Course Goals: Automatisierte Messsysteme

- Project Explorer
- Details zu Datentypen (string, numeric, boolean, enum…)
- Datatyp Cluster
- Weitere Strukturen
- Errorhandling
- Dokumentation von LabView-Programmen
- Programm - Architekturen (Sequence, State Machine)

Project-Explorer

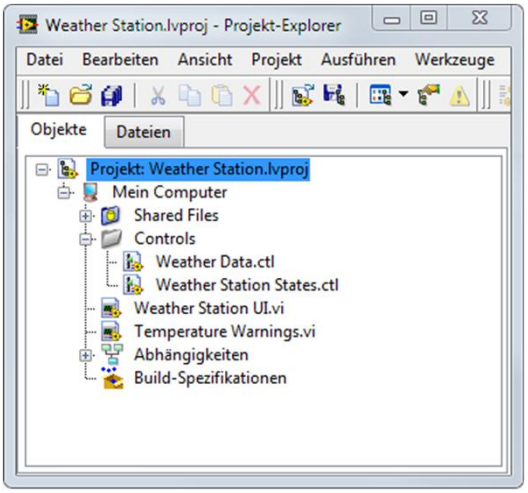
Project-Explorer

Dateitypen

Projektordner

Project-Explorer

- Dient zum Suchen, Öffnen und Organisieren von Projektdateien
- Verhindert, erkennt und behebt falsche Verknüpfungen
- Überträgt Dateien auf Zielsysteme
- Verwaltet Code für Build-Optionen
  - Ausführbare Dateien, Installer und ZIP-Dateien
- Arbeitet mit Versionsverwaltungs-providern



LabVIEW Project

Use projects to group together LabVIEW files and non-LabVIEW files, create build specifications, and deploy or download files to targets. A target is a device or machine on which a VI runs. When you save a project, LabVIEW creates a project file (.lvproj), which includes configuration information, build information, deployment information, references to files in the project, and so on.

You must use a project to build stand-alone applications and shared libraries. You also must use a project to work with an RT, FPGA, or PDA target. Refer to the specific module documentation for more information about using projects with the LabVIEW Real-Time, FPGA, and PDA Modules.

Project-style LabVIEW Plug and Play instrument drivers use the project and project library features. You can use project-style drivers in the same way as previous LabVIEW Plug and Play drivers.

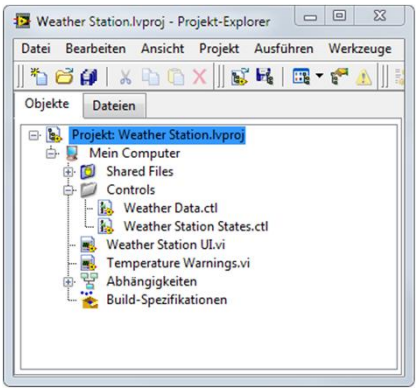
Project Explorer Window

Use the Project Explorer window to create and edit projects. Select **File»New Project** to display the Project Explorer window. You also can select **Project»New Project** or select **File»New** and then select **Empty Project** in the New dialog box to display the Project Explorer window.

LabVIEW-Dateien

Gängige LabVIEW-Dateitypen:

- LabVIEW-Projekt: *.lvproj
- Virtuelles Instrument (VI): *.vi
- Benutzerdefiniertes Element: *.ctl



LabVIEW uses many different types of files. In this class you will learn about three different LabVIEW files – LabVIEW projects, VIs, and custom controls.

Historically, LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. Today LabVIEW VIs can be extremely powerful and sophisticated programs with elegant graphical user interfaces.

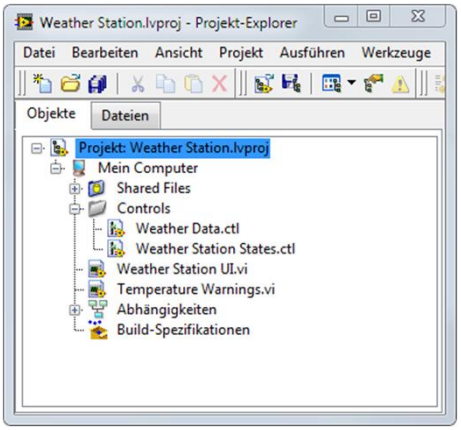
Later in this course you learn how custom controls can improve maintainability of your LabVIEW application.

LabVIEW projects can also include non-LabVIEW file types. For example, you can include documentation files.

Note: If students need to work with legacy code, you might briefly cover .llb files.

Hinzufügen von Ordnern zu einem Projekt

- Virtueller Ordner
  - Enthält Projektobjekte; keine direkte Entsprechung für einen Ordner im Dateisystem
- Ordner mit Autofüllfunktion
  - Fügt dem Projekt ein Verzeichnis hinzu
  - Passt sich fortlaufend an Änderungen im Projekt und auf dem Datenträger an



Adding Folders to a Project

Use the Project Explorer window to add folders to create an organizational structure for items in a LabVIEW project.

Adding auto-populated folders adds a directory on disk to the project. LabVIEW continuously monitors and updates the folder according to changes made in the project and on disk. A blue folder icon with a yellow cylinder identifies this type of folder. To disconnect an auto-populated folder from disk, right-click the auto-populated folder on the **Items** page and select **Stop Auto-populating** from the shortcut menu. LabVIEW disconnects the folder from the corresponding folder on disk. This option is available only to top-level folders and applies recursively to subfolders of auto-populated folders.

A virtual folder is a folder in the project that organizes project items and does not represent files on disk. A silver folder icon identifies this type of folder. You can convert a virtual folder to an auto-populated folder. Right-click the virtual folder and select **Convert to Auto-populating Folder** to display a file dialog box. Select a folder on disk to auto-populate with. An auto-populated folder appears in the project. LabVIEW automatically renames the virtual folder to match the disk folder and adds all contents of the disk folder to the project. If items in the directory already exist in the project, the items move within the auto-populated folder. Items in the virtual folder that do not exist in the directory on disk move to the target.

LabVIEW-Datentypen

Kontextmenü und Dialogfeld "Eigenschaften"

Numerische Typen

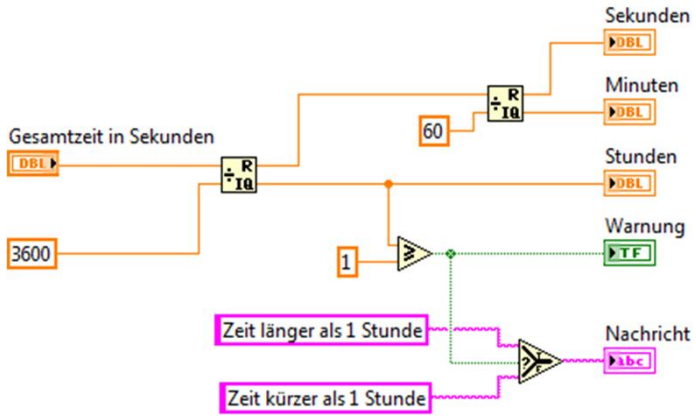
Boolesche Typen

String-Typen

Enums und andere Typen

LabVIEW-Datentypen

Blockdiagrammsymbole zeigen Angaben zum Datentyp an



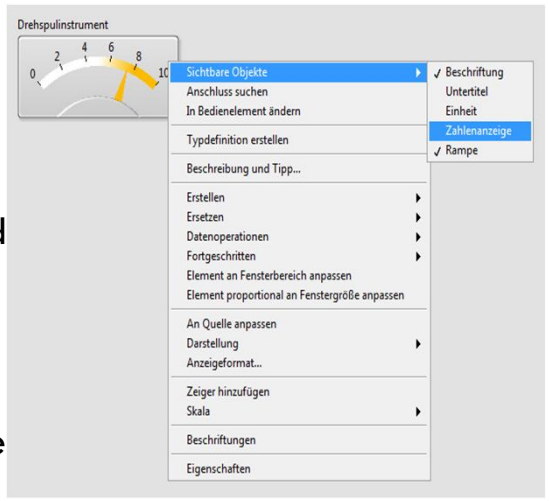
Terminal colors, text, arrow direction, and border thickness all provide visual information about the terminal.

For example, Orange represents floating point numbers. DBL indicates a double-precision floating point number.

Terminals with thick borders with arrows on the right are control terminals. Terminals with thin borders with arrows on the left are indicator terminals.

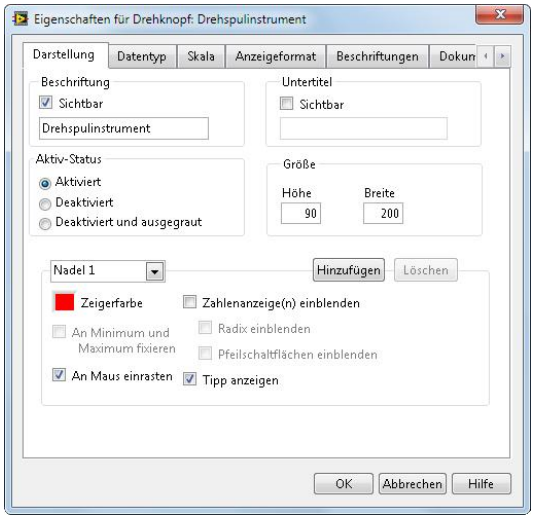
Kontextmenüs

- Zu allen LabVIEW-Objekten gibt es Kontextmenüs
- Mit Kontextmenüpunkten können Sie Aussehen und Funktionsweise von Objekten ändern
- Zum Öffnen des Kontextmenüs klicken Sie das betreffende Objekt mit der rechten Maustaste an



Dialogfeld "Eigenschaften"

- Alle LabVIEW-Objekte haben Eigenschaften
- Werden durch einen Rechtsklick auf das Objekt und Auswahl von **Eigenschaften** angezeigt
- Eigenschaftsoptionen ähneln Kontextmenüpunkten
- Durch Markieren mehrerer Objekte können Sie diese gleichzeitig konfigurieren



Instructor Demo:

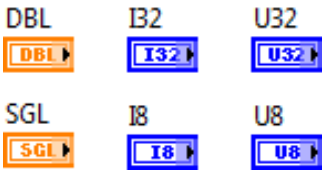
Demonstrate both shortcut menus and the **Properties** dialog box.

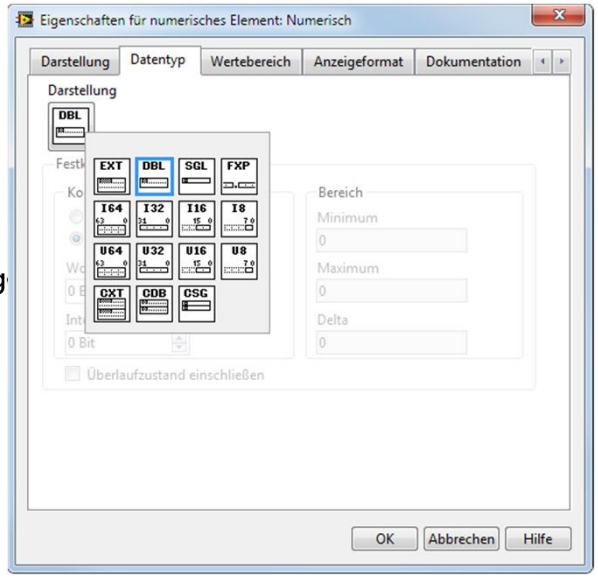
Show how to multi-select and configure multiple objects through the **Properties** dialog box. For example, show and hide labels for multiple controls.

Werte

Unterschiedliche Datentypdarstellungen:

- Fließkomma
- vorzeichenlose Integer
- vorzeichenbehaftete Integ





Integers represent whole numbers.

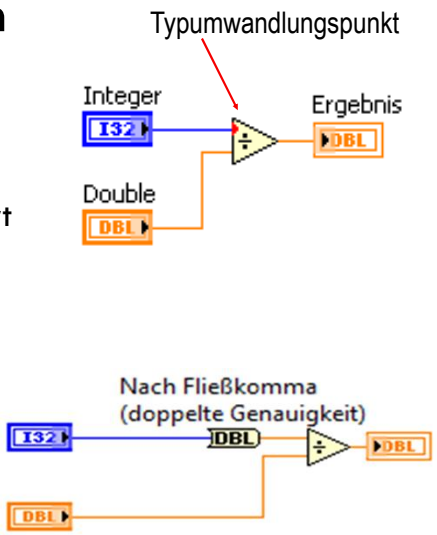
Unsigned integers are non-negative values.

Signed integers can be positive, negative, or zero.

Learn about implementing these data types, such as representation, Boolean action, and string display type. They also learn a couple new data types: Enum and dynamic.

Umwandeln von Zahlen

- Typumwandlungspunkte zeigen an, dass die Darstellung des Eingangswerts automatisch geändert wurde
  - Wird angezeigt, wenn ein Knoten eine Eingabe anderer Darstellung erwartet
- Es wird die mit mehr Bits erzeugte Darstellung bevorzugt
- Typumwandlung ist durch programmatische Umwandlung in den richtigen Datentyp vermeidbar



If you use two numbers of the same type with different bit widths, LabVIEW coerces the smaller to the larger of the two bit widths.

Examples:

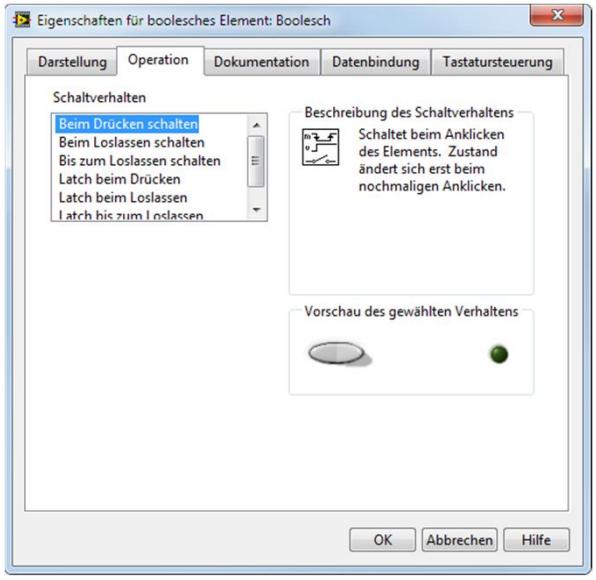
If you use a signed integer with an unsigned integer, LabVIEW coerces the unsigned integer to the signed integer.

If you use an integer with a floating point, LabVIEW coerces the integer to the floating point.

If you use a floating point with a complex integer, LabVIEW coerces the floating point to the complex integer.

Boolesche Werte

- Die Funktionsweise boolescher Elemente richtet sich nach dem eingestellten Schaltverhalten
- Boolesche Elemente haben nur die Werte TRUE und FALSE



Use Boolean controls and indicators to enter and display Boolean (TRUE/FALSE) values. For example, if you are monitoring the temperature of an experiment, you can place a Boolean warning light (LED) on the front panel to indicate when the temperature exceeds a certain level.

Instructor: Point out the difference between the label name and Boolean text. Show how the label matches the block diagram terminal label. Boolean text is cosmetic and appears only on the front panel.

Schaltverhalten boolescher Elemente

Point out that in everyday life we interface with many Boolean switches and buttons. These switches and buttons have different mechanical behaviors.

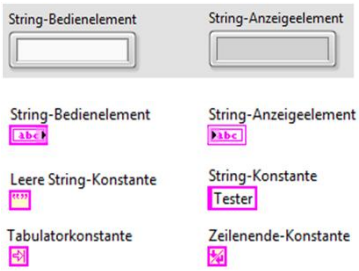
Light switches change state when the switch is flipped. The state stays the same until you flip the switch again.

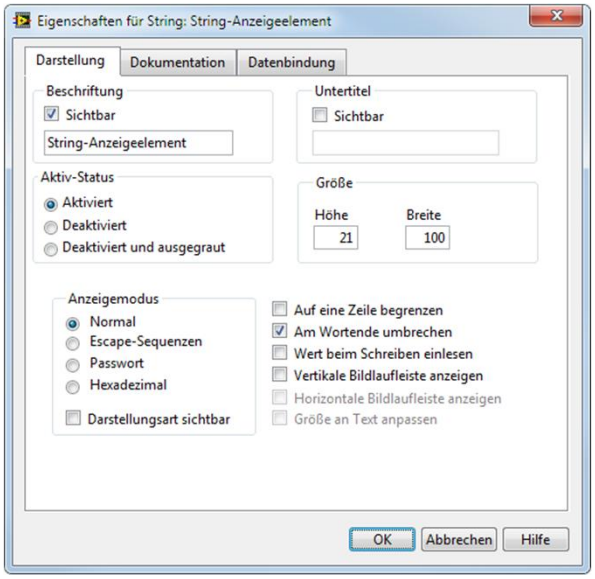
Buzzers and door bells change state on a button press. They change back when the button is released.

Mouse clicks and keyboard presses have a latch behavior. They change state on a button release. Furthermore, the key press or mouse click only takes effect when read by the system. When your computer system is sluggish, we sometimes see a delay in processing a mouse click or key press.

Strings

- Folge von ASCII-Zeichen
- Auf vielfältige Weise darstellbar:
  - Escape-Sequenz
  - Passwort
  - Hexadezimal





Uses for strings:

•Creating simple text messages.

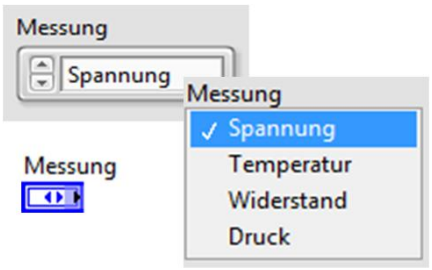
•Controlling instruments by sending text commands to the instrument and returning data values in the form of either ASCII or binary strings which you then convert to numeric values.

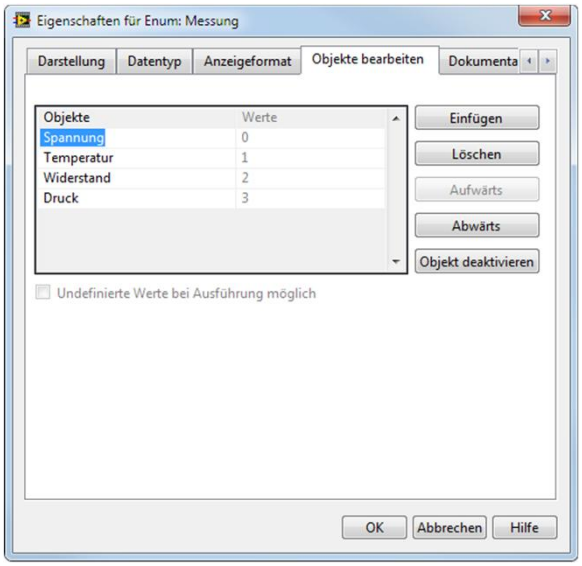
•Storing numeric data to disk. To store numeric data in an ASCII file, you must first convert numeric data to strings before writing the data to a disk file.

•Instructing or prompting the user with dialog boxes.

Enums

- Gibt Benutzern eine Liste mit Auswahlmöglichkeiten
- Jeder Eintrag besteht aus einem Wertepaar
  - String
  - 16-Bit-Integer





Enums are useful because they make strings equivalent to numbers, and it is easier to manipulate numbers than strings on the block diagram.

Point out that the data type of the enum terminal is blue, showing that the enum is passing an integer value.

Andere Datentypen

Eine vollständige Liste von Bedien- und Anzeigeelementtypen finden Sie in der *LabVIEW-Hilfe*

- Dynamisch 
  - Enthält die von einem Express-VI erzeugten oder erfassten Daten
- Pfad 
  - Gibt den Speicherort einer Datei oder eines Verzeichnisses in der plattformspezifischen Standardsyntax an
- Signalverlauf 
  - Enthält Daten, Startzeit und Schrittweite eines Signalverlaufs

Although it is easy to convert a Path to a String, it is best to use a Path data type when working with a location of a file or directory. Using the Path data type, LabVIEW can handle the folder specifiers (for example, a backslash on Windows).

Cluster

Verwendungsgründe

Vergleich mit Arrays

Erstellen von Cluster-Elementen und -Konstanten

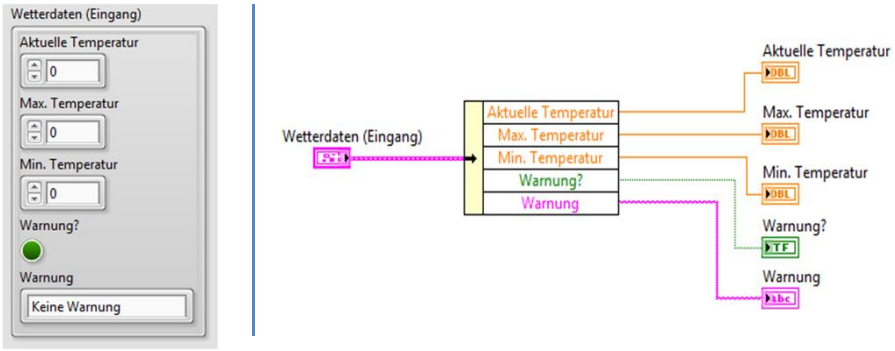
Sortieren von Elementen und Ändern der Cluster-Größe

Aufschlüsseln und Ändern von Clustern

Visualisierung von Daten

Cluster

- Gruppiert Daten unterschiedlichen Typs
- Entspricht in befehlsorientierten Programmiersprachen einem Datensatz oder einer Struktur

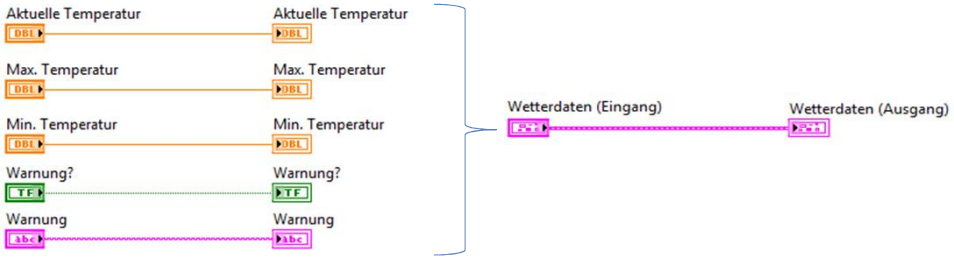


Additional items to mention:

- Bundling several data elements into clusters eliminates wire clutter on the block diagram and reduces inputs and outputs for subVIs.
- Most clusters on the block diagram have a pink wire and data type terminal.
- Clusters of numeric values, sometimes referred to as points, have a brown wire and data type terminal.
- You can wire brown numeric clusters to Numeric functions to perform the same operation simultaneously on all elements of the cluster.

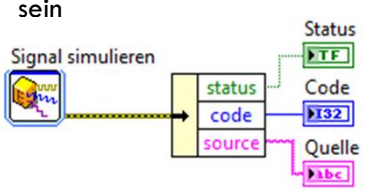
Verwendungsgründe für Cluster

- Strukturieren Daten
  - Logische Gruppierung zusammenhängender Datenwerte
  - Verständlicheres Blockdiagramm (weniger Verbindungen nötig)
- Erfordern weniger Anschlussfeldanschlüsse

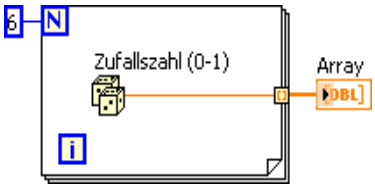


Cluster verglichen mit Arrays

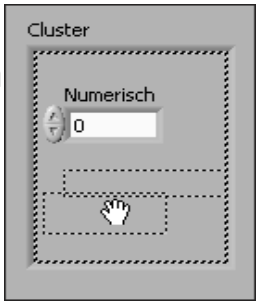
- Cluster haben eine feste Größe
- Cluster können unterschiedliche Datentypen enthalten
- Cluster können Bedienelemente, Anzeigeelemente oder Konstanten sein
  - Dafür müssen alle Elemente im Cluster Bedienelemente, Anzeigeelemente oder Konstanten sein



- Arrays sind in ihrer Größe flexibel
- Sie können nur einen Datentyp enthalten
- Arrays können Bedienelemente, Anzeigeelemente oder Konstanten sein



Erstellen von Cluster-Elementen



Neuer Cluster:

1. Cluster aus der **Elemente**-Palette des Frontpanels auswählen
2. Datenobjekt in Container einfügen
3. Bei Bedarf weitere Datenobjekte in Container einfügen

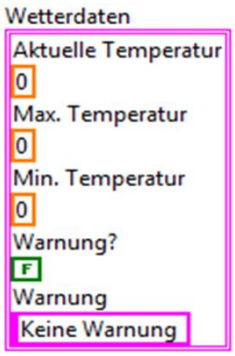
Von einem Blockdiagrammanschluss oder Verbindungselement aus:

1. Rechtsklick auf das Objekt und Auswahl von **Erstellen»Bedienelement** oder **Erstellen»Anzeigeelement**

Erstellen von Cluster-Konstanten

Neuer Cluster:

1. Cluster-Konstante aus der **Funktionen**-Palette im Blockdiagramm auswählen
2. Konstante in den Cluster-Container einfügen
3. Bei Bedarf weitere Datenobjekte hinzufügen

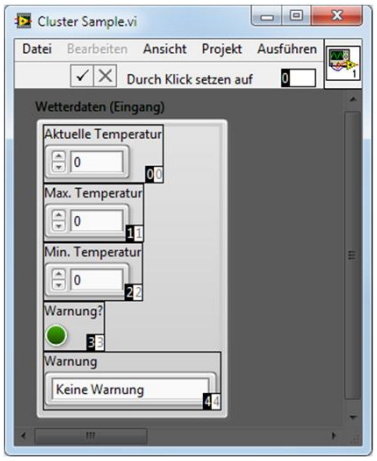


Von einem Blockdiagrammanschluss oder Verbindungselement aus:

1. Rechtsklick und Auswahl von **Erstellen»Konstante**

Cluster-Elementindex

1. Cluster-Elemente haben eine logische Reihenfolge, die nicht direkt ihrer Position im Cluster entspricht.
2. Zum Ermitteln oder Ändern der Clusterreihenfolge: Rechtsklick auf den Cluster-Rand und Auswahl von **Elemente im Cluster neu ordnen**



NATIONAL INSTRUMENTS | ni.com/training

Notes:

The first object you place in the cluster is element 0, the second is element 1, and so on.

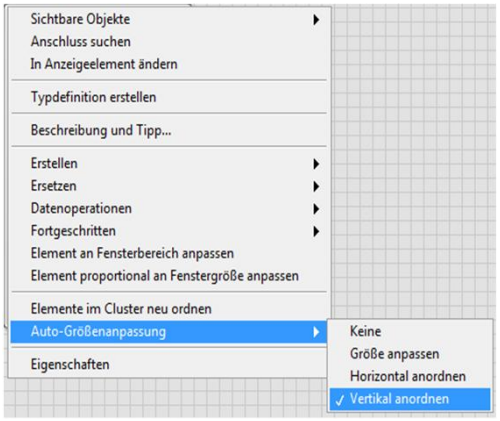
If you delete an element, the order adjusts automatically.

The cluster order determines the order in which the elements appear as terminals on the Bundle and Unbundle functions on the block diagram.

Use **AutoSizing»Arrange Horizontally** or **AutoSizing»Arrange Vertically** from the shortcut menu to arrange the elements in the cluster horizontally or vertically in order.

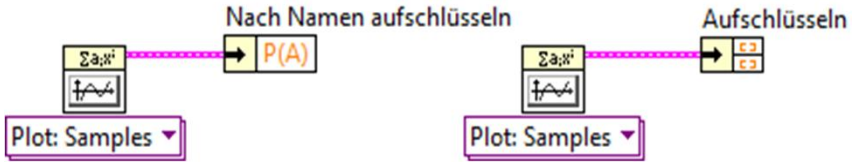
Automatische Größenanpassung von Clustern

- Automatische Größenanpassung verbessert die Anordnung der Cluster-Elemente
- Folgende Praktiken werden empfohlen:
  - Cluster-Elemente vertikal anordnen
  - Elemente möglichst kompakt anordnen
  - Elemente werden ihrer bevorzugten Reihenfolge nach angeordnet



Aufschlüsseln von Clustern

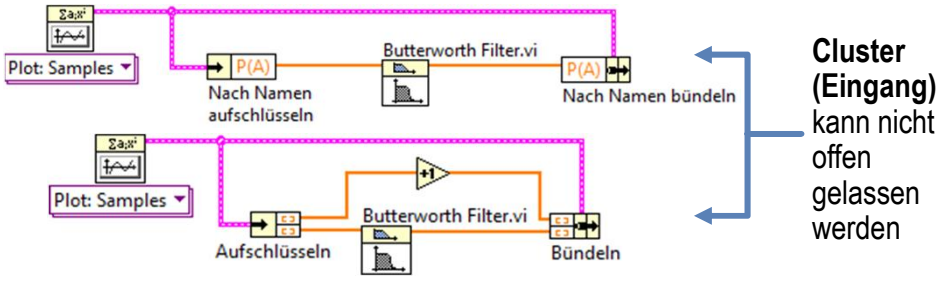
- Sollte nach Möglichkeit mit der Funktion "Nach Namen aufschlüsseln" durchgeführt werden
- Die Funktion "Aufschlüsseln" ist zu verwenden, wenn unbenannte Cluster-Elemente vorliegen



In this slide, the Polynomial Plot.vi outputs an XY Graph as a cluster of two arrays (X and Y). The Y array is labeled as **P(A)** so you can access it using the Unbundle By Name function. However, the X array is unlabeled so you cannot access it using the Unbundle By Name function. To access data in the X array, use the Unbundle function.

Bearbeiten von Clustern

- Sollte nach Möglichkeit mit der Funktion "Nach Namen bündeln" durchgeführt werden
- Die Funktion "Bündeln" ist zu verwenden, wenn unbenannte Cluster-Elemente vorliegen

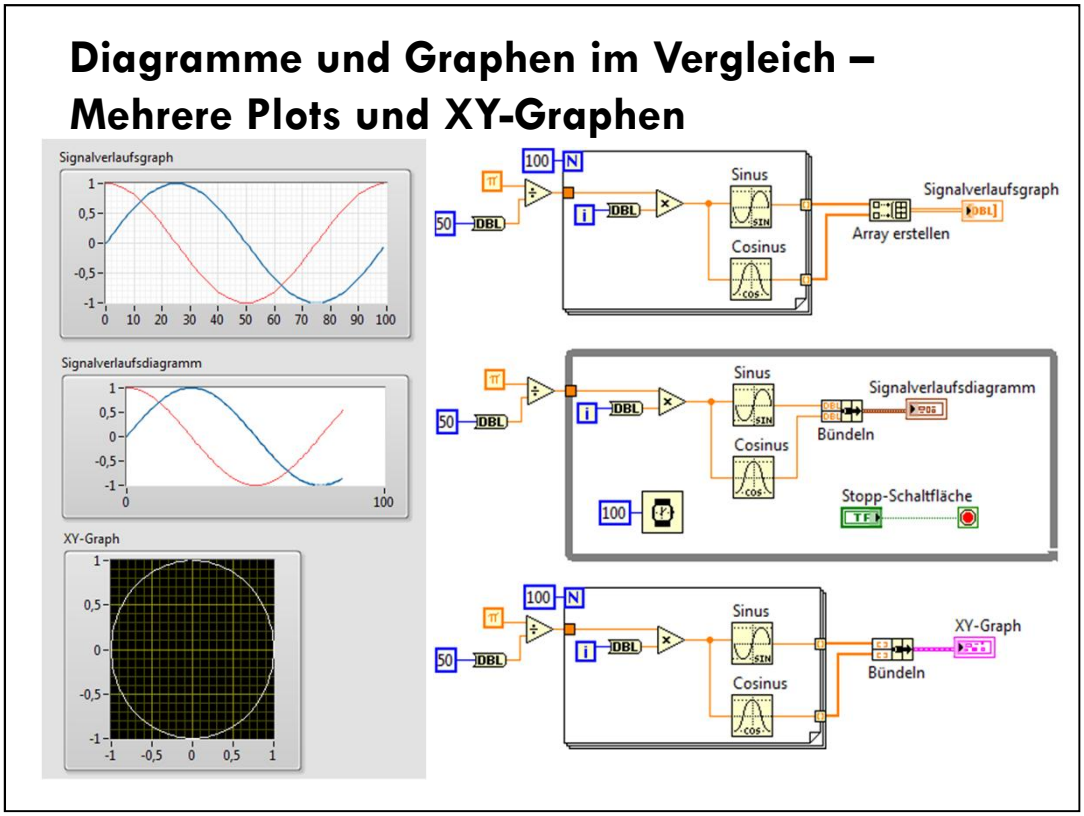


In this slide, both examples modify the **Y** array, (P(A), of the Polynomial Plot.vi with a Butterworth filter. In the bottom example, you also increment the X array of the Polynomial Plot.vi output by one.

Erstellen von Clustern im Blockdiagramm

- Programmatisch mit der Funktion "Bündeln"
- Wenn die gebündelten Elemente beschriftet sind, können Sie mit der Funktion "Nach Namen aufschlüsseln" auf diese zugreifen; anderenfalls mit der Funktion "Aufschlüsseln"

This technique is typically used to create multi-plot charts which is discussed in more detail later.

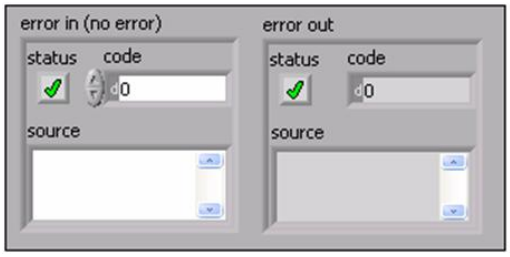


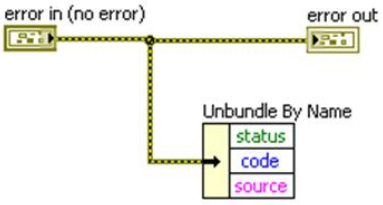
The Bundle function is often used to create multi-chart plot charts and XY plots.

The Build Array function is used to create multi-plot waveform graphs.

Fehler-Cluster

- Leiten Fehlerangaben weiter
- Enthalten folgende Elemente:
  - Status: boolescher Wert; bei einem Fehler TRUE
  - Code: vorzeichenbehafteter 32-Bit-Integer zur Fehleridentifizierung
  - Quelle: String zur Angabe des Verursachers





Undefinierte, unerwartete Daten, Standardwerte

Undefinierte oder unerwartete Daten

Prüfen Sie bei Rechenoperationen, ob an unerwarteter Stelle "Inf" oder "NaN" ausgegeben wird

- ∞ (Inf)
  - Unendlich
  - Wird bei Division durch Null ausgegeben
- NaN
  - Keine Zahl
  - Wird bei ungültigen Rechenoperationen ausgegeben, z. B. beim Berechnen der Quadratwurzel einer negativen Zahl

LabVIEW can process complex numbers using the functions on the **Complex** palette.

Standard verwenden, wenn nicht verbunden

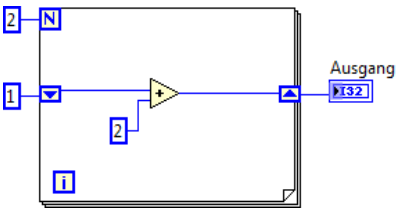
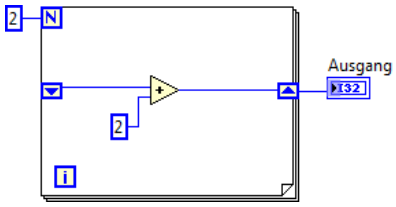
Standardwerte sind je nach Datentyp unterschiedlich:

Datentyp	**Standardwert**
Numerisch	0
Boolesch	FALSE
String	Leer

Nicht initialisierte Schieberegister verwenden Standardwerte für die erste Ausführung

Initialisierung von Schieberegistern

Einmal ausführen → VI läuft durch → Neu starten

Blockdiagramm	1. Ausführung	2. Ausführung
Initialisiertes Schiebe-register 	Ausgangswert = 5	Ausgangswert = 5
Nicht initialisiertes Schiebe-register 	Ausgangswert = 4	Ausgangswert = 8

Weitere Strukturen

Formelknoten

Lokale Variable

Globale VariableFehler-Cluster

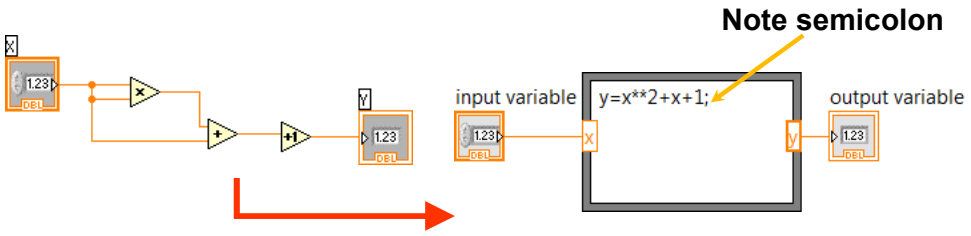
Shared Variable

Diagram Disable

Navigationsfenster

Formelknoten

Functions >> Programming >> Structures
Ermöglicht Eingabe von komplizierten Gleichungen
Input-Variablen werden im linken Rahmen angelegt
Output-Variablen werden im rechten Rahmen angelegt
Variablennamen sind case sensitive
Jede Formel muss mit einem Strichpunkt abgeschlossen werden
Context Help Window zeigt verfügbare Funktionen + Syntax



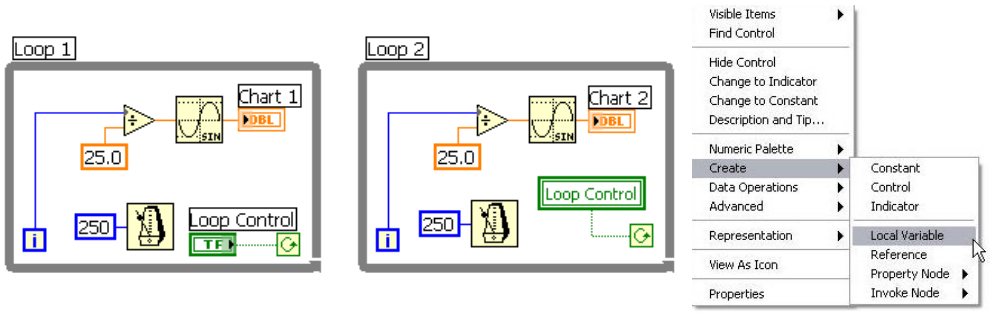
Sometimes it is preferable to program mathematical expressions with text-based function calls, rather than icons (which can take up a lot of room on the diagram).
Formula Node: allows you to implement complicated equations using text-based instructions.

- •Located in the **Structures** subpalette.
- •Resizable box for entering algebraic formulas directly into block diagrams.
- •To add variables, right-click and choose **Add Input** or **Add Output**. Name variables as they are used in formula. (Names are case sensitive.)
- •Statements must be terminated with a semicolon.
- •When using several formulas in a single formula node, every assigned variable (those appearing on the left hand side of each formula) must have an output terminal on the formula node. These output terminals do not need to be wired, however.

Compare the examples on the slide.

Lokale Variable (Details SS)

- Erlaubt Kommunikation zwischen parallelen Schleifen
- Ein control or indicator kann von mehreren Stellen gelesen bzw. geschrieben werden
  - Lokale Variablen entsprechen nicht dem Datenflussparadigma und sollten sparsam verwendet warden.



Local variables are located in the **Structures** subpalette of the **Functions** palette.

When you place a local variable on the diagram, it contains by default the name (owned label) of the first object you placed on the front panel.

You use a local variable by first selecting the object you want to access. You can either click on the local variable with the Operating tool and select the object (by owned label) you want to access, or pop up on the local variable and choose the object from the **Select Item** menu.

Next, you must decide to either read or write to the object. Right click on the local variable and choose **Change To Read** or **Change to Write**.

Global Variable

- Ermöglicht Datenaustausch zwischen Vis
- Globale Variable hat nur Frontpanel

Shared Variables

- Shared Variables are used to send data between VIs.
- Variable Types:
  - Single Process: share the data among VIs on the local computer.
  - Network-published: communicate between VIs, remote computers, and hardware through the Shared Variable Engine.

- Shared Variable must exist within a project library.
- Shared Variable must be deployed to be available to other projects and remote computers.

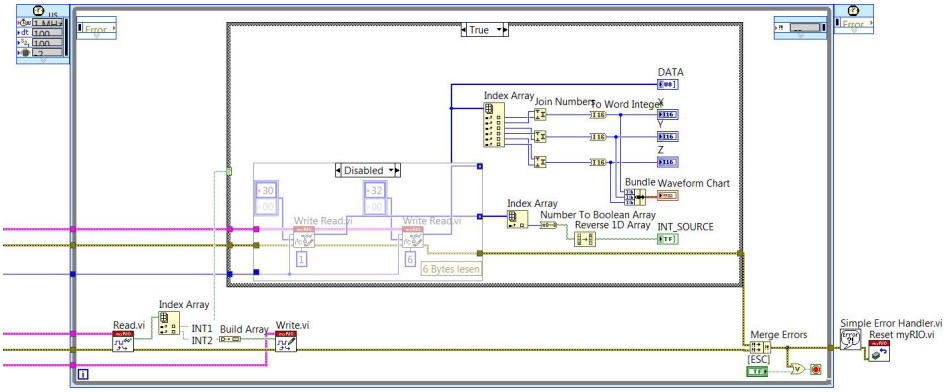
Shared variables are used to share data among VIs or between locations in an application that cannot be connected with wires. There are two variable types:

- Single Process: create shared variables that you want to read and write on a single computer.
- Network-published: create shared variables that you want to read and write on remote computers and targets on the same network.

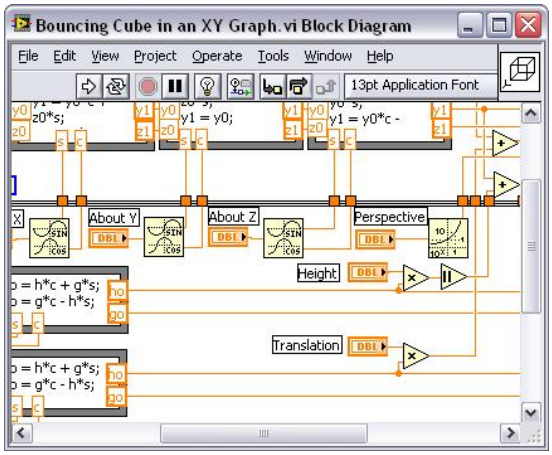
These shared variables must be inside project libraries. If you create a shared variable from a target or folder that is not inside a project library, LabVIEW creates a new project library and places the shared variable inside. You must deploy a shared variable for the variable to be available to other projects and remote computers. You can do by running the VI in which the shared variable resides. You also can right-click the owning project library of the shared variable and select **Deploy** from the shortcut menu.

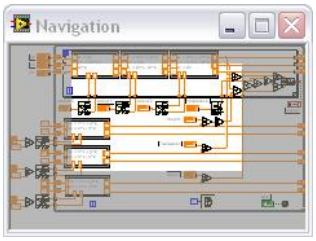
Disable Structure

- Erlaubt das Disablen von Bereiche des Blockdiagramms
- Nur der Enable-Rahmen wird ausgeführt.
- Hilfreich beim Debuggen



LabVIEW Navigation Window





- Zeigt den aktuellen Ausschnitt im Vergleich zum gesamten Front Panel oder Blockdiagramm
- Gut für große unübersichtliche Programme

Select **View»Show Navigation Window** to display this window.

Use the window to navigate large front panels or block diagrams. Click an area of the image in the **Navigation Window** to display that area in the front panel or block diagram window. You also can click and drag the image in the **Navigation Window** to scroll through the front panel or block diagram.

Fehlerbehandlung

Automatisch und manuell möglich

Funktion "Fehler zusammenfassen"

Fehler-Cluster

Warnungen

Fehlerbehandlung

Fehlerbehandlung: Vorhersehen, Erkennen und Beheben von Warnungen und Fehlern

- Nicht jedes potentielle Problem mit einem VI ist vorhersehbar
- Ohne Fehlerprüfung wissen Sie nur, dass ein VI nicht ordnungsgemäß funktioniert
- Durch Fehlerbehandlung erfahren Sie, warum und wo Fehler auftreten
  - Automatische Fehlerbehandlung
  - Manuelle Fehlerbehandlung

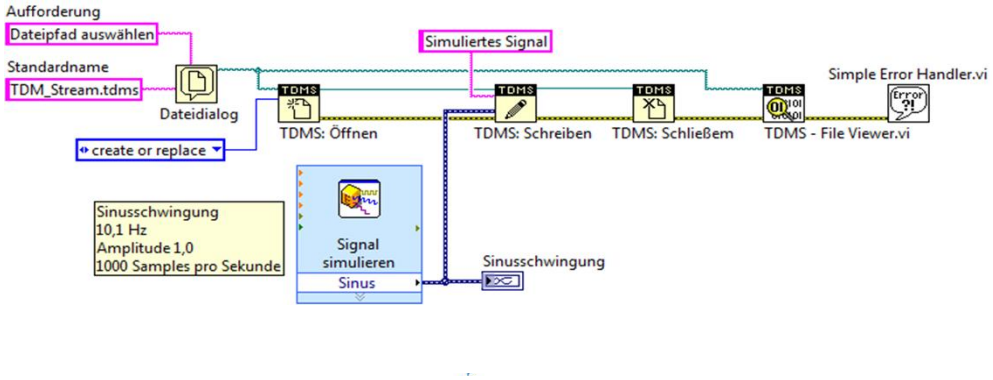
Automatische Fehlerbehandlung

- Standardmäßig reagiert LabVIEW automatisch auf einen bekannten Fehler, indem es:
  - die Ausführung unterbricht
  - den Verursacher des Fehlers hervorhebt
  - die **Fehlerliste** anzeigt
- Automatische Fehlerbehandlung:
  - ist für schnelle Prototypen akzeptabel
  - wird für die professionelle Entwicklung nicht empfohlen
  - ist für in LabVIEW erzeugte EXE-Dateien nicht verfügbar

Select **File»VI Properties** and select **Execution** from the **Category** pull-down menu to enable or disable automatic error handling for a specific VI.

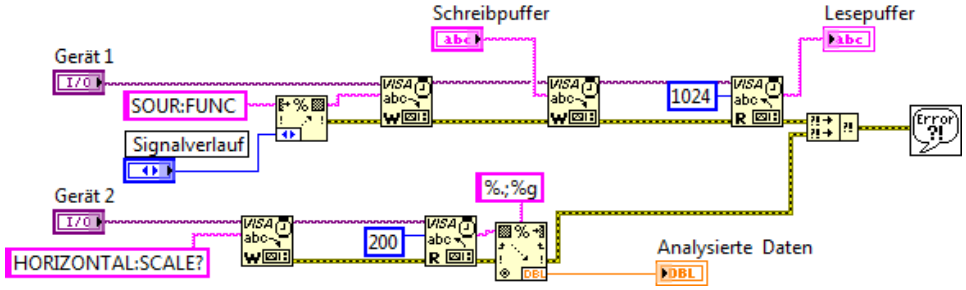
Manuelle Fehlerbehandlung

- Sie bestimmen die Anzeige der Dialogfelder
- Angaben zu Fehlern werden durch Verbinden des Anschlusses **Fehler (Ausgang)** mit dem **Fehler (Eingang)** des nachfolgend ausgeführten Knotens weitergeleitet
- Die Fehlerkette wird durch einen Aufruf des VIs "Einfacher Fehlerbehandler" beendet



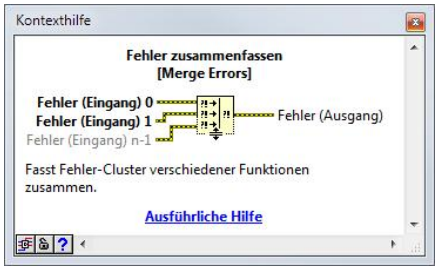
Fehler zusammenfassen

- Leitet Fehlern über Verbindungen weiter
- Fasst Fehler unterschiedlicher Quellen zusammen



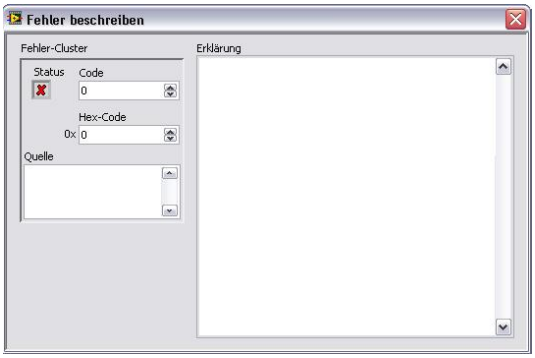
Funktion "Fehler zusammenfassen"

- Gibt den ersten gefundenen Fehler bzw. die erste Warnung aus
- Nicht zum Verketten von Fehlerangaben geeignet



Fehler-Cluster

- Die Fehlerein- oder -ausgänge von SubVIs werden durch Fehler-Cluster mit Angaben zu Fehlern versorgt
- Die Cluster **Fehler (Eingang)** und **Fehler (Ausgang)** enthalten folgende Angaben:
  - Status
  - Code
  - Quelle



Go to **Help»Explain Error** to access the **Explain Error** dialog box.

The **Status** component of the error cluster is a Boolean value that reports TRUE if an error occurred. Most VIs, functions, and structures that accept Boolean data also recognize this parameter. For example, you can wire an error cluster to the Boolean inputs of the Stop, Quit LabVIEW, or Select functions. If an error occurs, the error cluster passes a TRUE value to the function.

Code is a 32-bit signed integer that identifies the error numerically. A non-zero error code coupled with a status of FALSE signals a warning rather than a fatal error.

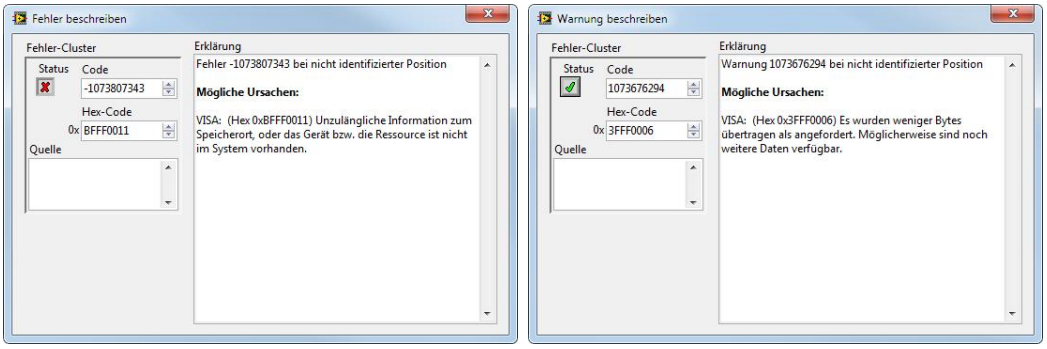
Source is a string that identifies where the error occurred

Fehler und Warnungen

Status = TRUE

Status = FALSE

Code ist ungleich Null



Fehler **Warnung**

Although most error codes are negative and warning codes are positive, this is not universally true.

Most products and VI groups produce only errors. Some products and VI groups can produce warnings. VISA is an example of a product group that can produce warnings.

Dokumentieren von Programmcode

VI-Beschreibungen

Hinweisstreifen

Beschriftungen

- Frei
- Objektgebunden

Dokumentieren von Programmcode

VI	Frontpanel	Blockdiagramm
• Name • Beschreibung	• Namensbeschriftungen • Hinweisstreifen • Beschreibung • Freie Beschriftungen	• Namensbeschriftungen • Freie Beschriftungen • Objektbeschriftungen • SubVI-Beschriftungen

Giving VIs, controls, and indicators logical and descriptive names adds readability and usability to front panels.

VI and object descriptions appear in the **Context Help** window when you move the cursor over the object. Create descriptions that describe the purpose of the VI or object. Include user instructions for the VI or object.

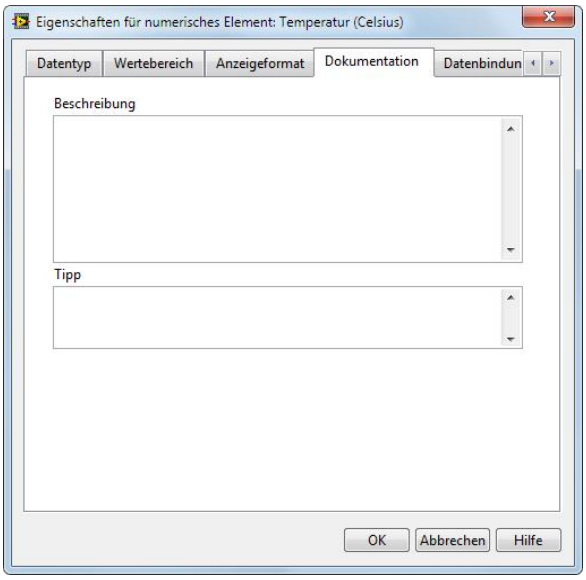
LabVIEW includes two kinds of labels—owned labels and free labels. Owned labels belong to and move with a particular object and annotate only that object. You can move an owned label independently, but when you move the object that owns the label, the label moves with the object. You can hide owned labels, but you cannot copy or delete them independently of their owners.

Documenting VI:

By opening a VI's properties a developer can add documentation to his or her VI. The documentation placed in the Description field of the VI Documentation window shows up in Context Help, and prints with the VI.

Erstellen von Beschreibungen und Tipps

Objekte werden im Dialogfeld **Eigenschaften** dokumentiert



Instructor:

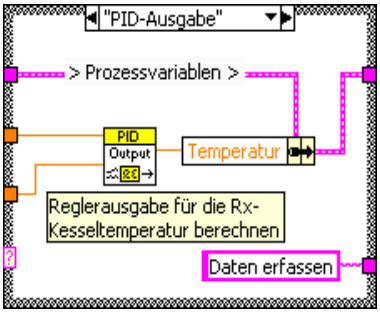
Demonstrate tips strips, descriptions and VI Properties, and how they appear in a VI. For example, VI descriptions show up in Context Help when you move the cursor over the object, and tip strips are brief descriptions that appear when you move the cursor over an object while a VI runs.

Create, edit, and view VI descriptions by selecting **File»VI Properties** and selecting **Documentation** from the **Category** pull-down menu. Create, edit, and view object descriptions by right-clicking the object and selecting **Description and Tip** from the shortcut menu. If you do not enter a tip in the **Description and Tip** dialog box, no tip strip appears.

Dokumentieren von Blockdiagrammcode

Freie Beschriftungen:

- beschreiben Algorithmen
- haben einen hellgelben Hintergrund
- werden durch einen Doppelklick an eine freie Stelle erzeugt



Objektbeschriftungen:

- erklären die Daten von Verbindungen und Objekten
- sind nicht vom Objekt trennbar
- haben einen transparenten Hintergrund
- werden über das Kontextmenü (**Sichtbare Objekte»Beschriftung**) erstellt

Use the following guidelines when commenting your VIs:

• Use comments to document algorithms and add reference information.

• Label structures to specify the main functionality.

• Label long wires to identify their use/contents. (To label a wire, right-click on the wire and select **Visible Items»Label**.)

• Label constants to specify the nature of the constant.

• It is not always necessary to show labels on functions and subVIs if they make the block diagram cluttered. A developer can find information about a function or subVI by using the **Context Help** window.

Zeitsteuerung eines VIs

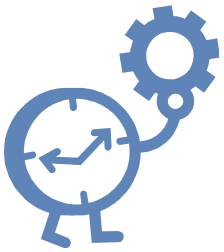
Gründe für Timing

Wartefunktionen und -Express-VIs

Zeitsteuerung eines VIs

Warum muss der zeitliche Ablauf eines VIs gesteuert werden?

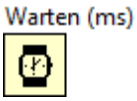
- Um die Frequenz zu bestimmen, mit der eine Schleife abläuft
- Um dem Prozessor Zeit für andere Tasks zu geben, z. B. zum Verarbeiten von Benutzereingaben

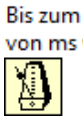


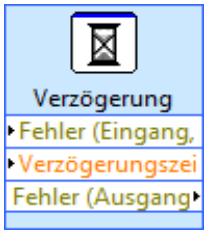
Wartefunktionen

Eine Wartefunktion in einer Schleife

- Ermöglicht ein zeitweiliges Pausieren des VIs
- In dieser Zeit kann der Prozessor andere Aufgaben erledigen
- Die Wartefunktion arbeitet mit der Betriebssystemuhr







Case-Strukturen

Teile einer Case-Struktur (Wiederholung Basics)

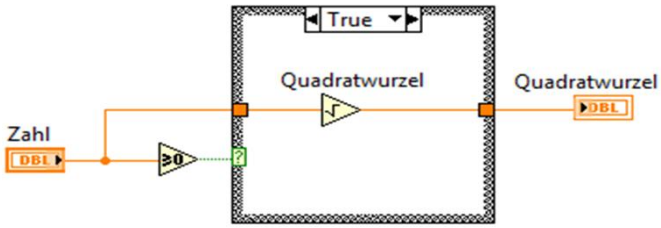
Enum-Case-Strukturen

Fehler-Case-Strukturen

Ein- und Ausgangstunnel (Wiederholung Basics)

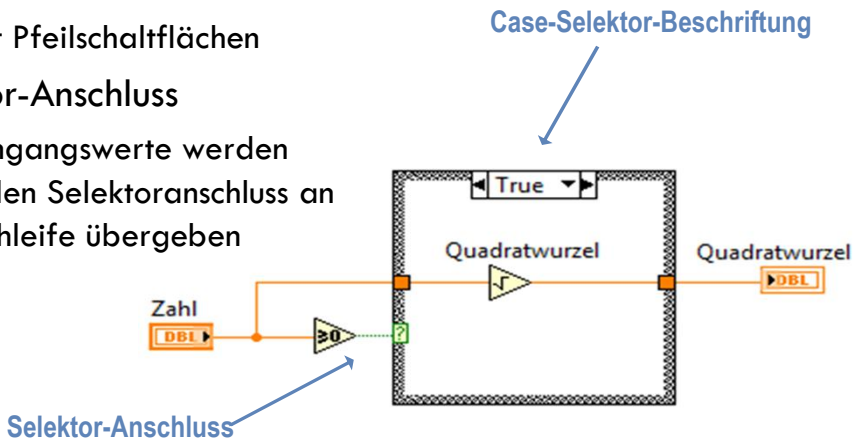
Case-Strukturen

- Haben mindestens zwei Unterdiagramme (Cases)
- Ermitteln anhand des Eingangswerts den auszuführenden Case
- Führen jeweils nur ein Unterdiagramm aus und zeigen dieses an
- Ähneln **case**- oder **if...then...else**-Anweisungen in befehlsorientierten Programmiersprachen



Case-Strukturen

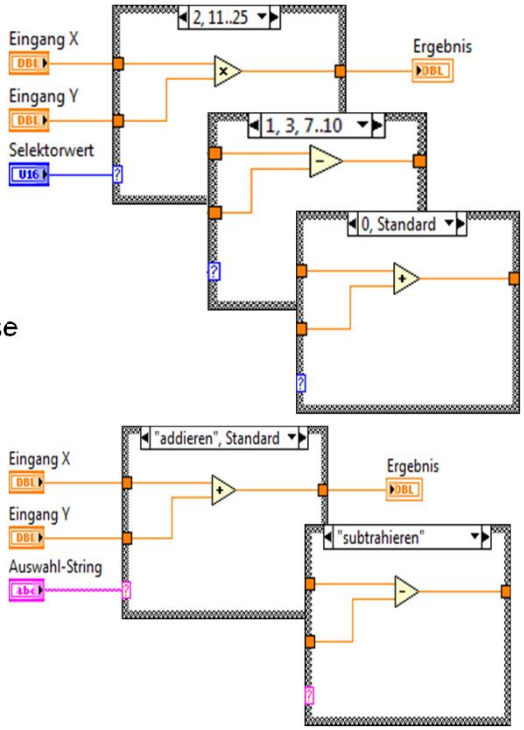
- Case-Selektor-Beschriftung
  - Gibt den Namen des aktuellen Cases an
  - Enthält Pfeilschaltflächen
- Selektor-Anschluss
  - Die Eingangswerte werden über den Selektoranschluss an die Schleife übergeben



Case-Strukturen

Selektoranschluss-Datentypen:

- Boolesch
  - TRUE- und FALSE-Case
- Fehler-Cluster
  - Fehler-Case/Kein-Fehler-Case
- Integer, String oder Enum
  - Struktur kann eine beliebige Anzahl von Cases enthalten
  - Enthält ein Standarddiagramm, so dass nicht jeder mögliche Eingangswert aufgeführt werden muss



Boolean input to the selector terminal creates two cases: True and False

Integer or string input to the selector terminal creates multiple cases.

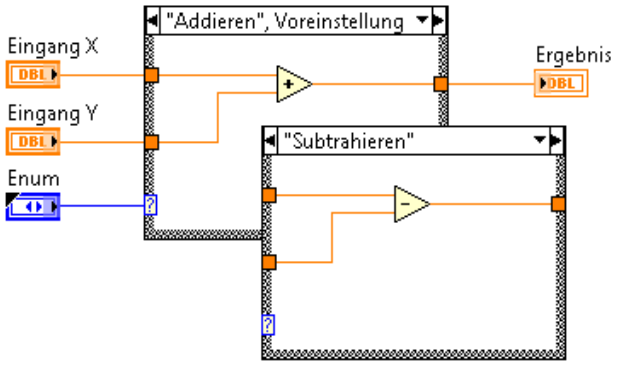
- Developer must add a case for each integer or string as necessary.
- If an undefined integer or strings is wired to the terminal, LabVIEW uses the Default case.

Enums

- Gives users a list of items from which to select a case.
- The case selector displays a case for each item in the enumerated type control.

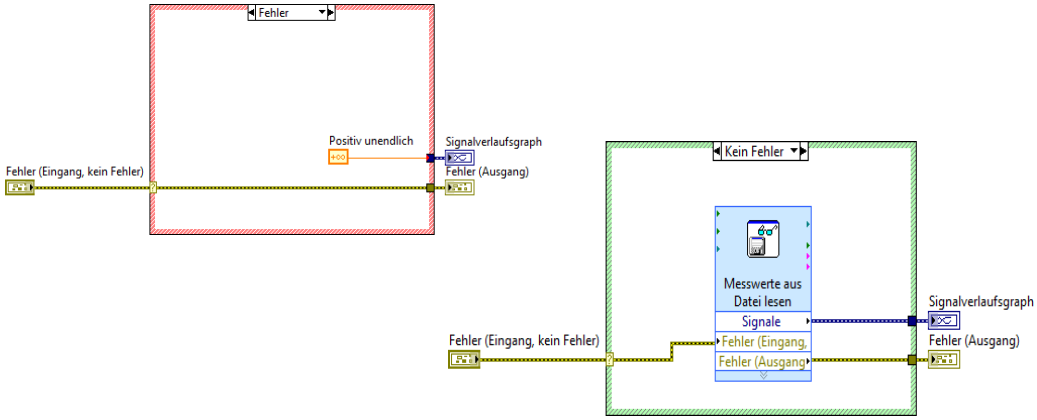
Enum-Case-Struktur

- Zeigt Benutzern eine Liste mit auswählbaren Einträgen an
- Im Case-Selektor wird zu jedem Eintrag im Enum-Bedienelement ein Case angezeigt



Fehler-Case-Struktur

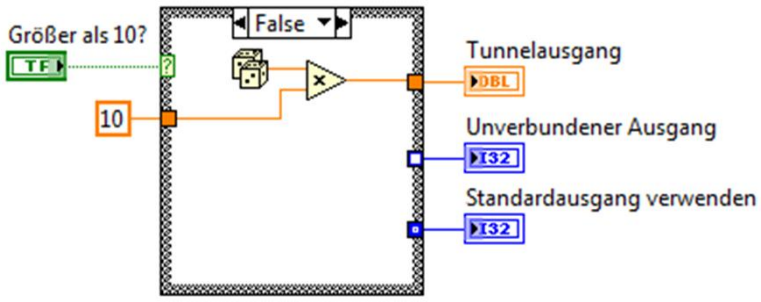
Mit Case-Strukturen können fehlerhafte Blockdiagramm-abschnitte übersprungen werden



Ein- und Ausgangstunnel

Sie können mehrere Ein- und Ausgangstunnel erstellen

- Eingangstunnel gelten für alle Cases
- Ausgangstunnel müssen für jeden Case festgelegt werden



Point out difference between tunnels that have been wired for all cases, tunnels that have not, and tunnels marked as Use Default If Unwired.

Point out that the unwired output tunnel will result in a broken **Run** arrow. The other two options will not cause a broken **Run** arrow.

Zoom in on the tunnels so that students can clearly see the differences.

Demonstrate how to enable or disable the Use Default If Unwired option.

Tell students to avoid using the Use Default If Unwired option on Case structure tunnels for the following reasons.

- Adds a level of complexity to the code.
- Complicates debugging your code.

Verwenden von sequenziellen Algorithmen und Zustandsautomaten

A. Sequenzielle Programmierung

B. Zustandsprogrammierung

C. Zustandsautomaten

D. Entwurfsmuster in LabView

A. Sequenzielle Programmierung

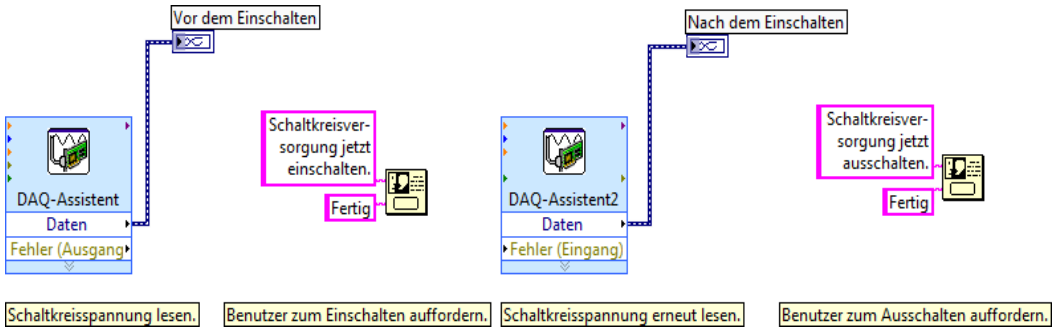
Durchflussparameter

Sequenzstrukturen

Fehler-Case-Strukturen

Sequenzielle Programmierung

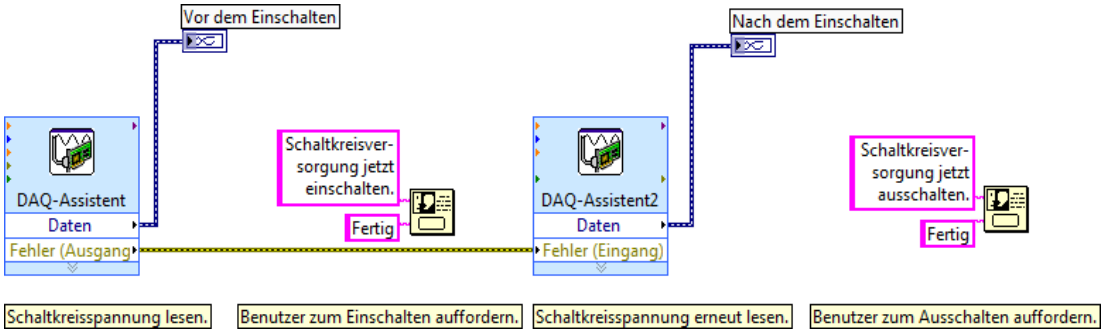
- VIs führen häufig aufeinanderfolgende Aufgaben aus
- Sequenzielle Programmierung ist aber in LabVIEW nicht zwingend



Beispiel: Die Ausführungsreihenfolge der folgenden Tasks ist nicht festgelegt; jeder Task kann theoretisch zuerst ausgeführt werden.

Mit Durchflussparametern bestimmte Reihenfolge

Die Ausführungsreihenfolge kann mit Fehler-Clustern und Referenzen vorgegeben werden



Use flow-through parameters, such as refnums and error clusters, to control execution order when natural data dependency does not exist.

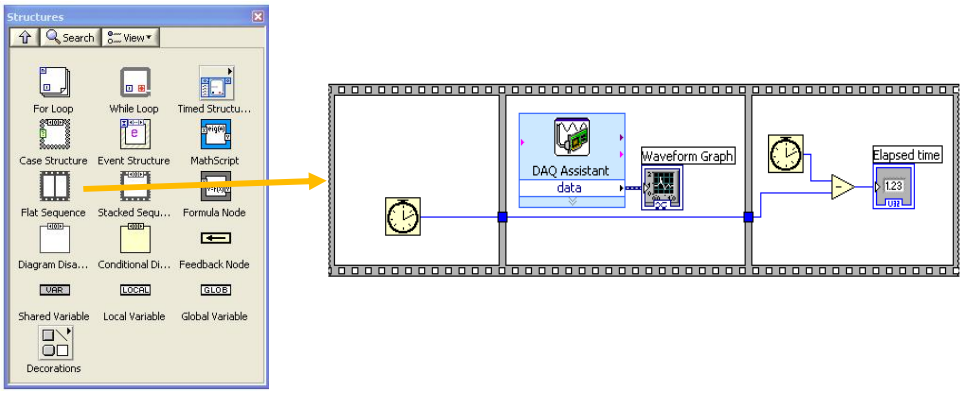
Not all LabVIEW nodes have flow-through parameters to ensure data dependency. For example, the One Button Dialog function does not have an error cluster input or output.

Sequence Structures

Functions >> Programming >> Structures

Executes diagrams sequentially

Right-click to add new frame



In a text-based language, program statements execute in the order in which they appear. In data flow, a node executes when data is available at all its input terminals. Sometimes it is hard to tell the exact order of execution. Often, certain events must take place before other events. When you need to control the order of execution of code in your block diagram, you can use a sequence structure.

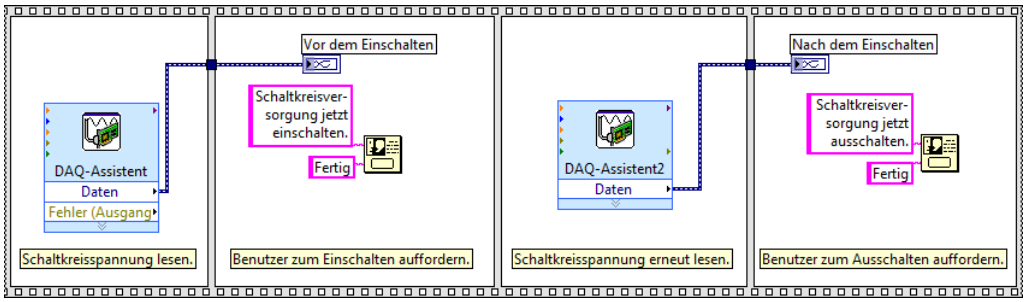
Sequence structure: Used to control the order in which nodes in a diagram will execute.

In the **Execution Control** subpalette.

- Looks like a frame of film.
- Used to execute diagrams sequentially.
- Right-click on the border to create a new frame.

Mit Sequenzstrukturen bestimmte Reihenfolge

- Sequenzstrukturen sind Strukturen, deren Rahmen nacheinander ausgeführt werden
- Der zweite Rahmen muss so lange mit der Ausführung warten, bis der erste beendet ist



Without flow-through parameters, such as error clusters and refnums, you can use sequence structures to the force execution order.

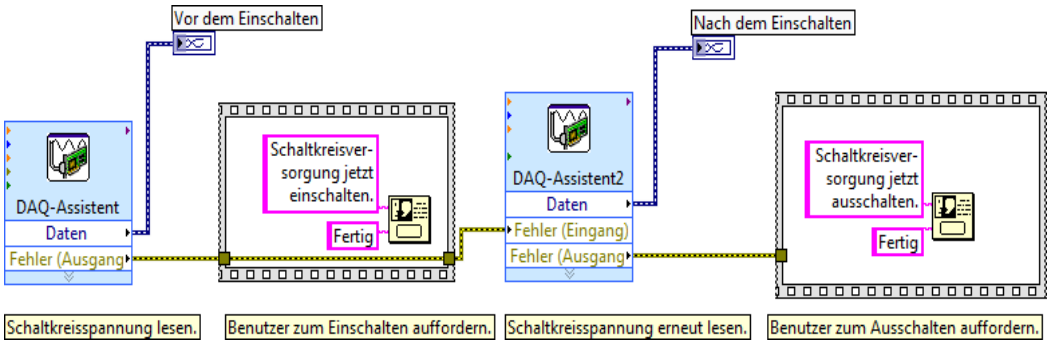
Sequence structures are not ideal and should be used sparingly.

- They do not take advantage of nodes that do support flow-through parameters.
- They do not guarantee that the update of terminals will happen before a dialog.

There are appropriate situations for Sequence structures, but in general, Sequence structures are not considered good LabVIEW programming.

Sequenzstrukturen sparsam nutzen

Die Ausführung lässt sich nicht inmitten einer Sequenz anhalten!

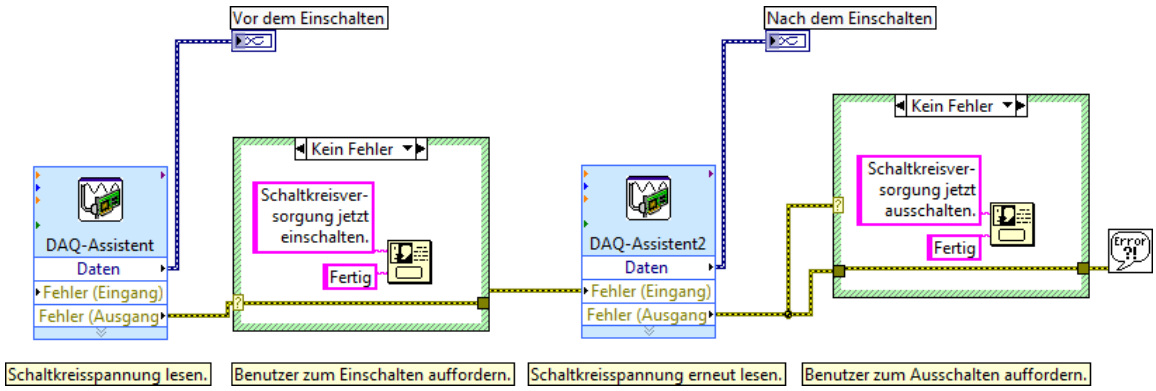


Single frame sequence structures are better than multi-sequence frames shown on previous slide.

In this diagram, the One Button Dialog functions pop-up even if the DAQ Assistant results in an error. What can we replace the sequence structures with?

Fehler-Case-Strukturen vorziehen

Ideal: Dialogfelder in Case-Strukturen platzieren und den Fehler-Cluster mit den Case-Selektoren verbinden



B. Zustandsprogrammierung

Zustandsprogrammierung

Zustandsübergangsdiagramme

Zustandsprogrammierung

Die Zustandsprogrammierung hilft bei Problemen, die mit sequenzieller Programmierung & Durchflussparametern nicht lösbar sind:

- Ändern der Abfolge
- Unterschiedliche Ausführungshäufigkeit der einzelnen Elemente
- Bedingte Ausführung von Elementen der Sequenz
- Sofortiges Beenden ohne Abwarten des Ausführungsendes

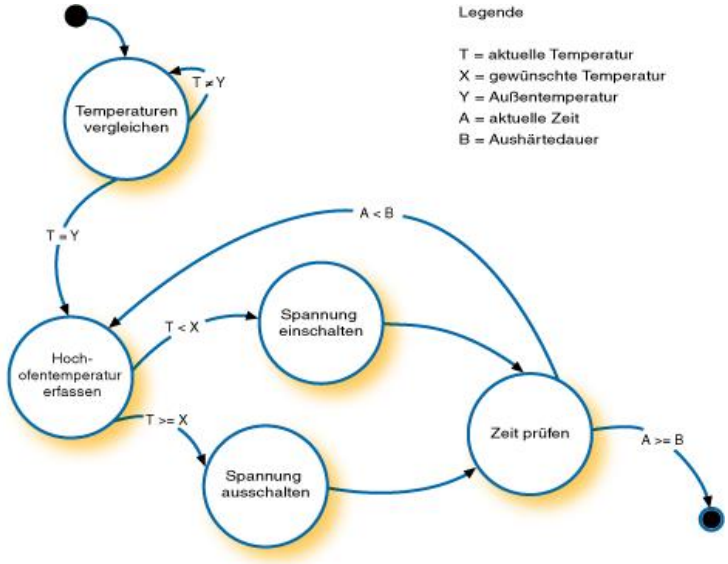
Zustandsübergangsdiagramm

Eine Art Ablaufdiagramm, das die Zustände eines Programms und Übergänge dazwischen anzeigt

- **Zustand:** Programmteil, der eine bestimmte Bedingung erfüllt, eine Handlung ausführt oder auf ein Ereignis wartet
- **Übergang:** Bedingung, Handlung oder Ereignis, bei der oder dem das Programm zum nächsten Zustand wechselt.

Zustandsübergangsdiagramm

Beispiel Hochofen:



C. Zustandsautomaten

Gängige Verwendungszwecke

Infrastruktur

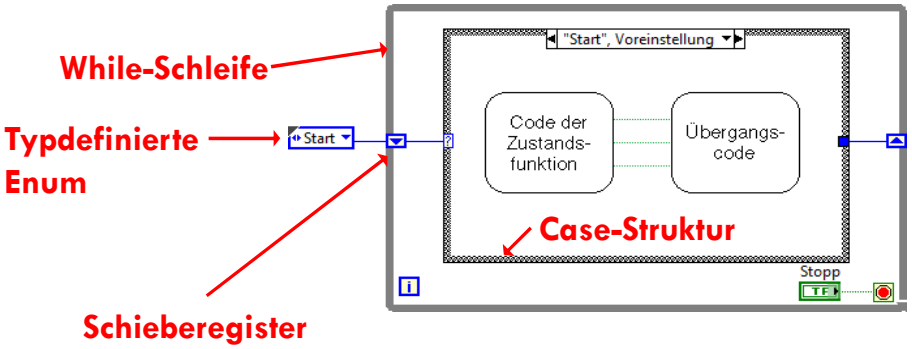
Übergänge

Zustandsautomaten

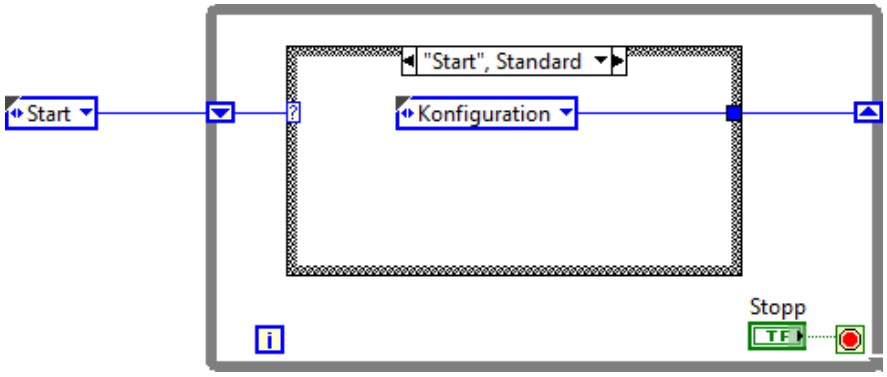
- Mit dem Automaten-Entwurfsmuster wird ein Zustands- oder Ablaufdiagramm implementiert
- Typische Verwendungszwecke von Zustandsautomaten:
  - Bei Bedienoberflächen (Zustand wechselt je nach Eingabe)
  - Bei der Prozessprüfung entspricht jeder Test einem Zustand

Aufbau des Zustandsautomaten

- Ein Automat hat mehrere Zustände und eine Übergangs-funktion zur Definition des nächsten Zustands
- Jeder Zustand kann zu einem oder mehreren Zuständen oder zum Ende des gesamten Prozesses führen.

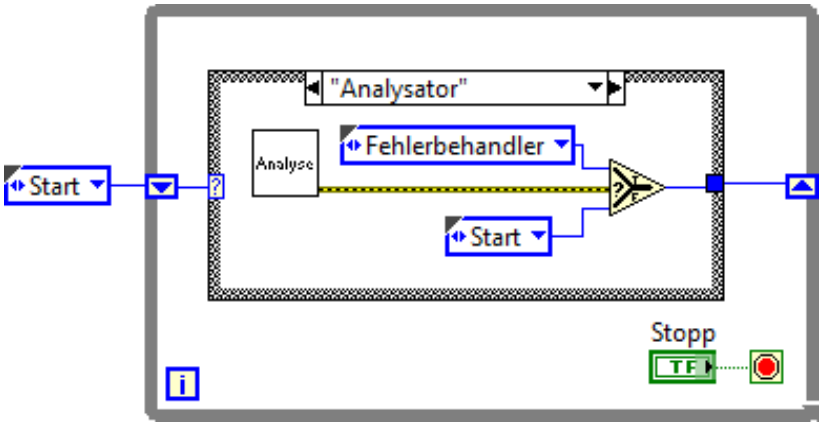


Zustandsautomaten – Standardübergang

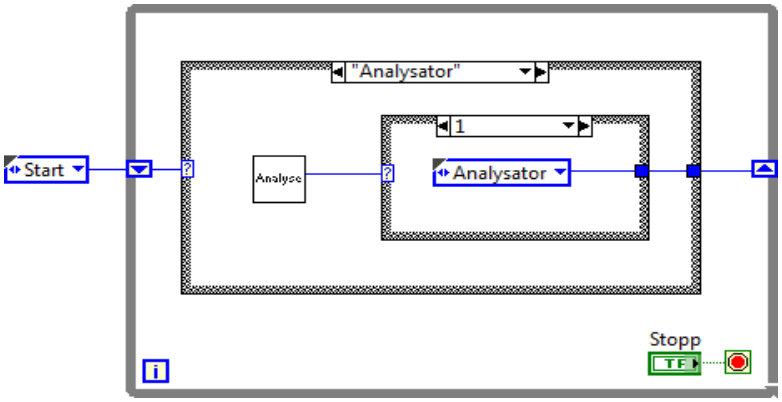


Point out here that a type defined enumerated control is a good method for controlling a state machine.

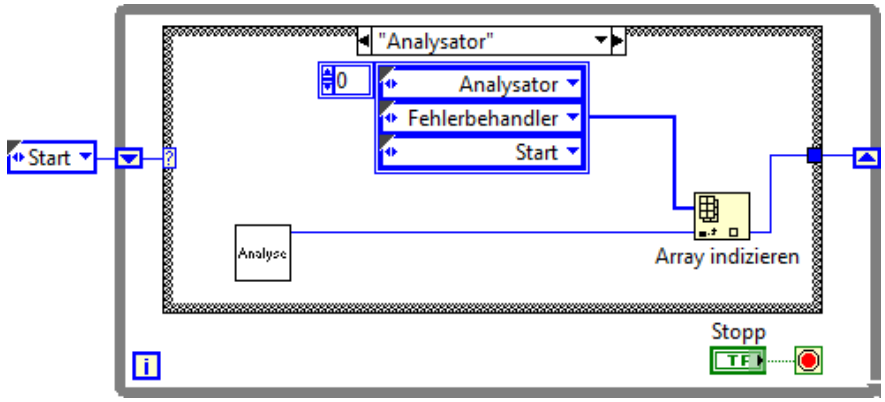
Zustandsautomaten – Übergang zwischen zwei Zuständen



Zustandsautomaten – Case-Struktur-Übergang



Zustandsautomaten – Array-Übergang



D. Entwurfsmuster in LabView

Simple VI Architecture

General VI Architecture

State Machine Architecture (siehe vorher B+C)

Erzeuger/Verbraucher (Ereignisse)

In this class we will go into depth covering the Producer/Consumer (Events) design pattern.

Other multi-loop design patterns include the Master/Slave and Queued Message Handler design patterns.

The Master/Slave is a slight variation of the Producer/Consumer design pattern.

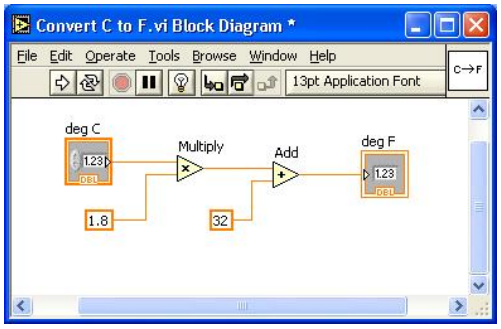
The Queued Message Handler is another loop-based design pattern.

Simple VI Architecture

Functional VI that produces results when run

- No “start” or “stop” options
- Suitable for lab tests, calculations

Example: Convert C to F.vi



Simple VI Architecture

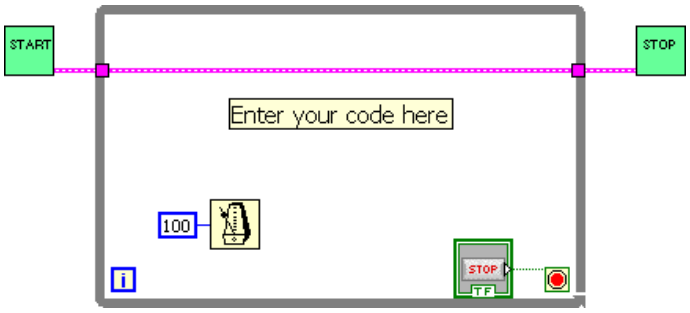
When making quick lab measurements, you do not need a complicated architecture. Your program may consist of a single VI that takes a measurement, performs calculations, and either displays the results or records them to disk. The measurement can be initiated when the user clicks on the run arrow.

In addition to being commonly used for simple applications, this architecture is used for “functional” components within larger applications. You can convert these simple VIs into subVIs that are used as building blocks for larger applications.

General VI Architecture

Three Main Steps

- Startup
- Main Application
- Shutdown



General VI Architecture

In designing an application, you generally have up to three main phases:

•*Startup*—Use this area to initialize hardware, read configuration information from files, or prompt the user for data file locations.

•*Main Application*—Generally consists of at least one loop that repeats until the user decides to exit the program, or the program terminates for other reasons such as I/O completion.

•*Shutdown*—This section usually takes care of closing files, writing configuration information to disk, or resetting I/O to its default state.

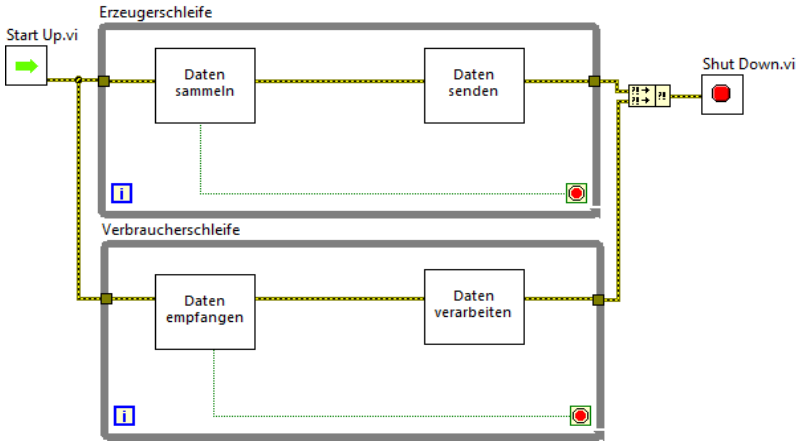
The diagram above shows this general architecture.

For simple applications, the main application loop can be fairly straightforward. When you have complicated user interfaces or multiple events (user action, I/O triggers, and so on), this section can get more complicated.

Erzeuger/Verbraucher Entwurfsmuster

- 2 Whileschleifen (Datenerzeugung + Datenverarbeitung getrennt)
- Oft mit Queues

(siehe SS)



The producer/consumer design pattern separates tasks that produce and consume data at differ rates. The parallel loops in the producer/consumer design pattern are separated into two categories, those that produce data and those that consume data- Data queues communicate data among the loops and buffer data among the producer and consumer loops.

Zusammenfassung – Quiz

1. Welche der folgenden Sachverhalte treffen bei einem funktionsuntüchtigen VI zu?
   a) Ein SubVI ist nicht ausführbar
   b) Das Blockdiagramm enthält eine Division durch Null
   c) Ein erforderlicher SubVI-Eingang wurde offen gelassen
   d) Ein boolesches Objekt wurde mit einer Zahlenanzeige verbunden

Answer is a, c, and d.

Zusammenfassung – Antworten

1. Welche der folgenden Sachverhalte treffen bei einem funktionsuntüchtigen VI zu?
   - **a) Ein SubVI ist nicht ausführbar**
   - b) Das Blockdiagramm enthält eine Division durch Null
   - **c) Ein erforderlicher SubVI-Eingang wurde offen gelassen**
   - d) Ein boolesches Objekt wurde mit einer Zahlenanzeige verbunden

A divide by zero might cause unexpected results but will not cause the run arrow to be broken.

Zusammenfassung – Quiz

2. Welche der nachfolgenden Komponenten gehören zu einem Fehler-Cluster?
   a) Status: Boolesch
   b) Fehler: String
   c) Code: 32-Bit-Integer
   d) Quelle: String

Answer is a, c, and d.

Zusammenfassung – Antworten

2. Welche der nachfolgenden Komponenten gehören zu einem Fehler-Cluster?
   - **a) Status: Boolesch**
   - b) Fehler: String
   - **c) Code: 32-Bit-Integer**
   - **d) Quelle: String**

Zusammenfassung – Quiz

3. Alle Fehler haben negative Fehlercodes und alle Warnungen haben positive Fehlercodes.
   a) Richtig
   b) Falsch

Answer is b.

Zusammenfassung – Antworten

3. Alle Fehler haben negative Fehlercodes und alle Warnungen haben positive Fehlercodes.
   a) Richtig
   b) **Falsch**

In an error cluster, a **status** of TRUE is always an error regardless of the **code** value. A non-zero **code** with a **status** of FALSE is considered a warning.

Zusammenfassung – Quiz

4. Die Funktion "Fehler zusammenfassen" verknüpft Fehlerangaben aus mehreren Quellen.
   a) Richtig
   b) Falsch

Answer is b.

Zusammenfassung – Antworten

4. Die Funktion "Fehler zusammenfassen" verknüpft Fehlerangaben aus mehreren Quellen.
   a) Richtig
   b) **Falsch**

Merge Errors function returns the first error found. If no error is found, it returns the first warning.

The Merge Errors function does not concatenate errors

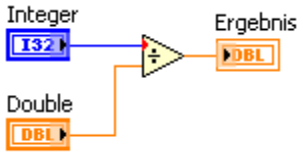
Zusammenfassung – Quiz

5. Was bedeutet es, wenn der Eingang einer Funktion mit einem roten Punkt (Typumwandlungspunkt) markiert ist?
   a) Es wurden Daten in eine Struktur übertragen
   b) Es wurde eine For-Schleife mit Bedingungsanschluss erstellt
   c) Ein Iterationsanschluss einer For-Schleife ist nicht verbunden
   d) Der an einen Knoten geleitete Wert wurde in seiner Darstellung geändert

Answer is d.

Zusammenfassung – Antworten

5. Was bedeutet es, wenn der Eingang einer Funktion mit einem roten Punkt (Typumwandlungspunkt) markiert ist?
   a) Es wurden Daten in eine Struktur übertragen
   b) Es wurde eine For-Schleife mit Bedingungsanschluss erstellt
   c) Ein Iterationsanschluss einer For-Schleife ist nicht verbunden
   d) Der an einen Knoten geleitete Wert wurde in seiner Darstellung geändert



Zusammenfassung – Quiz

6. Welche Struktur muss mindestens einmal ausgeführt werden?
   a) While-Schleife
   b) For-Schleife

Answer is a.

Zusammenfassung – Antworten

6. Welche Struktur muss mindestens einmal ausgeführt werden?
   a) **While-Schleife**
   b) For-Schleife

Zusammenfassung – Quiz

7. Was gibt es nur im Blockdiagramm?

   Bedienelement

   Konstante

   Anzeigeelement

   Anschlussfeld

Answer is b.

Zusammenfassung – Antworten

7. Was gibt es nur im Blockdiagramm?

a) Bedienelement

b) Konstante

c) Anzeigeelement

d) Anschlussfeld

Answer is b.

Zusammenfassung – Quiz

8. Welches Schaltverhalten führt dazu, dass ein boolesches Element beim Anklicken von FALSE auf TRUE wechselt und bis zum Auslesen des Werts auf TRUE verbleibt?
   a) Bis zum Loslassen schalten
   b) Beim Loslassen schalten
   c) Latch bis zum Loslassen
   d) Latch beim Loslassen

Answer is d.

Zusammenfassung – Antworten

8. Welches Schaltverhalten führt dazu, dass ein boolesches Element beim Anklicken von FALSE auf TRUE wechselt und bis zum Auslesen des Werts auf TRUE verbleibt?
   a) Bis zum Loslassen schalten
   b) Beim Loslassen schalten
   c) Latch bis zum Loslassen
   d) Latch beim Loslassen

Zusammenfassung – Quiz

9. Es können keine Arrays aus Arrays erzeugt werden.
   a) Richtig
   b) Falsch

Answer is False.

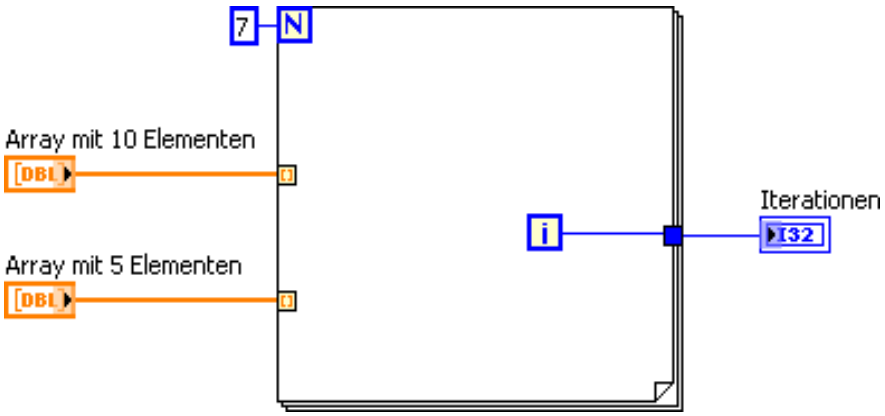
Zusammenfassung – Antworten

9. Es können keine Arrays aus Arrays erzeugt werden.
   a) Richtig
   b) Falsch

   Ein Array kann nicht in einen Array-Container eingefügt werden. Sie können aber zweidimensionale Arrays erstellen.

Zusammenfassung – Quiz

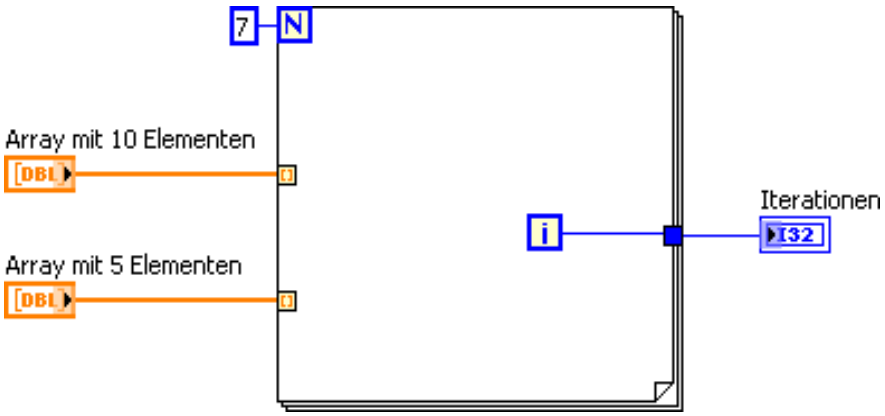
10. Welchen Wert hat die Anzeige Iterationen nach der Ausführung des VIs?



Answer is 4.

Zusammenfassung – Antworten

10. Welchen Wert hat die Anzeige Iterationen nach der Ausführung des VIs? **4**

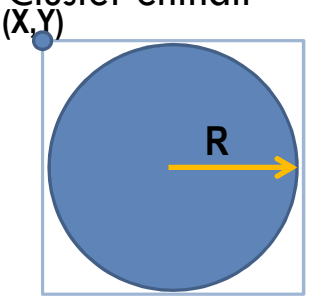


LabVIEW does not exceed the array size. This helps to protect against programming error. So the loop runs 5 times.

Although the for loop runs 5 times, the iterations are zero-based. Therefore the value of the **Iterations** indicators is 4.

Zusammenfassung – Quiz

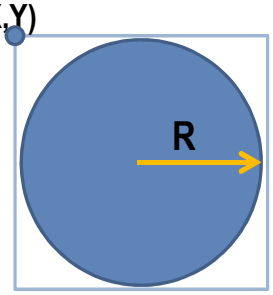
11. Ein Kreis ist durch x, y-Koordinaten und einen Radius definiert. In Zukunft soll evtl. die Kreisfarbe hinzukommen. Welche Datenstruktur sollte für den Kreis genutzt werden?
    a) Drei verschiedene Elemente (für Koordinaten und Radius)
    b) Cluster aus allen Werten
    c) Benutzerdefiniertes Element, das einen Cluster enthält
    d) Typdefinition, die einen Cluster enthält
    e) Array aus drei Elementen



Answer is d.

Zusammenfassung – Antworten

11. Ein Kreis ist durch x, y-Koordinaten und einen Radius definiert. In Zukunft soll evtl. die Kreisfarbe hinzukommen. Welche Datenstruktur sollte für den Kreis genutzt werden?
    a) Drei verschiedene Elemente (für Koordinaten und Radius)
    b) Cluster aus allen Werten
    c) Benutzerdefiniertes Element, das einen Cluster enthält
    d) Typdefinition, die einen Cluster enthält
    e) Array aus drei Elementen



Zusammenfassung – Quiz

11. Bei Verwendung einer Sequenzstruktur können Sie die Ausführung inmitten der Sequenz abbrechen.

    a) Richtig

    b) Falsch

Answer if False.

Zusammenfassung – Antworten

11. Bei Verwendung einer Sequenzstruktur können Sie die Ausführung inmitten der Sequenz abbrechen.

a) Richtig

b) Falsch

Die Ausführung lässt sich nicht inmitten einer Sequenz anhalten

Zusammenfassung – Quiz

12. Welche Vorteile bietet die Verwendung eines Zustandsautomaten anstelle einer Sequenzstruktur?
    a) Die Abarbeitungsreihenfolge kann geändert werden.
    b) Die Bestandteile der Abfolge können wiederholt werden.
    c) Elemente in der Abfolge können Bedingungen unterliegen.
    d) Das Programm kann an einer beliebigen Stelle angehalten werden.

Answer is a, b, c, and d.

Zusammenfassung – Antworten

12. Welche Vorteile bietet die Verwendung eines Zustandsautomaten anstelle einer Sequenzstruktur?
    a) **Die Abarbeitungsreihenfolge kann geändert werden.**
    b) **Die Bestandteile der Abfolge können wiederholt werden.**
    c) **Elemente in der Abfolge können Bedingungen unterliegen.**
    d) **Das Programm kann an einer beliebigen Stelle angehalten werden.**