oberen Rand zu erkennen. Am linken Rahmenabschnitt weist ein Fragezeichen auf deinen Auswahlanschluss hin, der angibt welcher Zweig durchlaufen wird. Die unterschiedlichen Case-Verzweigungen liegen übereinander, so dass man immer nur eine sieht. Durch Klicken auf die Links- bzw. Rechtspfeile zeichen sich die verschiedenen Auswahlen.

Übung

Baue ein VI mit einer Case-Struktur auf, um die Quadratwurzel einer positiven Eingangszahl zu ermitteln. Handelt es sich um eine negative Zahl, soll sich im VI ein Dialogfeld öffnen und einen Fehler ausgeben.

- Öffne ein VI Blank
- Platziere die in der nebenstehenden Abbildung dargestellten Ein- und Ausgabeelemente auf dem Frontpanel.

ABB.4.5

- Wechsle ins Blockdiagramm und erstelle das folgende Diagramm.

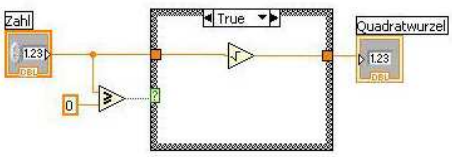


ABB.4.6

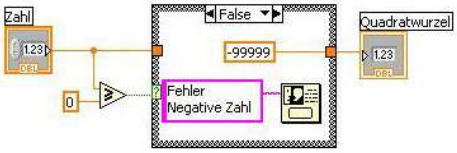


ABB.4.7

Die Funktion *One Button Dialog* in der Unterpalette *Time & Dialog* öffnet ein Dialogfeld, das eine Meldung ausgibt. Diese Meldung bleibt solange erhalten, bis der OK-Button angeklickt wird.

- Teste das VI.

4.1.5. Sequenz-Struktur

Mit der Sequenz-Struktur ist es möglich, die Reihenfolge der Abarbeitung der Programmteile festzulegen. Diese Struktur sieht aus wie ein Filmstreifen und führt zuerst Rahmen 0, dann Rahmen 1, dann Rahmen 2, ... aus. Erst wenn der letzte Rahmen ausgeführt wurde, verlassen die Daten die Struktur.

Die Sequenz befindet sich in der Unterpalette Functions-Structures. Wie bei Case wird jeweils nur ein Rahmen zur gleichen Zeit angezeigt. Durch rechten Mausklick auf den Strukturrahmen lässt sich ein Rahmen einfügen (*Add Frame After* oder *Add Frame Before*).

Wie leitet man Daten von einem Sequenzrahmen in einen der darauffolgenden weiter? Hierfür stellt LabVIEW die lokale Sequenz-Variable zur Verfügung. Um diesen Anschluss zu bekommen, wählt man mit einem Rechtsklick auf den Sequenzrahmen *Add Sequence Local*. Sobald Daten an die lokale Sequenz-Variable angeschlossen werden, erscheint ein auswärts gerichteter Pfeil in dem Rahmenanschluss, der die Datenquelle enthält. Die Anschlüsse der folgenden Rahmen erhalten einen nach innen gerichteten Pfeil. Damit wird deutlich, dass der Anschluss Daten für den jeweiligen Sequenzrahmen abgeben kann.

Übung

Erstelle ein VI, das die Zeit berechnet, die vergeht, bis eine eingegebene Zahl mit einer zufällig erzeugten Zahl übereinstimmt.

- Öffne ein VI Blank
- Platziere die in der nebenstehenden Abbildung dargestellten Ein- und Ausgabeelemente auf dem Frontpanel.

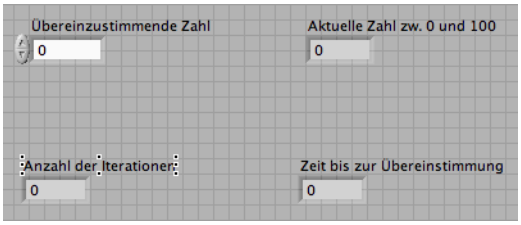


ABB.4.8

- Wechsle ins Blockdiagramm und erstelle die folgenden Diagramm.
  - ▷ Wähle eine *Stacked Sequence Structure*. Füge in den 1. Rahmen ein *Tick Cound (ms)* ein und klicke auf den Rahmen mit der rechten Maustaste und erzeuge ein *Add Sequence Local*. vgl. Abb. 4.9
  - ▷ Verbinde die Uhr mit dieser lokalen Sequenzvariablen.
  - ▷ Wähle *Add Frame After* nach Rechtsklick auf den Rahmen und füge nach Abb. 4.10 die Objekte an. In einer While-Schleife werden Zufallszahlen zwischen 0 und 100 erzeugt und mit einer frei gewählten Zahl verglichen. Die aktuelle Zufallszahl und die Anzahl der Iterationen wird ausgegeben.

▷ Füge ein weiteres Frame an und ergänze mit den in Abb. 4.11 dargestellten Objekten, mit welchen die Zeitspanne bestimmt wird.

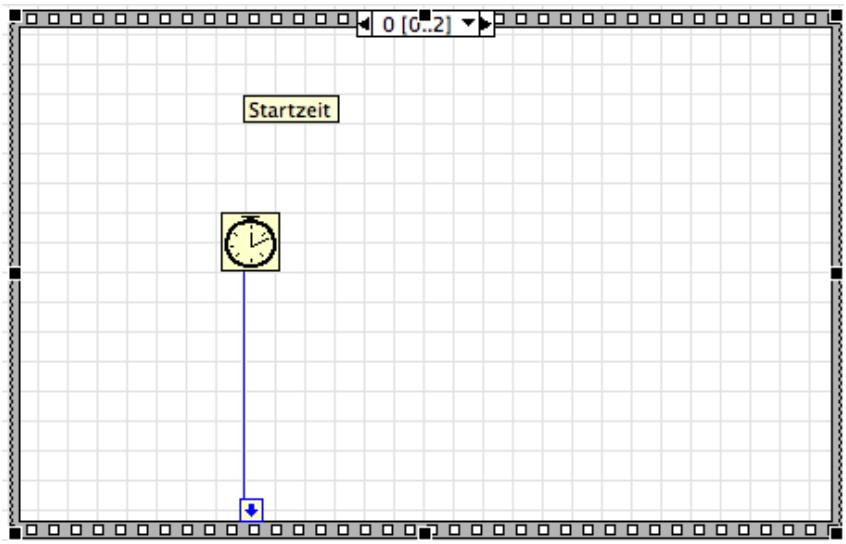


ABB.4.9

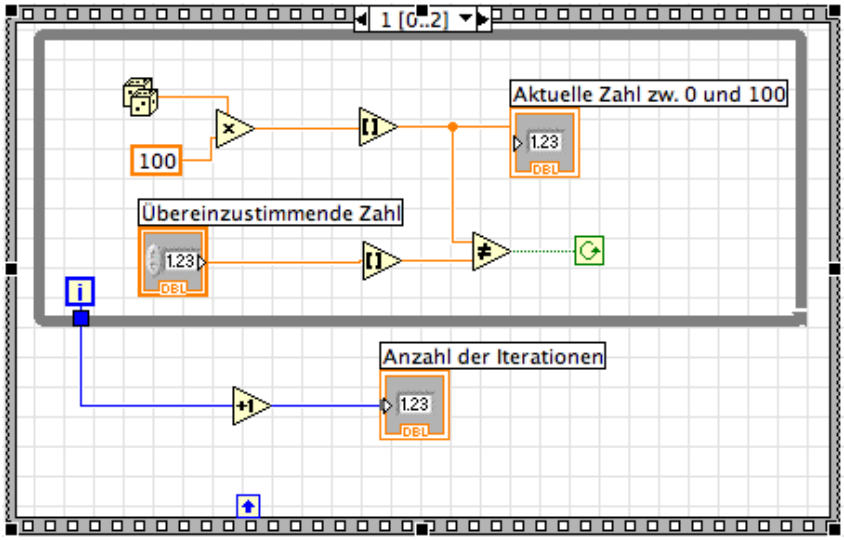


ABB.4.10

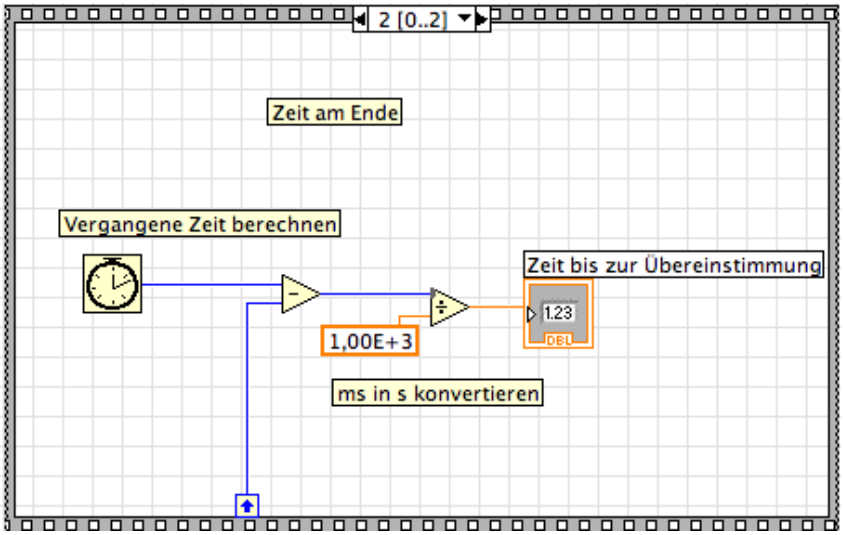


ABB.4.11

4.1.6. Formelknoten

Der Formelknoten ist ein Rahmen, in den algebraische Formeln direkt in das Blockdiagramm eingegeben werden können. Es ist manmal besser, Formeln nicht mit LabVIEW-Symbolen einzugeben, da sie bei komplexeren Termen schnell unübersichtlich werden.

Gib z. B. den Funktionsterm $y = x^3 - x$ mit Hilfe der LabVIEW-Funktionen ein. Im Blockdiagramm sieht das dann folgendermaßen aus und ist nicht so ohne weiteres zu durchschauen.

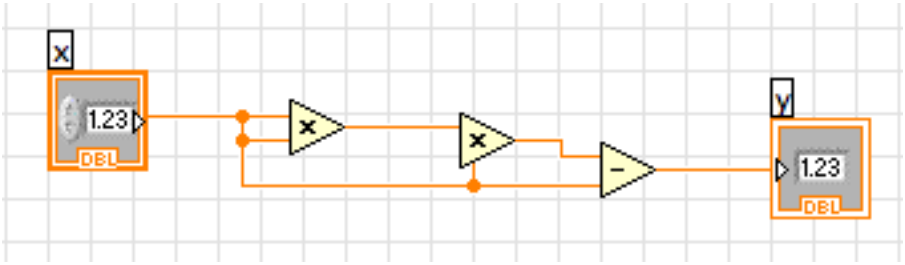


ABB.4.12

Dieselbe Funktionsgleichung kann übersichtlich in gewohnter Notation eingefügt werden.

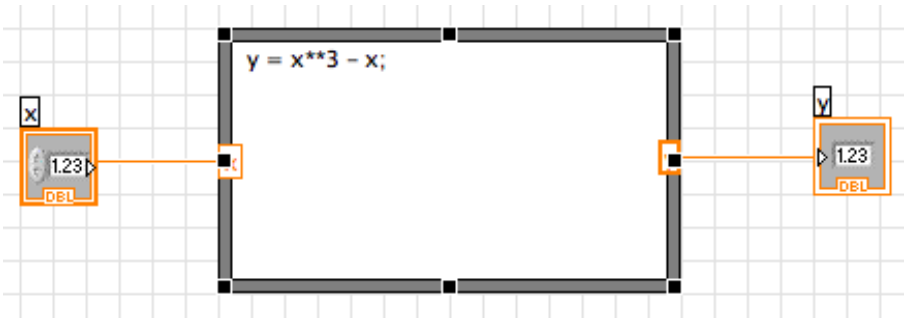


ABB.4.13

Durch Rechtsklick auf den Rand des Formelknotens kann mit *Add Input* bzw. *Add Output* der Formelknoten mit einem Eingabe- bzw. Ausgabeport versehen werden.

5. Arrays und Cluster

5.1. Arrays

Ein LabVIEW-Array ist ein Datenfeld mit gleichen Typen in den einzelnen Zellen des Feldes. Die Zellen wird über die Indizes zugegriffen, welche im Bereich von 0 bis $N-1$ liegen, wenn N die Gesamtzahl der Zellen des Arrays ist.

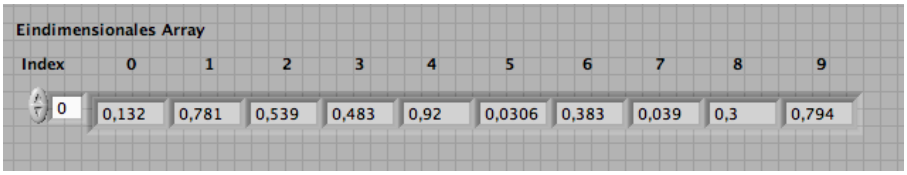


ABB.5.1 1D-Array

5.1.1. Erstellen von Arrays

Ein Array lässt sich erzeugen, indem ein Arrayobjekt *Array* aus *All Controls - Array&Cluster* ins Frontpanel geholt wird. Zu Beginn ist der Datentyp der Arrayelemente undefiniert. Wird eine Eingabe in das erste Arrayelement vorgenommen (rechte Maustaste - Auswahl der Dateneingabe), dann nimmt das Blockdiagramm des Arrays automatische die Farbe des neuen Datentyps an. Sollen mehrere Elemente gezeigt werden, vergrößert man einfach das Arrayobjekt nach rechts. Jenes Element, welches am nächsten bei der Indexanzeige liegt, hat die dort angezeigte Elementnummer.

5.1.2. Indizierung mit der For- oder While-Schleife

Mit For- oder While-Schleifen können Arrays innerhalb ihrer Bereiche automatisch elementweise angelegt werden. Dazu wird das Arrayobjekt mit dem gewünschten Datentyp versehen und die Ergebnisse der Schleifen mit dem Arrayobjekt verbunden. Bei der For-Schleife wird dann automatisch das Arrayfeld belegt. Bei der While-Schleife muss noch die automatische Indizierung eingeschaltet werden.

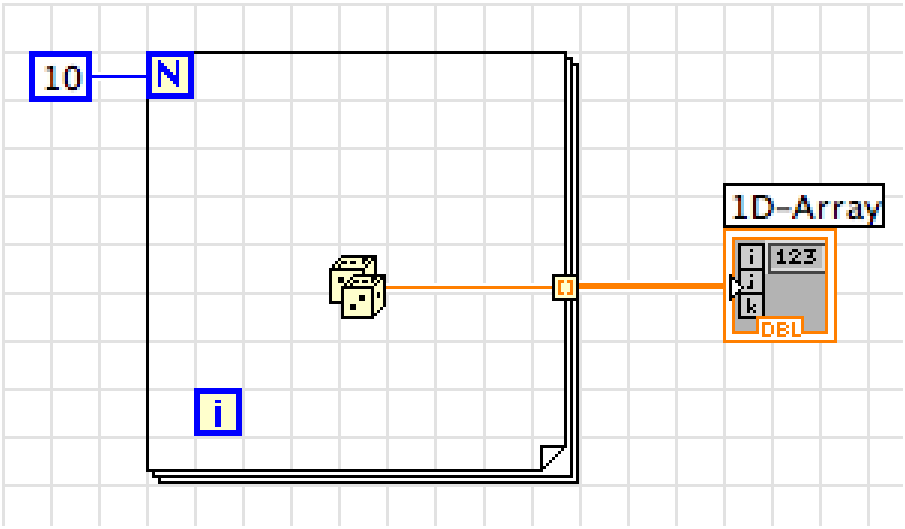


ABB.5.2 Array mit For-Schleife erzeugen

5.1.3. Zweidimensionale Arrays

Bei zweidimensionalen Arrays wird der Zugriff über einen Spaltenindex und einen Zeilenindex ermöglicht. Auch diese Indizes beginnen bei null.

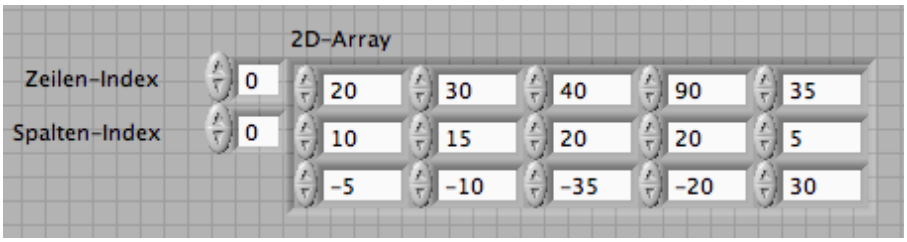


ABB.5.3 2D-Array

Ein 2D-Feld lässt sich mit Hilfe zweier ineinander geschachtelter For-Schleifen realisieren. Die innere For-Schleife erzeugt spaltenweise die einzelnen Zeilen und die äußere For-Schleife setzt diese Zeilen übereinander.

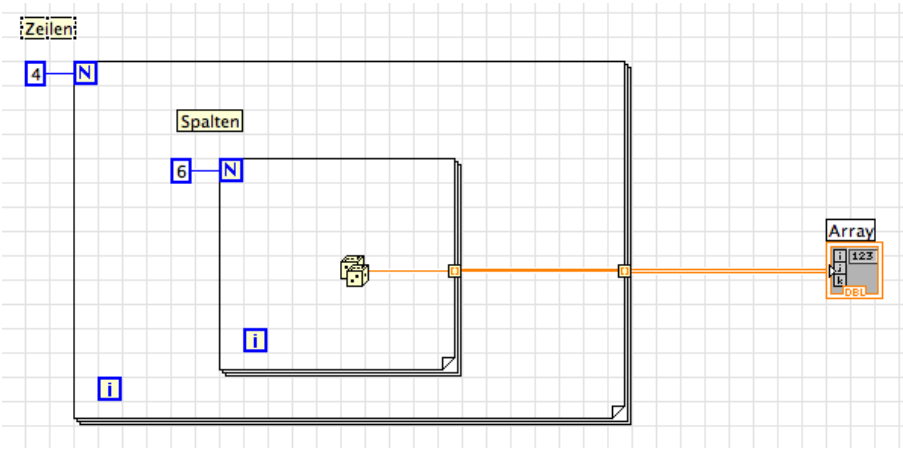


ABB.5.4 2D-Array mit For-Schleifen erzeugen

Übung

Erzeuge eine 10x10-Feld mit geschachtelten For-Schleifen und ein 1D-Array (maximale Größe 100 Elemente) mit einer While-Schleife, die mit STOP unterbrochen werden kann.

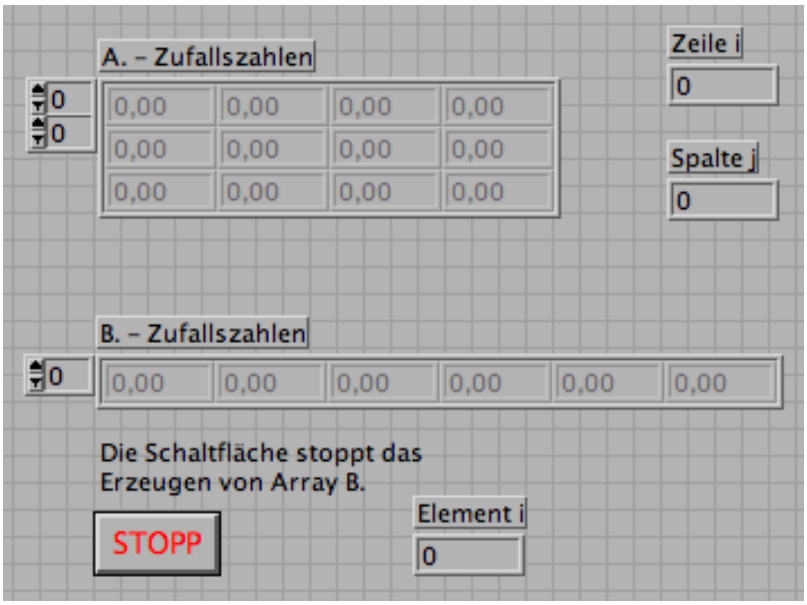


ABB.5.5

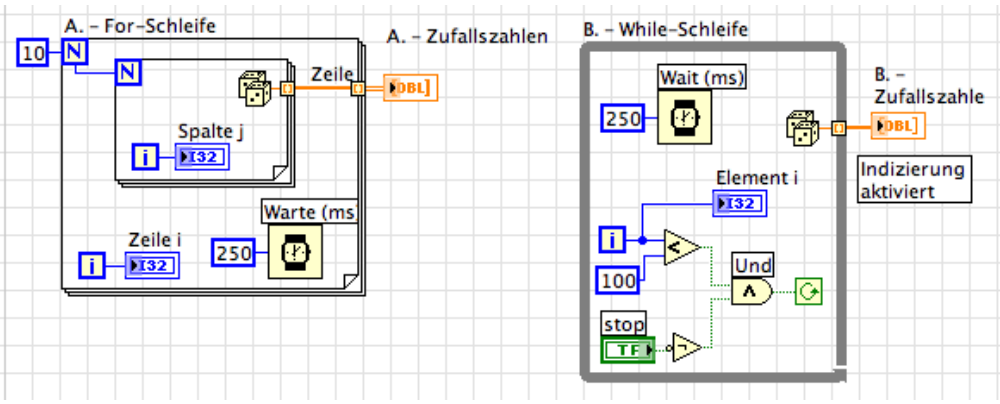


ABB.5.6

Bevor man die Array-Daten aus einer While-Schleife erhält, muss man erst am *Loop Tunnel* das *Enable Indexing* aktivieren.

5.1.4. Funktionen zur Array-Bearbeitung

In der Unterpalette *Array* von *All Functions* findet man zahlreiche Funktionen, mit denen sich Arrays manipulieren lassen. Einige wichtige Beispiele:

5.1.5. Array initialisieren

Mit *Initialize Array* lassen sich n-dimensionale Arrays mit einem beliebigen Wert initialisieren.

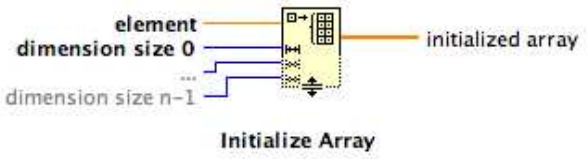


ABB.5.7

Im nebenstehenden Beispiel wir ein eindimensionales Feld von 8 Spalten mit dem Wert 10.

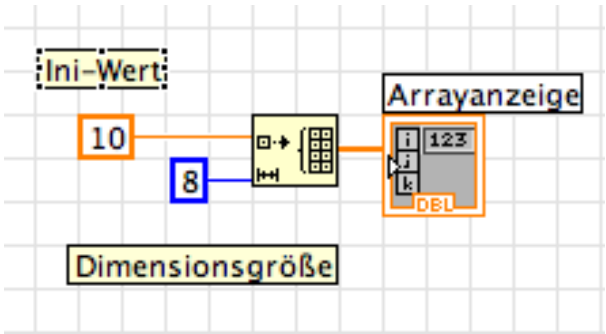


ABB.5.8

5.1.6. Array erstellen

Build Array ermöglicht es, Arrays und Elemente zu einem neuen Array zusammenzuführen. Die Funktion kann nach unten vergrößert werden, um die Anzahl der Eingänge zu erhöhen.

Im folgenden Beispiel werden zwei Arrays und ein Element zu einem neuen Array zusammengefügt. Um den Eingangstyp zu wechseln ist mit einem Rechtsklick *Concatenate Inputs* zu wählen.

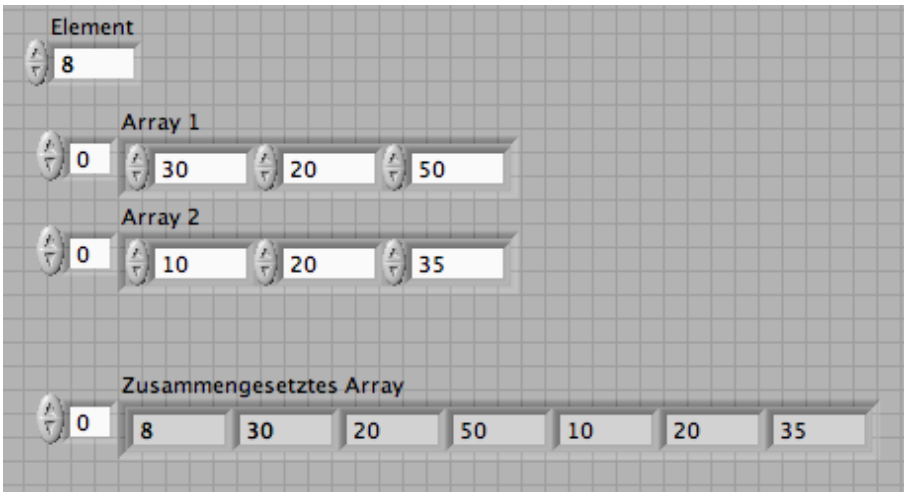


ABB.5.9

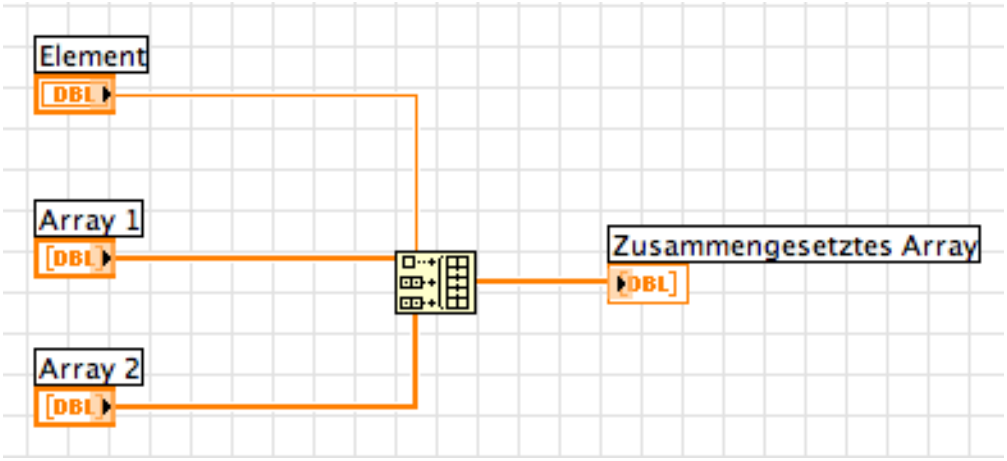


ABB.5.10

5.1.7. Teil-Array

Array Subset gibt einen Teil eines Array als sogenanntes Subarray zurück. Begonnen wird bei einem Index mit einer bestimmten Anzahl von Elementen.

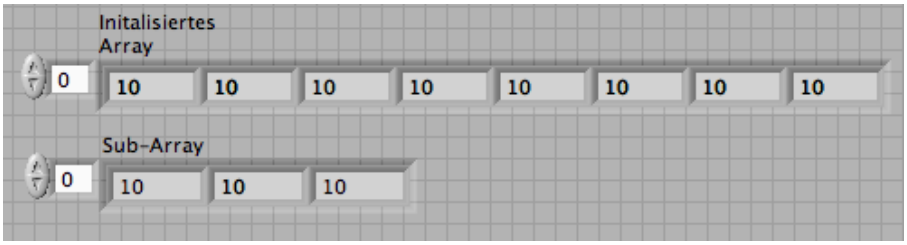


ABB.5.11

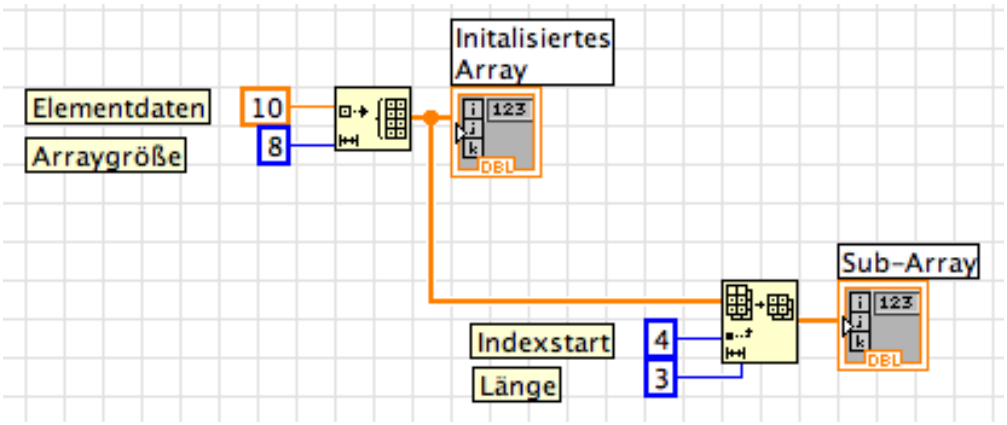


ABB.5.12

5.1.8. Polymorphie

Polymorphe Funktionen nehmen Argumente unterschiedlichen Typs an, verhalten sich aber für jede Typvariante gleich. LabVIEW interpretiert die Operationen für die verschiedenen Eingangstypen artgerecht.

Im folgenden Beispiel sollen 10 Zufallszahlen in einen Array 1, der Zähler der For-Schleife im Array 2, die Summe von Array 1 und Array 2 im Feld Array 1+Array 2 gespeichert werden. In einem Feld Array 1*Skalierung soll das Produkt aus den Elementen des Array 1 mit einem Skalierungsfaktor abglegt werden. Die Felder werden in einem Schaubild *Waveform Graph* dargestellt.

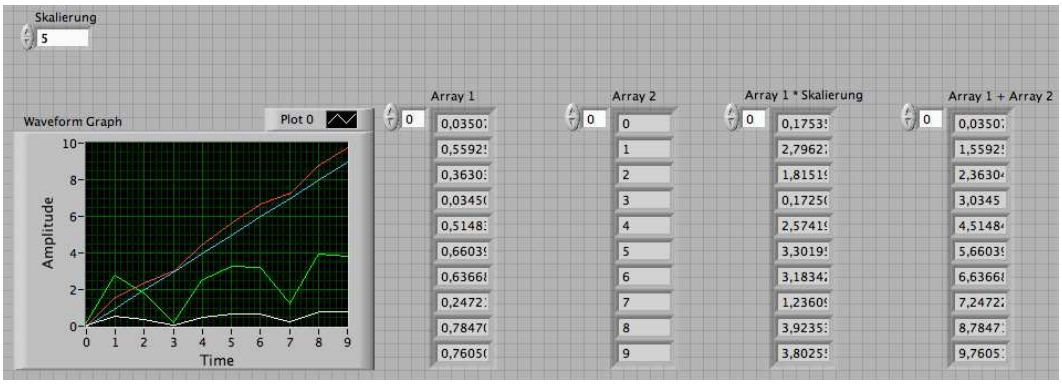


ABB.5.13

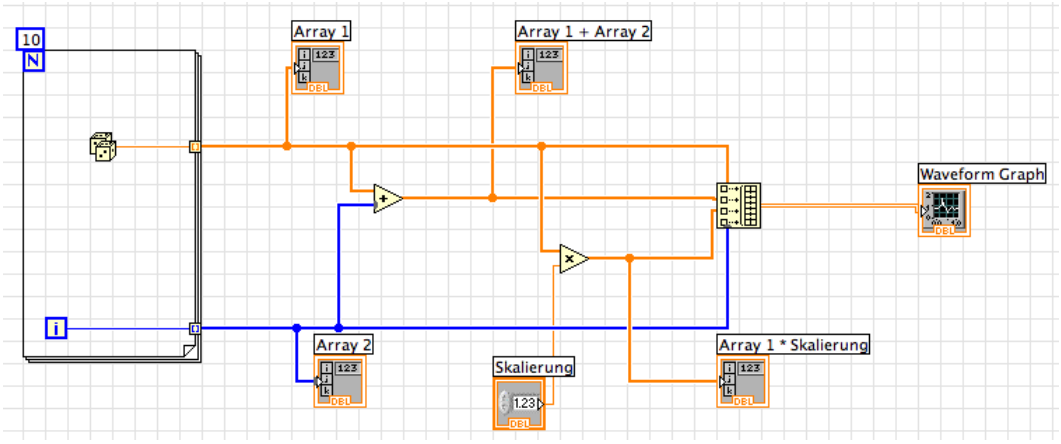


ABB.5.14

5.2. Cluster

Ein Cluster ist eine Verallgemeinerung eines Arrays und entspricht einem Struct in C++ oder einem Record in Pascal. Im Gegensatz zu Arrays kann in ein Cluster Daten unterschiedlichen Typs aufnehmen, die dann im Blockdiagramm in einer einzigen Leitung geführt werden. Damit wird das Kabelgewirr reduziert. Im Unterschied zu Arrays, die dynamisch ihre Größe ändern können, haben Cluster eine feste Größe, d. h. eine feste Anzahl von Drähten. Auch für Cluster gilt die Polymorphie, wenn die Datentypen übereinstimmen.

5.2.1. Erstellen von Clustern

Im Frontpanel kann aus der Unterpalette *Array & Cluster* ein Cluster-Objekt platziert werden. Innerhalb dieses Clusters lässt sich jedes Objekt des Frontpanels einfügen.

Cluster können entweder nur Eingabe- oder nur Anzeigeelemente enthalten, weil das Cluster entweder eine Eingabe oder eine Anzeige sein kann.

Das erste Objekt, das im Cluster platziert wird, ist das Element Null, das zweite ist das Element Eins usw. Wenn ein Cluster mit einem anderen Cluster verbunden werden soll, müssen Reihenfolge und Datentyp identisch sein.

5.2.2. Daten bündeln und Cluster-Element ersetzen

Soll ein Element in einem Cluster ersetzt werden, muss im Blockdiagramm die Funktion *Bundle* angepasst werden, damit sie die gleiche Anzahl von Eingangsanschlüssen hat, wie Elemente im Cluster sind.

5.2.3. Cluster aufschlüsseln

Die Funktion *Unbundle* zerlegt einen Cluster in jede seiner Komponenten. Die Ausgangskomponenten werden von oben nach unten in derselben Reihenfolge angeordnet, die sie auch im Cluster hatten.

Übung

In der Übung soll ein Cluster erzeugt werden. Dann soll das Cluster aufgeschlüsselt und wieder gebündelt werden und die Werte dann in einem anderen Cluster angezeigt werden.

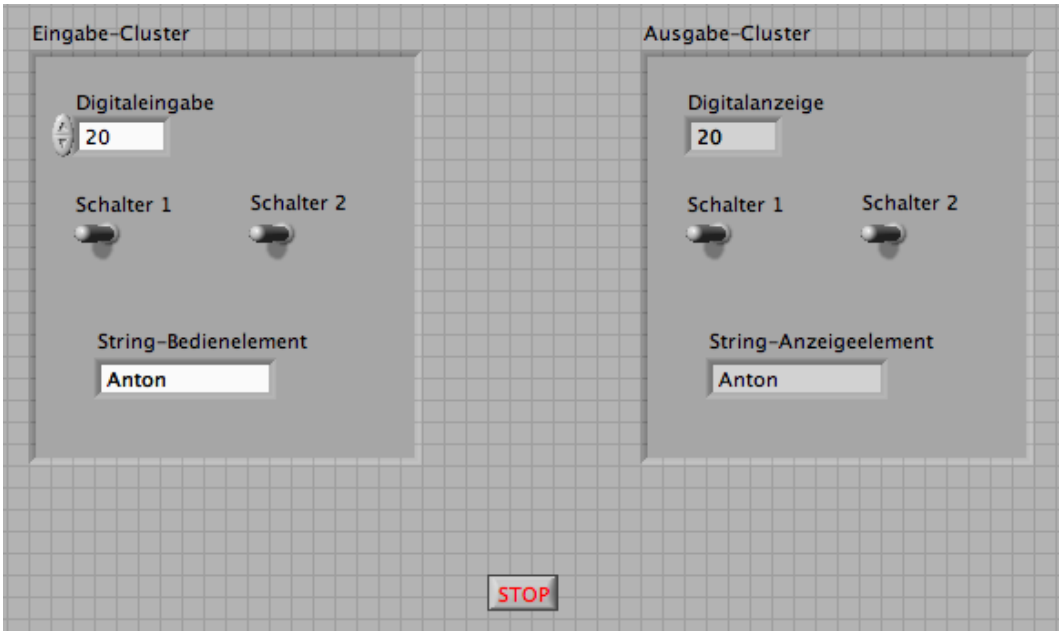


ABB.5.15

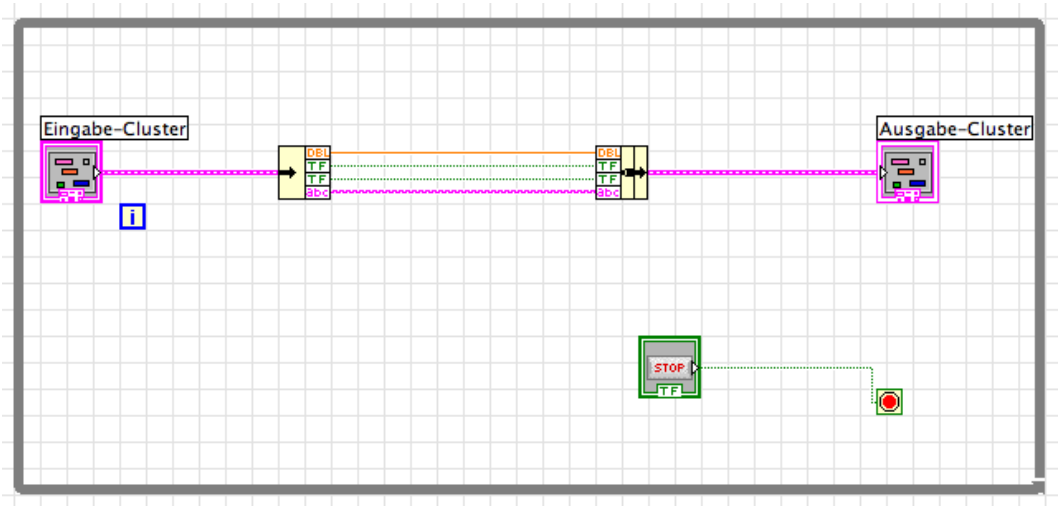


ABB.5.16

- Öffne ein neues VI-Fenster und platziere ein Cluster-Objekt und nenne es *Eingabe-Cluster*.
- Setze in das Cluster eine Zahleneingabe, zwei boolesche Schalter und eine Eingabe für einen String.
- Erzeuge ein Ausgabe-Cluster, indem du das Eingabe-Cluster kopierst und die Anzeigen entsprechend umbenennst.
- Mit *Change to Indicator* kannst du dann den Cluster vom Eingabe-Objekt in eine Ausgabe-Objekt verändern.
- Überprüfe, ob Eingabe-Cluster und Ausgabe-Cluster dieselbe Cluster-Reihenfolge haben, indem du jede Clusterbegrenzung im Frontpanel anklickst und *Reorder Controls in Cluster* aufrufst.
- Platziere eine Stopptaste im Frontpanel.
- Baue das in der Abb. 5.16 dargestellte Blockdiagramm auf (Eingabe-Cluster, Aufschlüsseln, Bündeln, Ausgabe-Cluster)

6. Grafische Darstellung von Daten

In LabVIEW ist eine Vielzahl von Visualisierungsmethoden vorhanden.

6.1. Kurvendiagramme (Wave Charts)

Die Darstellung von Messwerten in Abhängigkeit von einer Variablen wird meist mit einer sogenannten Liniengrafik durchgeführt. Man spricht von einem Kurvendiagramm (Waveform Chart). Neue Daten werden als y-Werte an die schon vorhandenen Daten in Abhängigkeit der Zeit t (x-Achse) angefügt. Das Diagramm hat drei verschiedene Aktualisierungsmodi:

- Modus für Streifenschreiber (Strip Chart)
- Modus für Oszilloskopdiagramme (Scope Chart)
- Modus für Laufdiagramme (Sweep Chart)

Diese Optionen findet man mit einem Rechtsmausklick auf das Kurvendiagramm und der Option *Advanced - Update Mode*.

Beispiel

Darstellung eines Kurvendiagramms für eine Temperaturmessung.

Um elegant mit Temperaturmessungen umgehen zu können, erstellen wir erst ein einfaches SubVI aus dem Thermometer.vi, das in den Abbildungen 6.1 und 6.2 zu sehen ist. Das „Demo Voltage Read.vi“ simuliert Spannungsverläufe von 0 bis 1 V von einer Datenerfassungskarte.

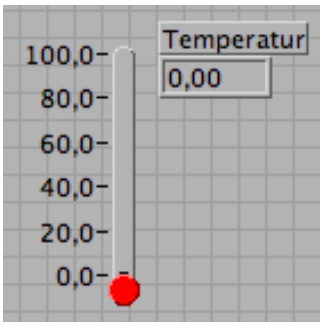


ABB.6.1

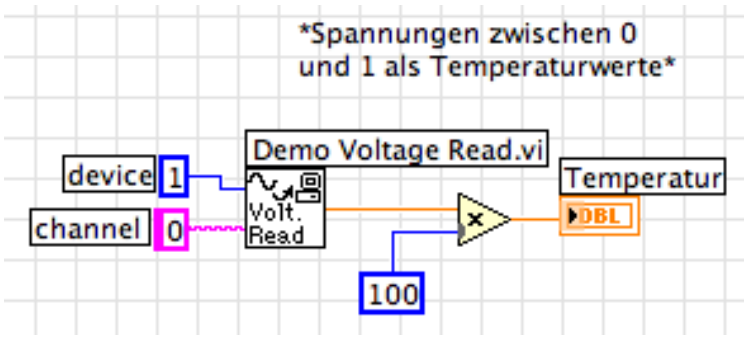


ABB.6.2

- Öffne das Thermometer.vi.
- Erstelle oder ergänze das Symbol für das SUB-VI. Öffne das zugehörige Popup-Menü und wähle *Symbol bearbeiten ...* (Edit Icon ...).
- Erstelle den Anschlussblock, indem du im Front Panel im Symbol *Anschluss anzeigen* (Show Connector) auswählst. Da das Frontpanel nur aus einer Anzeige besteht, zeigt auch der Anschlussblock nur einen Anschluss, der sich orange einfärbt.
- Wähle aus dem Popup-Menü des Symbols mit einem rechten Mausklick *Symbol anzeigen* (Show Icon), um zum Symbol zurückzukehren.
- Verwende das Positionier-Werkzeug, um einen Bereich des Blockdiagramms zu markieren, der als SubVI dienen soll. In diesem Fall sollten device, channel, Voltage Read und das Multiplikations-Objekt enthalten sein. LabVIEW ist ziemlich intelligent, um den User soweit zu unterstützen, dass korrekte Eingaben und Ausgaben realisiert werden können. Markiert man das ganze Block-Diagramm und wählt den Befehl *Bearbeiten » SubVI erstellen* (Edit » Create SubVI), um das Sub-VI automatisch zu erstellen, dann wird - wie gewünscht - ein Objekt mit einem Ausgang für die Temperaturwerte erzeugt.
- Es wird beim Sichern das bearbeitete VI als SUB-VI (vgl. 6.3) abgelegt und zusätzlich ein Modul, das man sinnvollerweise als SUB-VI-Modul bezeichnet. Das SUB-VI-Modul entspricht dem VI in Abb. 6.2.

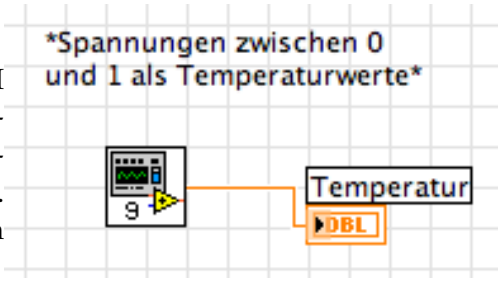


ABB.6.3

Mit dem TempSUB-VI lassen sich nun auf einfache Weise Temperaturdiagramme aufzeichnen. Im folgenden Beispiel wird mit Hilfe einer While-Schleife das Diagramm

erzeugt. Jeder Schleifendurchgang liefert einen Messwwert, der direkt im Diagramm dargestellt wird.

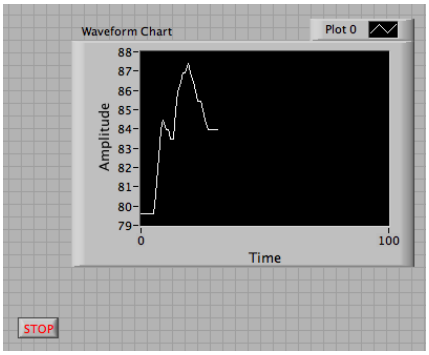


ABB.6.4

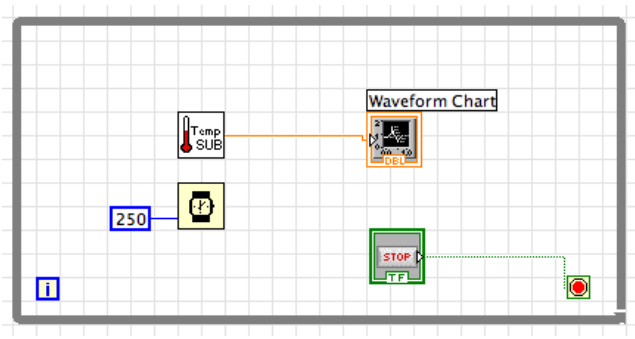


ABB.6.5

Sollen mehrere Temperaturkurven aufgenommen werden, müssen die Temperaturwerte in einem Cluster zusammengefasst werden (Cluster - Bundle). Dann können sie in einem Kurvendiagramm aufgezeichnet werden.

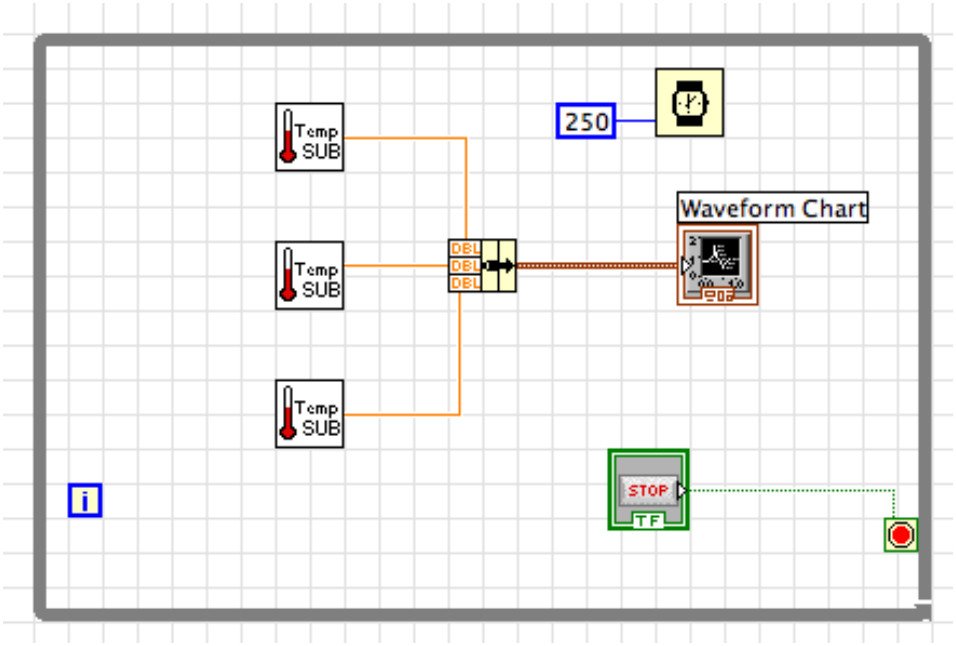


ABB.6.6

Diese Diagramme können alle auf derselben y-Achse angezeigt werden (Overlay Plots) oder jedes Diagramm auf einer eigenen y-Achse (Stacked Plots).

6.2. Graphen

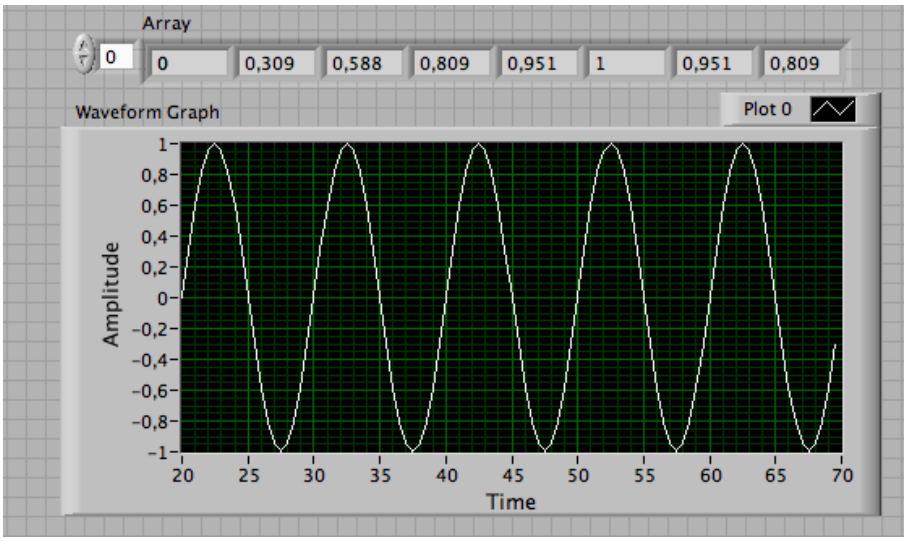
Im Gegensatz zu Diagrammen, die einen Messwert nach dem anderen zeichnen, stellen Graphen Daten aus bereits erzeugten Arrays in einem Zug dar. LabVIEW stellt drei Typen für Graphen zur Verfügung:

- Kurvengraphen
- XY-Graphen
- Intensitätsgraphen

6.2.1. Kurvengraphen

Die 1D-Array-Werte aus einer For-Schleife können direkt mit dem Anschluss eines Kurvengraphen verbunden werden. Jede Wiederholung der For-Schleife erzeugt einen Punkt der Kurve und speichert seinen y-Wert im Kurven-1D-Array, das an der For-Schleifengrenze erzeugt wird. Bei dieser Verbindung ist der Startwert von x automatisch null und x wird immer um eins erhöht, d. h. $\Delta x = 1$. Sollen x und Δx veränderbar sein, dann muss ein Cluster zwischen das 1D-Array der For-Schleife und den Kurvengraphen gesetzt werden.

Beispiel Sinuskurve im *Wave Graph* darstellen



Front

ABB. 6.7

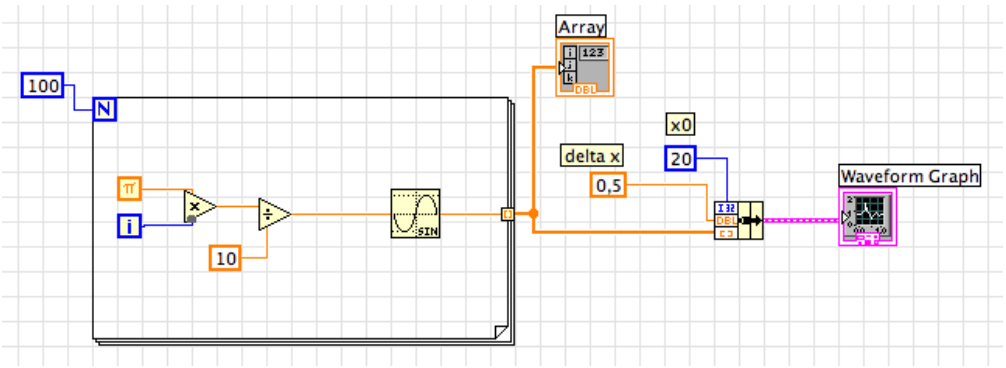


ABB. 6.8

- Öffne ein neues VI und platziere einen Kurvengraphen (*Wave Graph*) ins Frontpanel. Passe die Sklaierung auf den Wertebereich der Sinusfunktion an (*Properties - Scales*).
- Erstelle das in Abb. 6.8 abgebildete Blockdiagramm. Die Sinusfunktion $\sin(x)$ berechnet die Sinuswerte, wobei x im Bogenmaß angegeben sein muss. Im Blockdiagramm werden die x-Werte als 0, $0{,}1\pi$, $0{,}2\pi$, $0{,}3\pi$, $0{,}4\pi$, ..., 10π vorgegeben.

  Sobald die Schleife mit ihren Durchgängen fertig ist, bündelt die Funktion *Bundle* den Anfangswert von x, den Δx-
- Wert und das 1D-Array für die graphische Darstellung. Aus dem Kontextmenü ist zu erkennen, dass das *Waveform Graph*-Objekt dies verlangt.

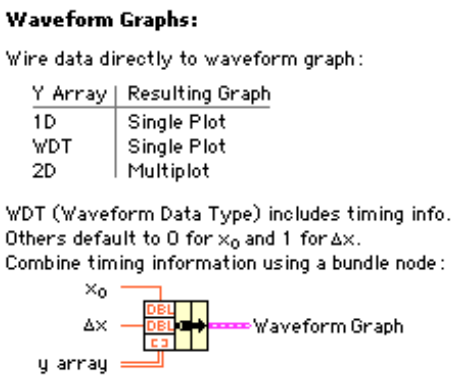


ABB. 6.9

- Ändere den Δx-Wert auf 0,5 und den Anfangswert auf 20 und starte erneut das VI.

- Wird für den x-Wert null und für den Δx der Wert eins gesetzt, dann kann das Kurvenarray aus der For-Schleife direkt an den Anschluss des Kurvengraphen angeschlossen werden.

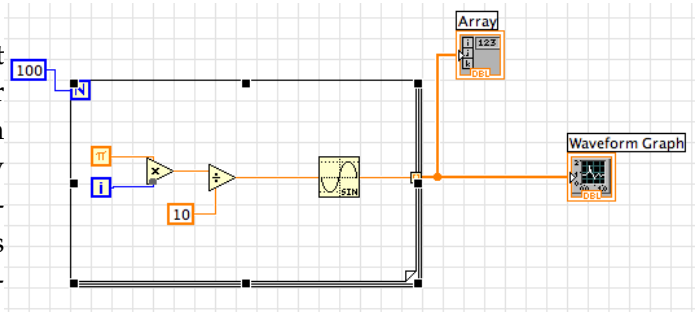


ABB.6.10

- Sollen zwei Funktionen gleichzeitig dargestellt werden, dann muss aus den 1D-Arrays der Forschleife ein 2D-Array zwischen Forschleife und Waveform Graph geschaltet werden. *Waveform Graph* nimmt dann jeweils eine Zeile aus dem 2D-Array und zeichnet die Schaubilder.

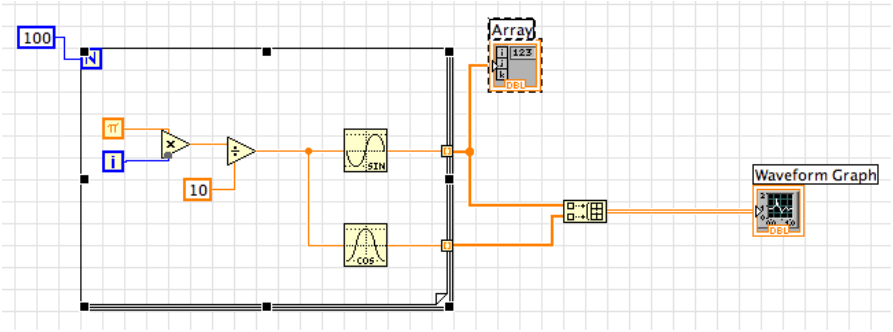


ABB.6.11

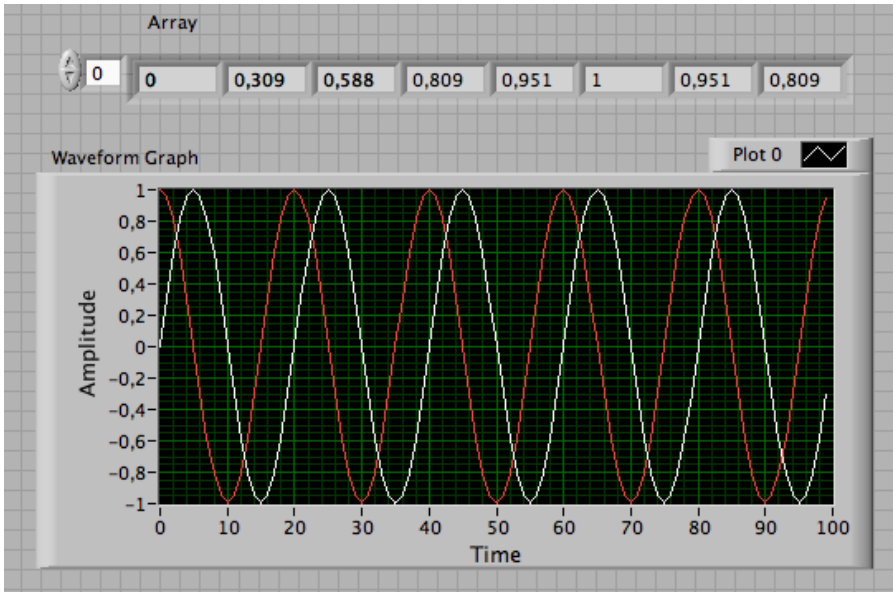


ABB.6.12

6.2.2. XY-Graphen

Soll z. B. ein Kreis vom Radius r gezeichnet werden, dann gilt $x = r \cdot \cos x$ und $y = r \cdot \sin x$. Die x-Werte sind nicht mehr äquidistant. In LabVIEW kann die Darstellung dann mit einem sogenannten XY-Graphen erfolgen. Das XY-Objekt erwartet am Eingang ein gebündeltes X-Array (oberer Eingang) und Y-Array (unterer Eingang). Die Funktion *Bundle* fasst das X- und Y-Array zu einem Cluster zusammen, der an das XY-Graph-Objekt angeschlossen wird. Bei Mehrfachdarstellungen muss für jede Darstellung ein Cluster gebildet und dann in ein 2- oder mehrdimensionales Array überführt werden.

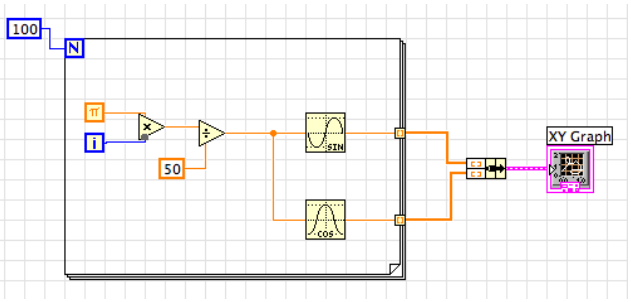


ABB.6.13

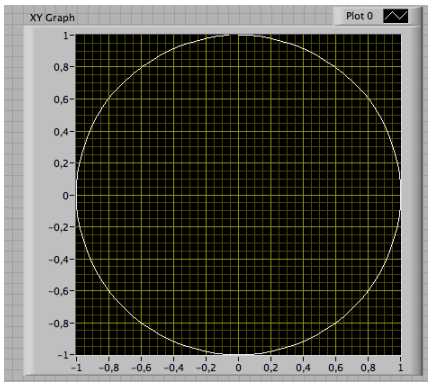


ABB.6.14

6.2.3. Intensitätsgraph

Müssen mehrere unabhängige Variable berücksichtigt werden, ist oft eine Darstellung als Intensitätsdiagramm oder Intensitätsgraph anschaulicher. Im LabVIEW-Beispiel HeatEquation.vi ist eine typische Anwendung für einen Intensitätsgraphen dargestellt.

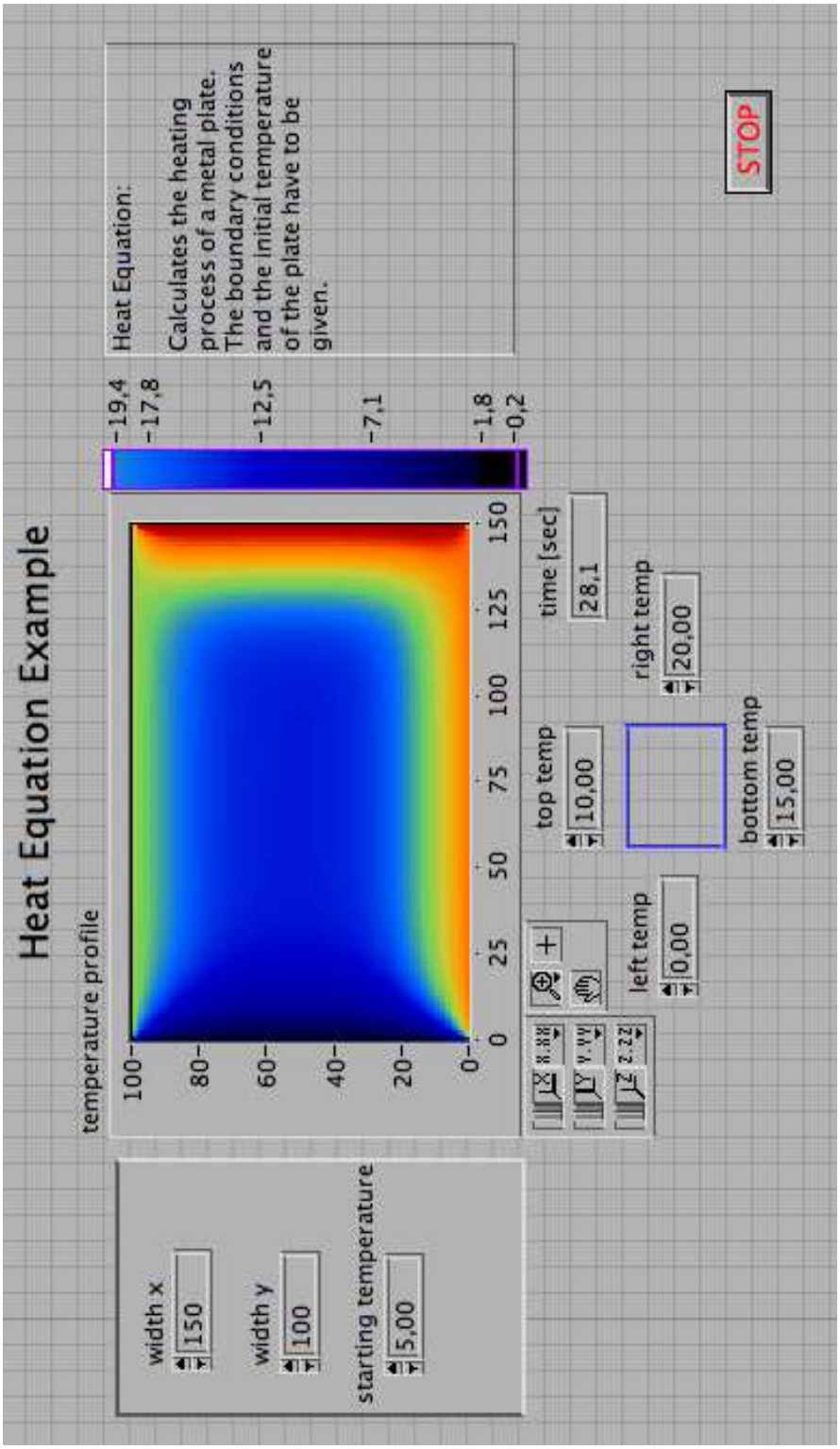

Heat Equation Example
temperature profile
Heat Equation:
Calculates the heating process of a metal plate. The boundary conditions and the initial temperature of the plate have to be given.
-19,4
-17,8
-12,5
-7,1
-1,8
-0,2
100
80
60
40
20
0
0
25
50
75
100
125
150
width x
150
width y
100
starting temperature
5,00
top temp
10,00
time [sec]
28,1
left temp
0,00
right temp
20,00
bottom temp
15,00
STOP

ABB.6.15

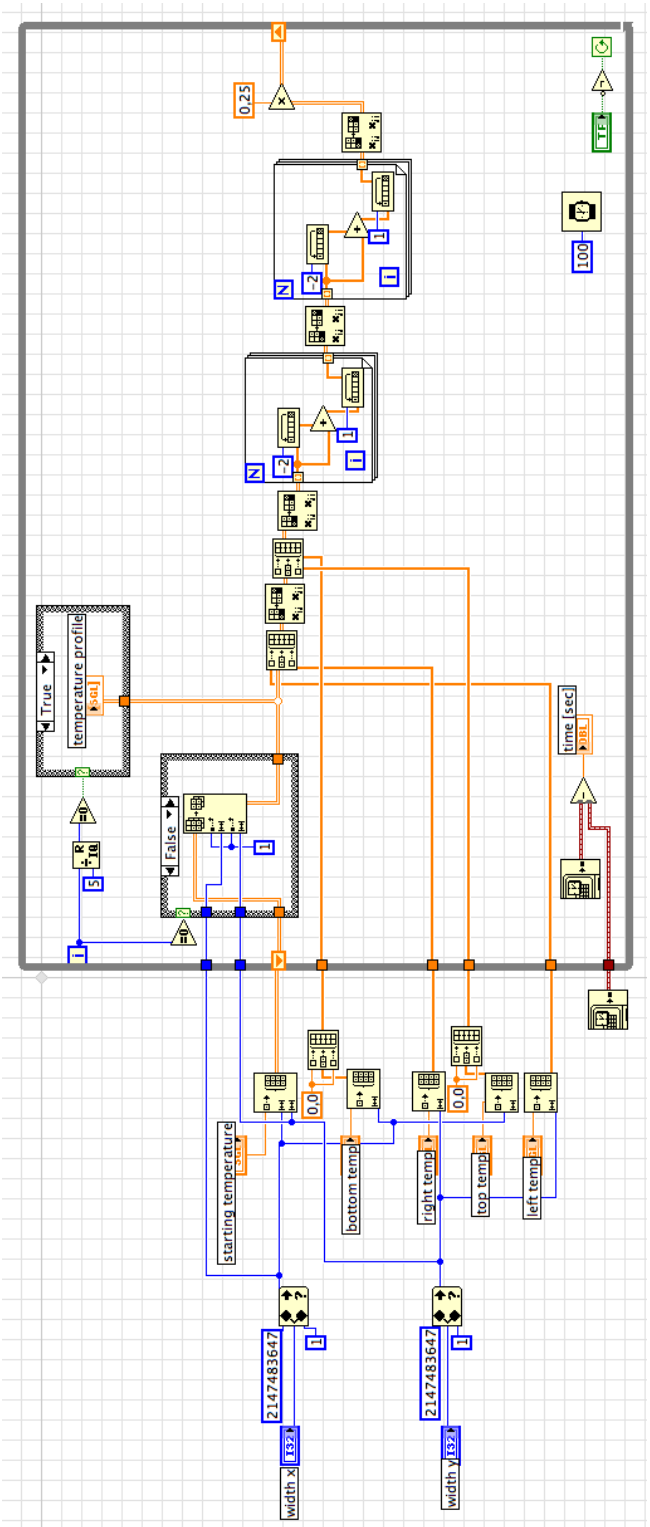

0.25
100
True
temperature profile
False
time [sec]
starting temperature
bottom temp
right temp
top temp
left temp
0.0
0.0
2147483647
2147483647
width x
width y

ABB.6.16

7. Strings und Datei I/O

7.1. Strings

Eine Zeichenkette (String) ist eine Zusammenstellung von ASCII-Zeichen. Häufig werden Daten (Text oder Zahlen) als Zeichenkette in einer Datei gespeichert. Steuerelemente für Strings werden mit einem Backslash (\) angegeben, z. B. \n für New Line (entspricht \0A), \t für Tab (entspricht \09), \s für Space (entspricht \20). Im Frontpanel können in der Unterpalette *String & Path* Control- bzw. Indicator-Strings ausgewählt werden. Wenn die Steuerzeichen eine Rolle spielen, können sie mit '\' *Codes Display* (Rechtsklick im Textfeld) angezeigt werden.

Zeichenketten haben in LabVIEW auch die Option *Password Display*-Anzeige, womit für jedes eingegebene Zeichen ein * angezeigt wird.

Strings lassen sich in LabVIEW mit verschiedenen Funktionen bearbeiten, die in der Unterpalette *String* der Funktionen des Blockdiagramms zu finden sind.

- *String Length* ermittelt die Anzahl der Zeichen in einem String.

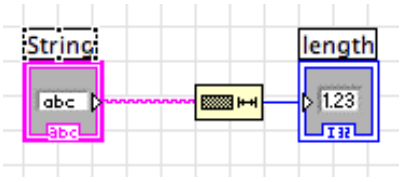


ABB. 7.1

- *Concatenate Strings* fügt verscheidene Strings zu einem einzelnen String zusammen.

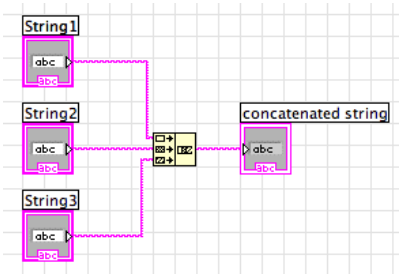


ABB. 7.2

- *Format Into String* wandelt eine Zahl in eine Zeichenkette um. Der *Initial String* wird dem Stringergebnis vorangestellt. Mit dem % Zeichen beginnt die Festlegung von *format string*. Bei der Angabe %Zahl1.Zahl2 bestimmt Zahl1 die Feldbreite und Zahl2 die Anzahl der Ziffern nach dem Komma. Ein *f* formatiert die Eingangszahl als Fließkommazahl in Dezimalschreibweise, *d* als ganze Zahl und *e* als Fließkommazahl in wissenschaftlicher Schreibweise.

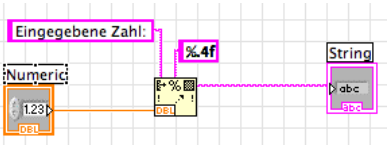


ABB.7.3

Übung

Gib eine Dezimalzahl für eine Spannungsmessung ein und wandle sie in einen String um mit 3 Nachkommastellen. Das Ergebnis soll dann in der Form „Ausgangsspannung 4,500 V“ ausgegeben werden, wenn als Spannungswert 4,5 V eingegeben wird. In einer weiteren Anzeige soll noch die Stringlänge angegeben werden.

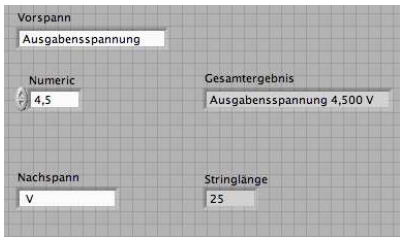


ABB.7.4

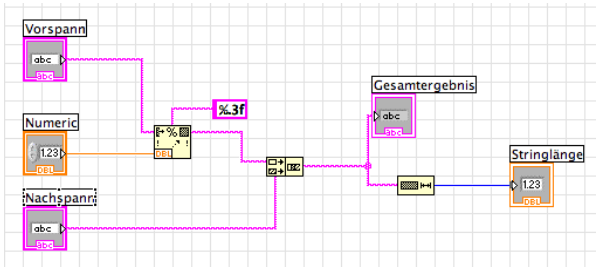


ABB.7.5

- Mit Hilfe der Parsing-Funktionen lassen sich Teilstrings in Zahlen umwandeln.
  - ▷ Mit *String Subset* lassen sich Teilstrings einer bestimmten Länge aus einem String extrahieren.

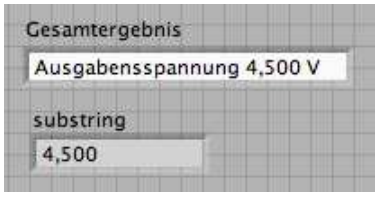


ABB.7.6

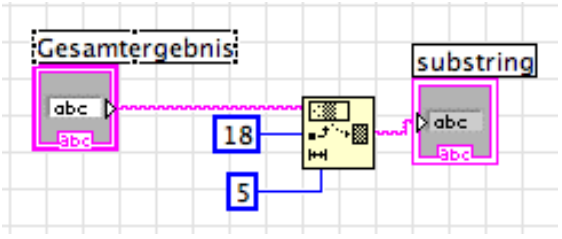


ABB.7.7

  - ▷ Mit *Scan From String* lassen sich Strings in Zahlen umwandeln.

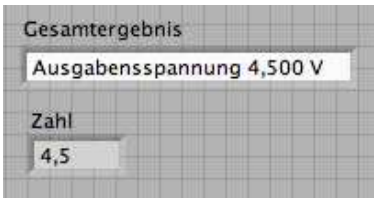


ABB.7.8

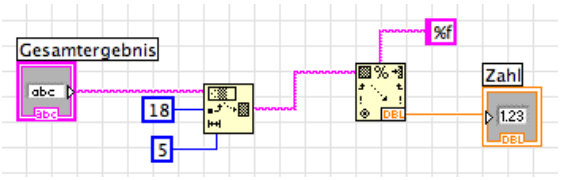


ABB.7.9

7.2. Datei-Ein- und Ausgabe

Messwerte müssen häufig in einer Datei zur späteren Bearbeitung abgelegt werden. Die Daten können als ASCII-Zeichen oder als Binärdaten abgelegt und wieder gelesen werden. In der Unterpalette *File I/O* des Blockdiagramms sind die Ein-/Ausgabe Dateifunktionen zu finden.

7.2.1. Einfache ASCII Datei Ein- und Ausgabe

Mit *Write To Spreadsheet File* bzw. *Read From Spreadsheet File* kann in eine Tabellenkalkulation geschrieben bzw. aus einer Tabellenkalkulation gelesen werden.

Write Characters To File bzw. *Read Characters From File* schreibt Zeichen in eine Datei bzw. liest Zeichen aus einer Datei.

Beispiel: Schreiben einer ASCII-Datei für eine Tabellenkalkulation

Diese Funktion konvertiert 2D oder 1D Felder mit single-precision Zahlen in Strings schreibt sie dann spaltenweise in eine Datei. Mit diesen Spreadsheed-VI wird das entsprechende File geöffnet oder neu angelegt, ehe in es geschrieben wird.

Die Daten des VI Abb. 6.11 sollen in einer Datei im ASCII-Format abgelegt werden.

- Öffne das VI 060523GraphSinCos.vi und modifiziere es so, dass die beiden Arrays in eine Datei geschrieben werden, wobei jede Spalte ein Datenarray enthält.

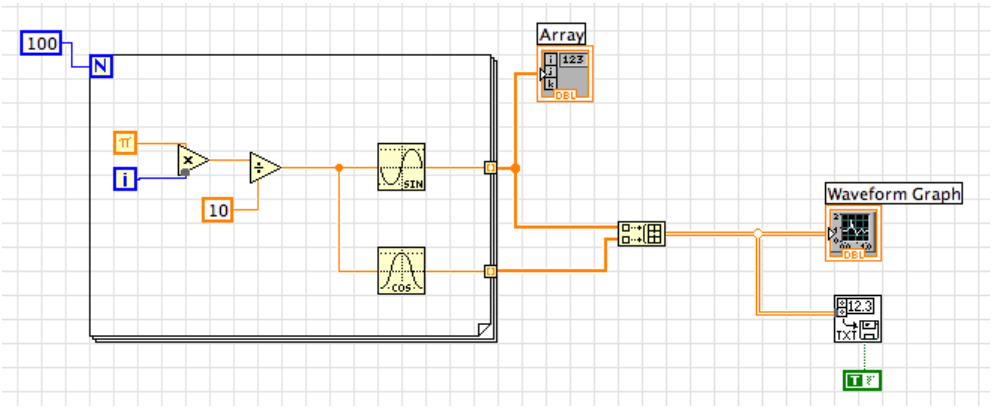


ABB.7.10

Die Funktion *Write To Spreadsheet File* konvertiert das 2D-Array in einen String für eine Tabellenkalkulation und schreibt sich in eine Datei. Die Boolesche Konstante *transpose* legt fest, ob das 2D-Array, bevor es in die Datei geschrieben wird, transponiert werden muss, d. h. von Reihen auf Spalten umgestellt werden soll. Mit einem Texteditor kann die erzeugte Textdatei SinCos gelesen werden.

Beispiel: Aus einer ASCII-Tabellenkalkulation lesen

Die Daten des VI Abb. 7.10 sind in der Datei SinCos. Diese Daten sollen aus SinCos gelesen und dann als Graph dargestellt werden.

- Öffne ein neues VI und platziere einen *Waveform Graph* im Frontpanel.
- Setze ins Blockdiagramm die Funktion *Read Characters From File*

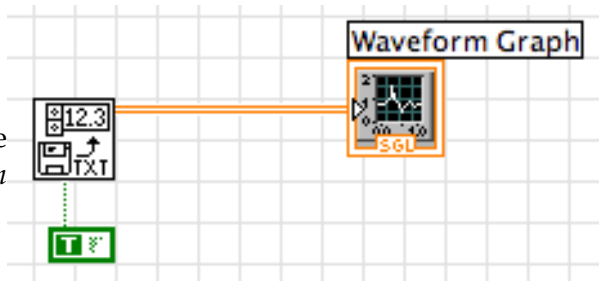


ABB.7.11

Die Funktion *Read From Spreadsheet File* konvertiert die zwei Spalten der Tabellenkalkulation wieder in ein 2D-Array.

Textdateien lesen und schreiben.

Manchmal sollen auch andere Datentypen in eine ASCII-Datei übertragen werden oder aus einer ASCII-Datei eingelesen werden. Dies kann mit den beiden VIs *Write File* und *Read File* erfolgen:

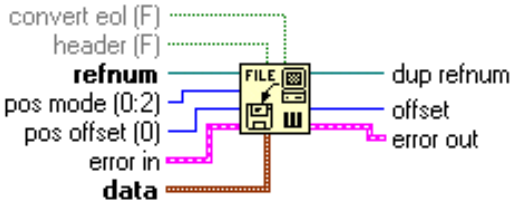


ABB.7.12 Write File

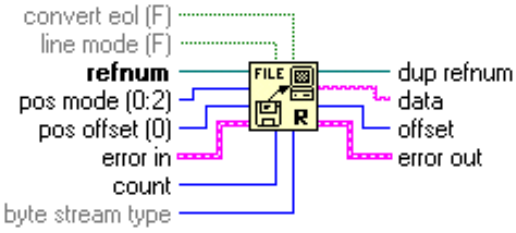


ABB.7.13 Read File

Am Eingang *Data* des VI können verschiedene Datentypen angegeben werden.

Wenn nicht auf die einfachen Spreadsheet-Funktionen zurückgegriffen wird, sind immer die drei Schritte *Open/Create/Replace File*, *Write File* bzw. *Read File* und *Close File* auszuführen. Diese Funktionen stehen in der Unterpalette *File I/O*. Nach dem *Open/Create/Replace File* wird von LabVIEW eine Dateireferenznummer (*refnum*) erzeugt, die an die Funktion *Write File* bzw. *Read File* weitergereicht. Sobald die ASCII-Zeichen in die Datei geschrieben sind, gibt *Write File* bzw. *Read File* die *refnum* an *Close File* weiter. Am Ende der Kette steht sinnvollerweise ein sogenannter *Simple Error Handler*, um evtl. Fehler anzuzeigen.

Die *refnum* wird von LabVIEW verwendet, um zu erkennen, auf welche Datei sich die jeweilige Aktion bezieht. Der User braucht sich um den Wert der *refnum* nicht zu kümmern, sondern nur darauf zu achten, dass die Datei-I/O-Funktionen mit ihnen verbunden sind. Die Funktionen werden mit der *refnum* auch in der richtigen Reihenfolge ausgeführt.

An zwei einfachen Beispielen sollen die Dateioperationen erläutert werden.

1. Textdatei schreiben

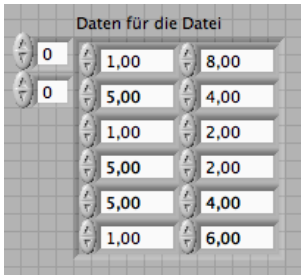


ABB.7.14

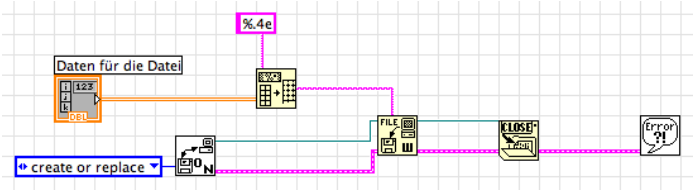


ABB.7.15

- Erzeuge auf dem Frontpanel ein zweidimensionales Feld mit Eintragungen wie in Abb. 7.14 dargestellt.
- Im Blockdiagramm wird mit der Funktion *Array to Spreadsheet String* aus der Unterpalette *String* das 2D-Array in ein Tabelle von Strings umgewandelt, welche mit TABs die Spalten trennt und einen Zeilentrenner einsetzt.

2. Aus einer Textdatei lesen

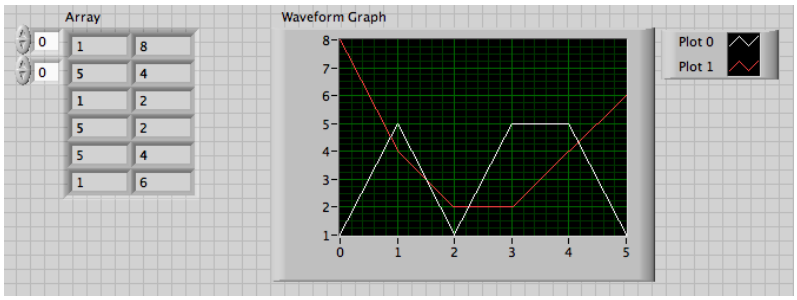


ABB.7.16

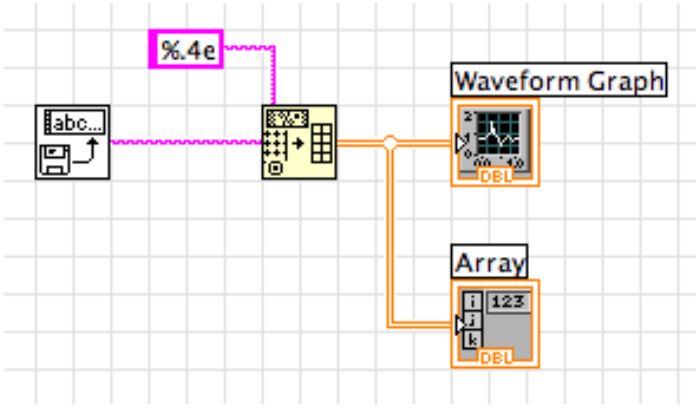


ABB.7.17

- Erzeuge wie in Abb. 7.16 auf dem Frontpanel ein zweidimensionales Feld mit leeren Zellen. Das Feld erhält die Zahlenwerte erst nach der Programmausführung. Die ASCII-Datei hat nebenstehenden Inhalt:

1,0000E+0	8,0000E+0
5,0000E+0	4,0000E+0
1,0000E+0	2,0000E+0
5,0000E+0	2,0000E+0
5,0000E+0	4,0000E+0
1,0000E+0	6,0000E+0

- Im Blockdiagramm wird mit der Funktion *Array to Spreadsheet String* aus der Unterpalette *String* das 2D-Array in ein Tabelle von Strings umgewandelt, welche mit TABs die Spalten trennt und einen Zeilentrenner einsetzt.
- Damit die Plots korrekt Spalte für Spalte übernommen und grafisch im *Waveform Graph* dargestellt werden können, muss in diesem Objekt das Array Feld um 90 ° gegen den Uhrzeigersinn gedreht werden. Dies kann im Frontpanel des *Waveform Graph* mit der Option *Transpose Array* erfolgen.

8. Messdatenerfassung

8.1. Analoge und digitale Mess-Signale

Jede zu erfassende physikalische Größe muss zuerst in ein elektrisches Signal umgewandelt werden, meistens in eine entsprechende Spannung. So gibt es eine Vielzahl von Messumformern für die Temperatur, welche die physikalischen Eigenschaften von Wärme und Material ausnutzen, um eine Temperatur in ein elektrisches Signal umzuwandeln.

Signale können digital sein, d. h. sie haben nur zwei eindeutige Zustände, ein hohes oder ein niedriges Niveau. Daneben können die Signale analog sein. Diese Signale enthalten Informationen in der fortlaufenden Veränderung des Signals in Abhängigkeit von der Zeit.

8.1.1. Analog/Digital-Wandler

Analog/Digital-Wandler (A/D-Wandler) wandeln eine analoge Spannung in ein digitales, dem Rechner verständliches Signal um. So misst ein Digitalvoltmeter (DVM) eine Spannung und zeigt diese in Dezimalzahlen an. A/D-Konverter arbeiten häufig nach dem Verfahren der sukzessiven Approximation. Daran sind die Funktionsgruppen Referenzquelle, Digital-/Analog-Wandler, Näherungsregister und Komparator beteiligt. Zuerst wie im Näherungsregister das höchste Bit gesetzt. Bei einem 4-Bit-Register wird die Binärzahl 1000, also (etwa) die Hälfte der größtmöglichen 4-Bit-Zahl 1111. Diese 4-Bit-Zahl wird mit dem D/A-Wandler in eine analoge Spannung konvertiert, die mit dem Komparator mit der zu messenden Spannung verglichen wird. Ist die zu messende Spannung größer, bleibt das höchste Bit im Näherungsregister gesetzt, andernfall wrid es auf 0 zurückgesetzt. Danach wird das nächsthöhere Bit im Näherungsregister gesetzt, so dass sich die Binärzahl 1100 oder 0100 ergibt. Die zweite Binärzahl wird wider mit der zu messenden Spannung verglichen. Dieser Vorgang wird bis zum niederwertigsten Bit, dem LSB (least significant bit) wiederholt, das bei einem 4-Bit-Register einem 1/16 und bei 12-Bit einem 1/4096 des Wandelbereichs entspricht. Am Ende ergibt sich eine Digitalzahl, die nach der D/A-Wandlung zu der zu messenden Spannung

mit der Genauigkeit LSB proportional ist. Um eine ordentliche Erfassung der Signalfreqeunz zu erhalten, ist nach dem Nyquist-Theorem mindestens das Doppelte der maximalen Signalfrequenz als Abtastrate zu wählen

8.1.2. Digital/Analog-Wandler

Digital/Analog-Wanlder (D/A-Wandler) wandeln eine digitale Größe in Form einer mehrstelligen Binärzahl in eine proportionale analoge Größe wie Spannung oder Stromstärke um. Ein gängiges Umwandlungsverfahren ist wieder die sukzessive Approximation, bei dem ein Widerstandsnetzwerk den eigentlichen D/A-Wandler bildet.

8.1.3. Testen und Konfigurieren der angeschlossenen Hardwarekomponenten

Die Konfiguration der verschiedenen Hardwarekomponenten wird unter *Werkzeuge* mit *Measurement & Automation Explorer* zusammengefasst. Alle Konfigurationseinstellungen des Anwenders stehen in LabVIEW zur Verfügung. Dieses Werkzeug gibt es bislang nur unter Windows.

Übung Testen und Konfigurieren einer Datenerfassungskarte

- Starte den *Measurement & Automation Explorer*.
- Öffne den Abschnitt *Geräte und Schnittstellen*. Unter *NI-DAQmx-Geräte* findest du das USB-6008, wenn du das NI-6008-Device angeschlossen hast.
- Klicke *Selbsttest* an. Es sollte die Meldung „Das Gerät hat den Selbsttest bestanden“ erscheinen. Der Reiter *Eigenschaften* ist beim NI-6008 nicht aktiv.
- Klicke *Testpanels* an. Hier lassen sich verschiedene Funktionen überprüfen und einstellen.
  - *Kanalname* ist z. B. Dev1/ai0
  - *Modus* ist z. B. endlich mit einer Rate von 10000 Hz und 1000 gelesenen Werten.
  - *Min-Messwert* bzw. *Max-Messwert*.
  - *Eingangskonfiguration* im Differential-Modus stehen zwar nur halb so viele Eingangskanäle zur Verfügung wie im Single-Ended Modus. Dafür werden im Differential-Modus Störungen auf den Messleitungen unterdrückt.
- Schließe einen Sinusgenerator an den Kanal Dev1/ai0 und untersuche das Ergebnis im Test-Panel.

Die technischen Daten des 6008-Devices sind unter http://www.ni.com/pdf/manuals/371303f.pdf nachzulesen. Dort sind auch weitere Eigenschaften, z. B. digitale Ein-/Ausgänge oder Counter beschrieben.

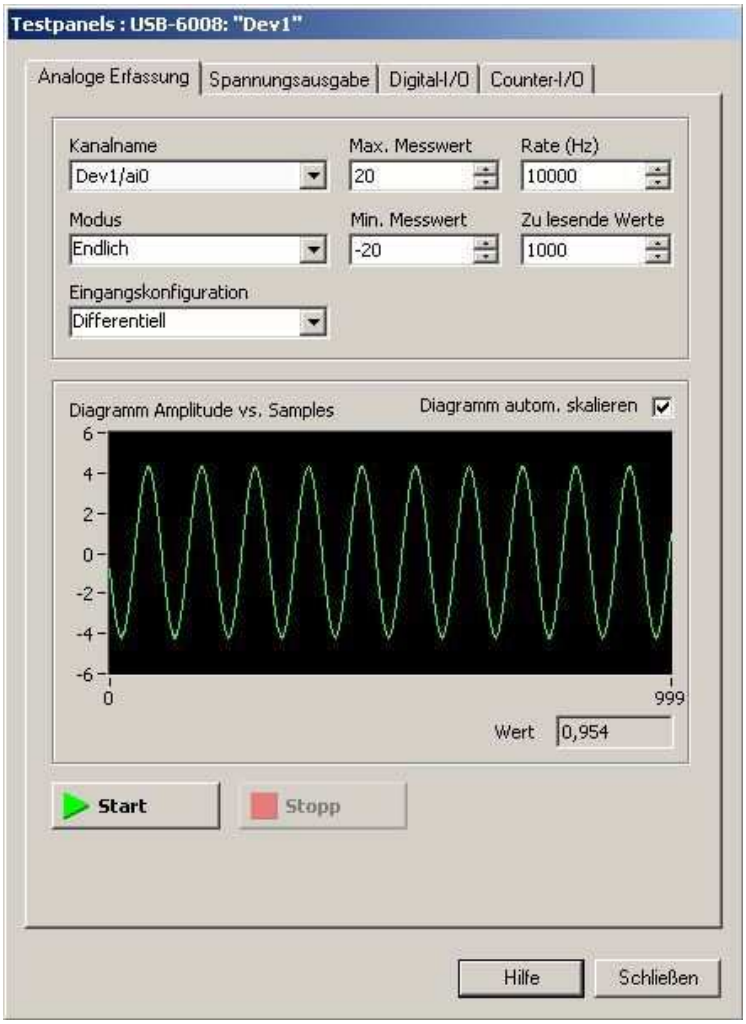


ABB.8.1

8.1.4. Wizard-Technologie für die Datenerfassung

Die Wizard-Technologie (wizard = zauberhaft) erleichtert die Einstieg in die Datenerfassung. In Form von Fragen und Antworten führt der Assistent den Anwender bei der Erstellung von Datenerfassungsapplikationen. Daneben stehen häufig benötigte Anwendungen für den sofortigen Einsatz bereit, wie z. B. Oszilloskope, Multimeter, Funktionsgeneratoren bereit. In LabVIEW wurde ein sogenannter Lösungsassistent integriert.

Übung: Kennenlernen des Lösungsassistenten

Aus dem Startbildschirm heraus kann unter *New* die Option *Data Aqusition with NI-DAQmx.vi* angewählt werden. Es erscheint ein einfaches Frontpanel-VI. Im Blockdiagramm 8.2) sind die detaillierten Hinweise für die Benutzung des Lösungsassistenten gegeben.

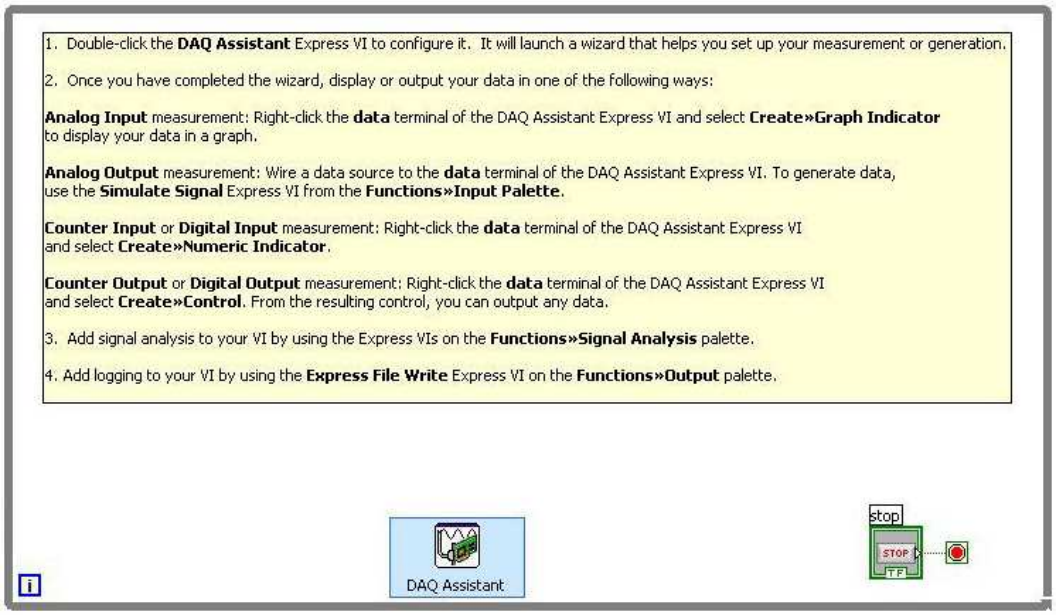


ABB. 8.2

- Nach einem Doppelklick auf das *DAQ-Assistant* Objekt lassen sich die verschiedenen Operationen auswählen, z. B. *Analoge Erfassung*.
- Im folgenden Fenster werden die Operationen, z. B. *Spannung*, *Temperatur*, . . . ausgewählt. Wir wählen bei angeschlossenem Funktionsgenerator *Spannung*.
- Jetzt werden die physikalischen Kanäle für die Task ausgewählt, z. B. *Dev 1 ai0* Danach müssen noch die Anzahl der zu erfassenden Abtastwerte oder kontinuierliche Erfassung eingestellt werden. Auch Timing und Triggerung sind manchmal wichtig.