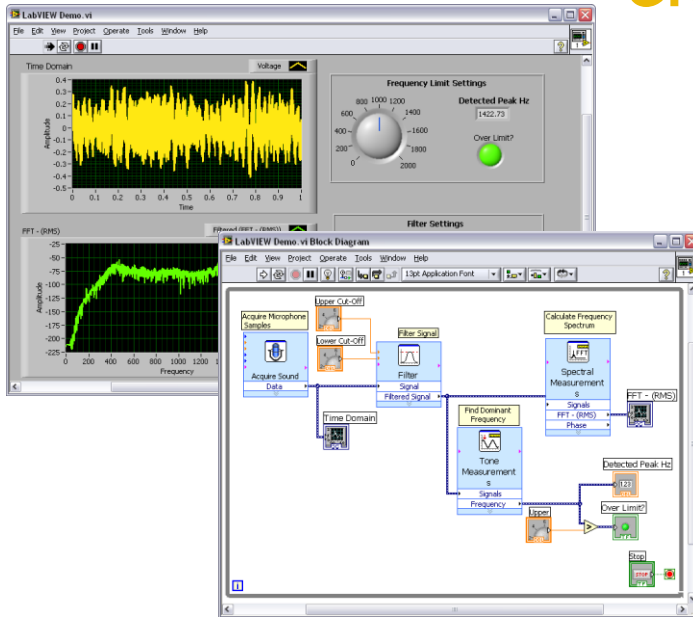


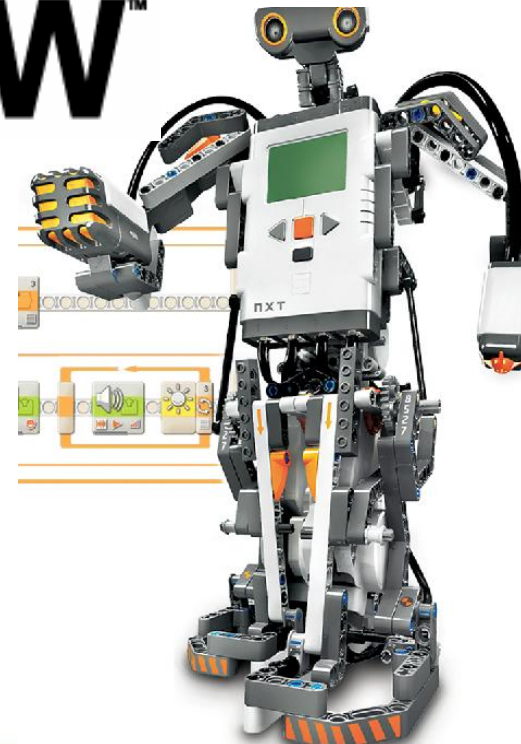
# Computerunterst. Messwerterfassung

GRAPHICAL PROGRAMMING

FOR ENGINEERS AND SCIENTISTS



# LabVIEW™



DI (FH) Werner Bischof  
[werner.bischof@bischof-it.at](mailto:werner.bischof@bischof-it.at)  
[bf@htl-Kapfenberg.at](mailto:bf@htl-Kapfenberg.at)

# Notenfindung

## A) wöchentliche Moodle Multiple Choice Tests:

3-5 Fragen; 5min; ab 50% positiv

Stoff jeweils bis zur letzten Einheit

## B) Mitarbeit und Aufgaben:

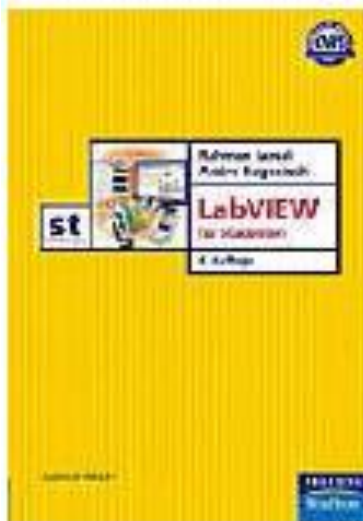
Kriterien: Problemlösung; Funktionalitätswissen!; Dokumentation;  
Kontexthilfe; Icon; Fehlerverarbeitung; Modularisierung (SubVIs);  
Anwesenheit

**Ziel: Projektarbeit mit LabVIEW oder CLAD (Maturafach)**

**bei Problemen mit Programmen > rufen Sie mich > mailen Sie mir das Programm!**



# Deutsche Fachbücher mit der Studentenversion von LabVIEW



**LabVIEW für Studenten**  
Autor: R. Jama/A. Hagedorn  
Verlag: Pearson Studium, 08/2004  
576 Seiten;  
mit CD-ROM (LabVIEW7);  
4., veränderte Ausgabe  
Preis: 49,95 €  
ISBN 3-8273-7154-6



**Einführung in LabVIEW**  
Autoren: W. Georg/E. Metin  
Verlag: Hanser Fachbuch-verlag  
Leipzig, 03/2006  
(2. Ausgabe 09/2006)  
328 Seiten; broschiert  
Preis: 39,90 €  
ISBN 3-446-40400-7



**Elektrische Messtechnik**  
Autor: R. Lerch  
Verlag: Springer Verlag 09/2006  
600 Seiten  
Preis: 42,95 €  
ISBN 3-540-34055-6



**Handbuch für die Programmierung mit LabVIEW**  
Autor: B. Müttelein  
Verlag: Elsevier Verlag,  
ab Jan 2007; 460 Seiten  
Preis: 49,50 €  
ISBN (978)3-8274-1761-9

# Inhalt der Vorlesung

- Arbeiten mit einer Programmiersprache (LabVIEW)
  - Erstellen von eigenen Programmen.
  - Erstellen von Unterprogrammen (subroutines); Modularisierung
  - Standard State Machine; Event Handler; Flußdiagramme; Producer-Consumer-Design; Master-Slave-Design;
  - Datenerfassung: I/O Karte; GPIB-Bus (Oszi,...); RS232;...
  - Allg. Probleme mit LabVIEW lösen. (Q137; Ampel; Datenerfassung; ...)
- LabVIEW-Konzept
  - Meßwerte erfassen, sichern und bearbeiten
  - Verwenden von mathematischen und komplexen Analysefunktionen
  - Arbeiten mit verschiedenen Datentypen zB. Arrays, Cluster
  - Darstellen und drucken der Ergebnisse

# Kapitelübersicht im Skriptum

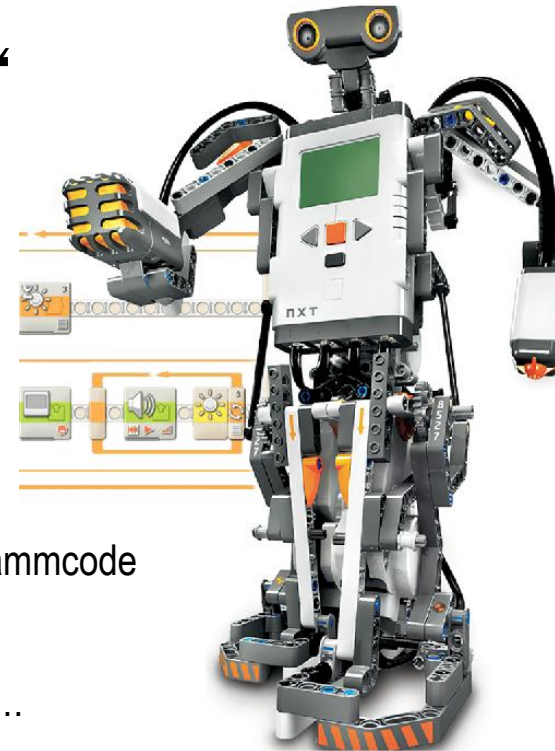
- Kapitel 1 Einführung in LabVIEW
- Kapitel 2 Einführung in virtuelle Instrumente
- Kapitel 3 LabVIEW-Umgebung
- Kapitel 4 Erstellen des Frontpanels
- Kapitel 5 Erstellen des Blockdiagramms
- Kapitel 6 Ausführen und Fehlersuche in VIs
- Kapitel 7 Erstellen von VIs und SubVIs
- Kapitel 8 Schleifen und Strukturen
- Kapitel 9 Gruppieren von Daten mit Strings, Arrays und Clustern
- Kapitel 10 Graphen und Diagramme
- Kapitel 11 Datei-I/O
- Kapitel 12 Dokumentieren und Drucken von VIs
- Addons GPIB; VISA; DAQ-Karte siehe Übungsskript

# Kapitel 1 Einführung in LabVIEW

- (Laboratory Virtual Instrument Engineering Workbench) ist eine grafische Programmiersprache, die Symbole anstelle von Textzeilen verwendet, um Applikationen zu erstellen.

„LabVIEW ist wie Lego!“

- textbasierte Programmiersprachen, sind befehlsorientiert, LabVIEW ist datenflussorientiert.
- VIs oder virtuelle Instrumente sind LabVIEW-Programme, mit denen Geräte nachgebildet werden.
- Frontpanel HMI
- Blockdiagramm hier befindet sich der Quellcode auch als G- oder Blockdiagrammcode
  - Blockdiagramm ähnelt einem Flussdiagramm.
- viele verschiedene Toolkits für LabVIEW z.B.: PID, Real Time; NXT-Toolkit;....

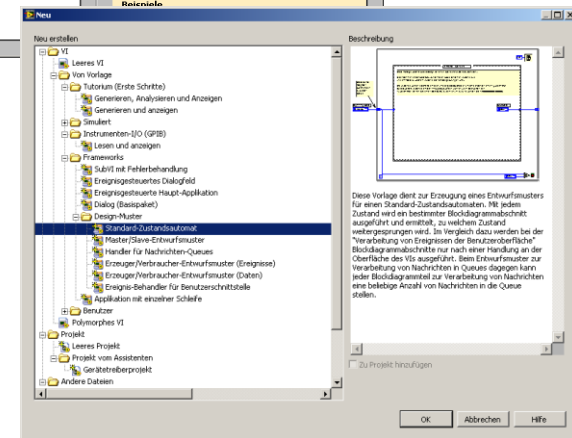
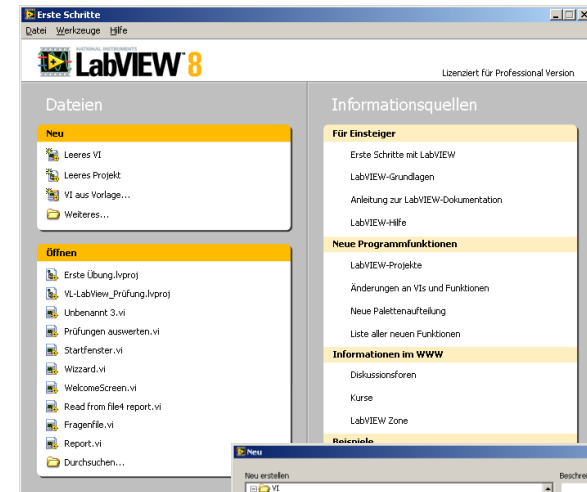


# VI-Vorlagen, Beispiel-VIs

- VI-Vorlagen:  
SubVIs,  
Funktionen,  
Strukturen und  
Frontpanel-Objekte.

für allgemeine Anwendungen der Messtechnik  
und den Einstieg ins Programmieren

(wählen Sie Datei»Neu, um zum Dialogfeld Neu zu  
gelangen.)





# VI-Vorlagen, Beispiel-VIs

- Hunderte von Beispiel-VIs, die sich in Ihre eigenen VIs integrieren lassen.

(Lissajous; moon; 3D-solarsystem)

- Beispiele

veränderbar

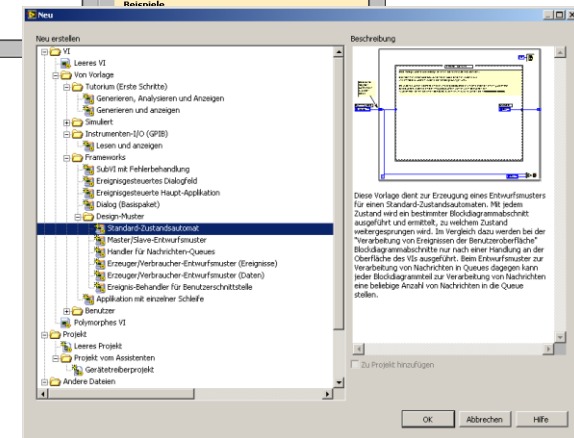
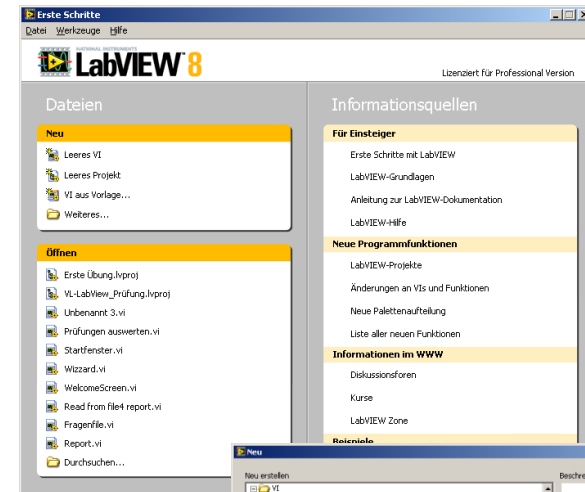
kopierbar

erweiterbar

Programmstil ist erkennbar und

die Beispiele sind dokumentiert !!

(Suchen Sie Beispiel-VIs mit Hilfe der NI-Suchmaschine. Wählen Sie dazu Hilfe»Beispiele suchen.)



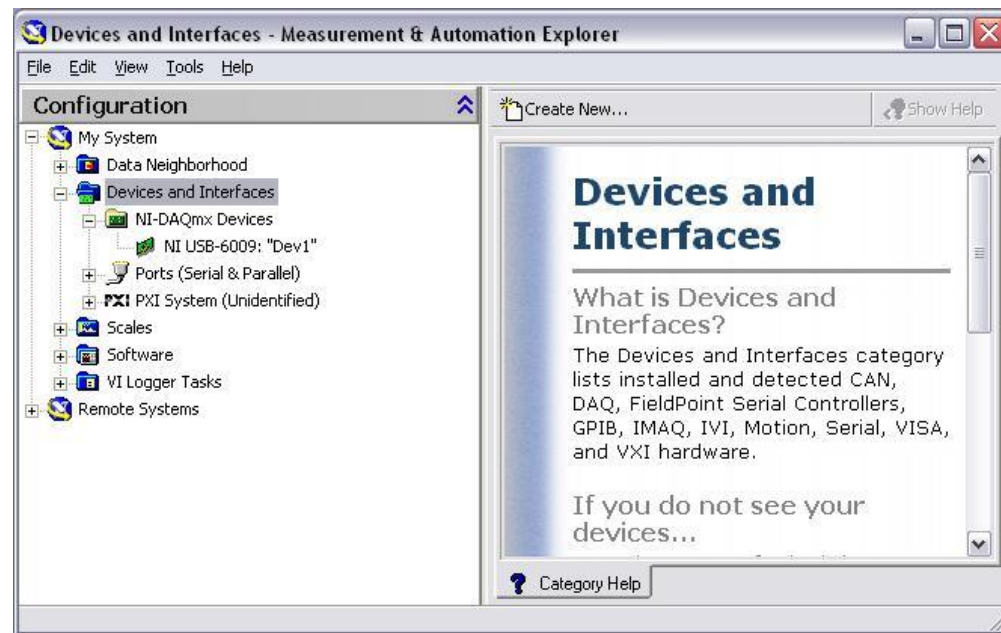


# MAX Measurement & Automation Explorer

- Konfiguration von Messgeräten.
- Hard- und Software von National Instruments. (DAQ-Konfiguration; GPIB)
- VI Logger



Measurement  
& Automation



# Kapitel 2 Einführung in virtuelle Instrumente

- LabVIEW-Programme werden als VIs (Virtuelle Instrumente) bezeichnet, weil sie in
    - Erscheinungsbild und
    - Funktion echten Messgeräten wie Oszilloskopen oder Multimetern nachgebildet werden.
  - VIs arbeiten mit Funktionen, die
    - Eingaben von der Benutzeroberfläche oder
    - aus anderen Quellen verarbeiten.
  - Ein VI enthält die folgenden drei Komponenten:
    - **Frontpanel:** Bedienoberfläche (HMI).
    - **Blockdiagramm:** Quellcode, -> Funktionen des VIs.
    - **Symbol und Anschlussfeld:** Kennzeichnet das VI, so dass Sie es in einem anderen VI verwenden können.
- Ein VI, das einem anderen VI untergeordnet ist, wird als SubVI bezeichnet. Ein SubVI entspricht einem Unterprogramm in textbasierten Programmiersprachen.

# Frontpanel: Bedien- und Anzeigeelemente

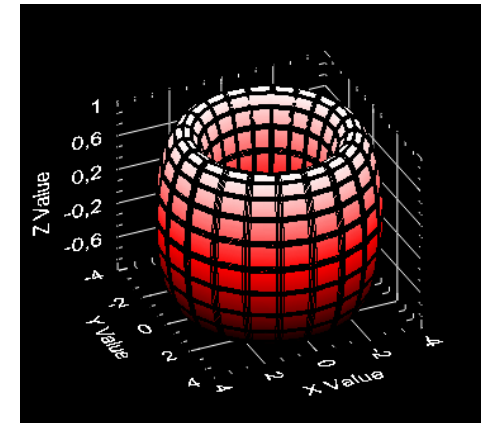
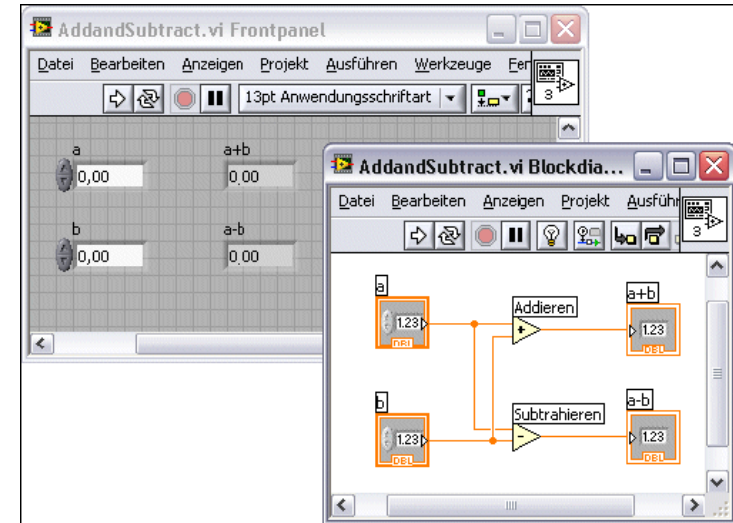
-stellen die interaktiven Ein- und Ausgangsanschlüsse eines VIs dar.

1. Bedienelemente: Drehschalter, Druckschalter oder Drehregler.

Sie dienen dazu, das Blockdiagramm des VIs mit Daten zu versorgen.

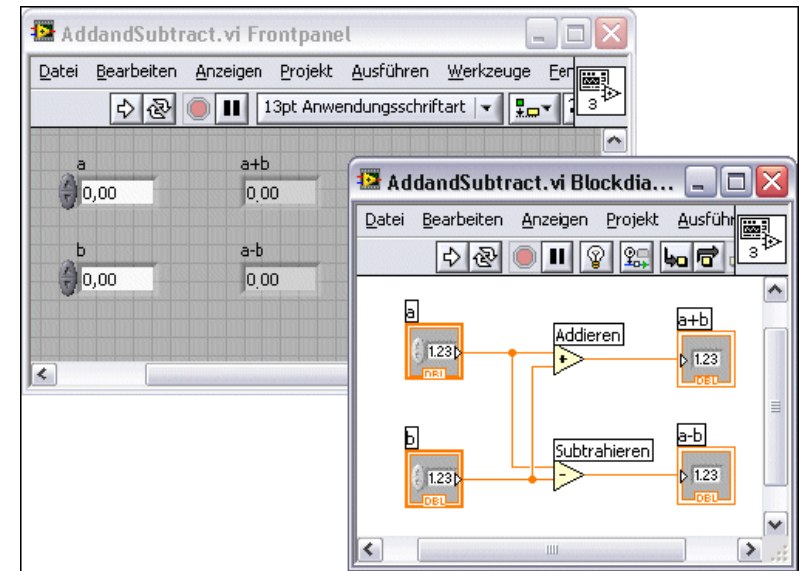
2. Anzeigeelementen alle Ausgabeelemente wie Graphen oder LEDs.

Mit Anzeigeelementen werden die Ausgänge und Ausgabeelemente von Messgeräten simuliert und die vom Blockdiagramm ausgegebenen Werte angezeigt.



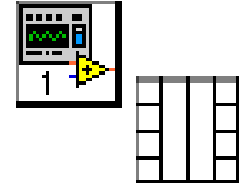
# Blockdiagramm

- Frontpanel erstellen, anschließend können Sie mit Hilfe grafisch dargestellter Funktionen Programmcode hinzufügen, um die Frontpanel-Objekte zu steuern.
- (oder doch besser umgekehrt)
- Das Blockdiagramm enthält den grafischen Quellcode, der auch als G- oder Blockdiagramm-Code bezeichnet wird.

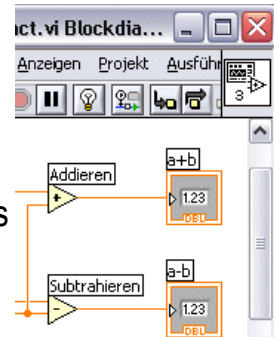


Symbole, Funktionen und Verbindungen.

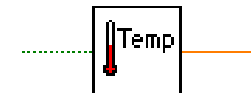
# Symbol- und Anschlussfeld



- VI (z.B.: addieren und subtrahieren von 2 Zahlen) benötigt ein
  - VI-Symbol (Icon), und das
    - Das VI-Symbol ist die grafische Darstellung eines VIs.
    - Es kann eine Kombination aus Text- und Grafikelementen enthalten.
    - Wenn Sie ein VI als SubVI verwenden, kennzeichnet das Symbol das SubVI im Blockdiagramm des Haupt-VIs.
    - Um das Symbol zu editieren und zu verändern, führen Sie einen Doppelklick darauf aus.
  - Anschlussfeld, mit denen das VI als SubVI (Unterprogramm) verwendet werden kann.
    - mehrere Anschlüsse, die den Bedien- und Anzeigeelementen dieses VIs entsprechen.
    - Eingänge und Ausgänge festgelegt, die mit dem VI verbunden werden müssen, damit es als SubVI genutzt werden kann.



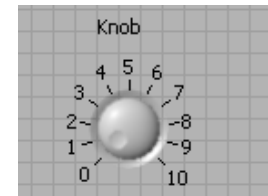
Weitere Informationen zu Symbolen finden Sie im Kapitels 7



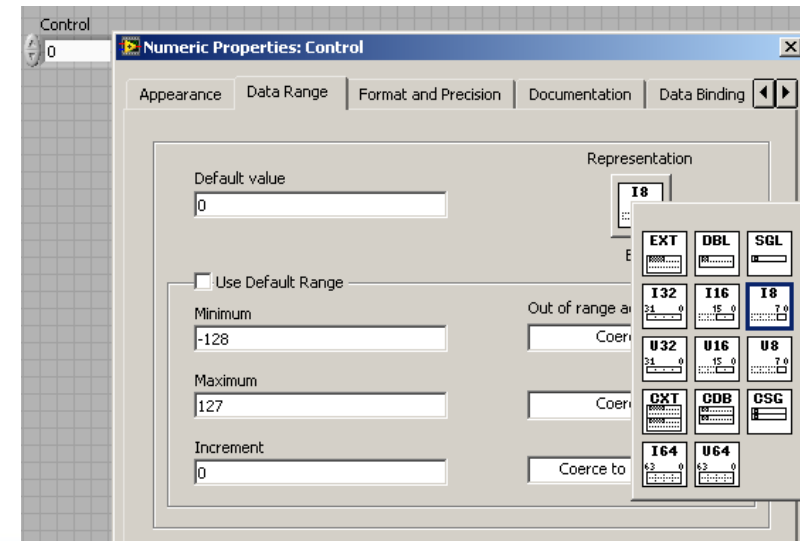
# Symbole

- zeigen den Datentypen eines Bedien- oder Anzeigeelements an entweder als:

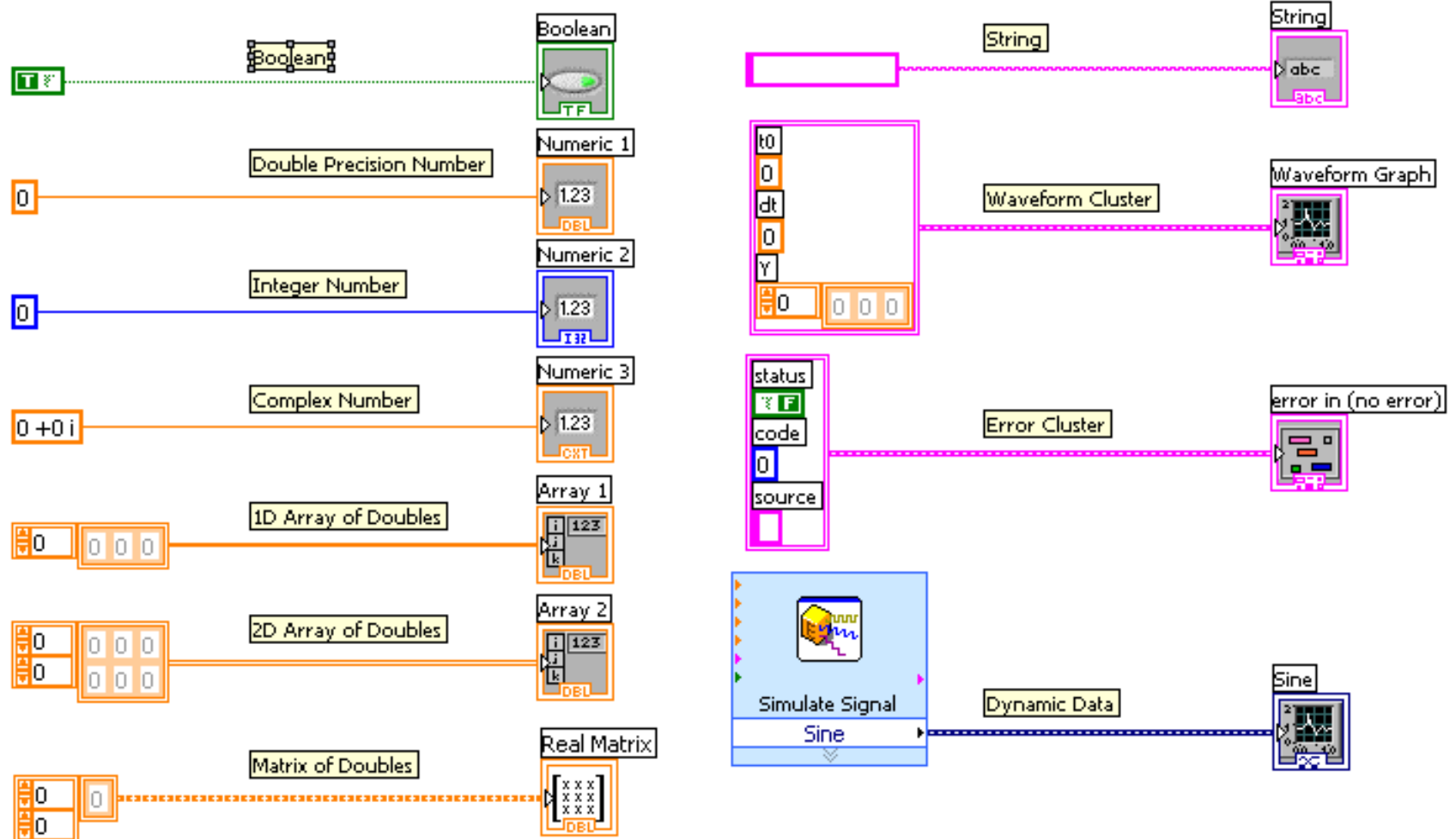
- Bild oder als
- Datentyp-Symbol dargestellt werden.



z.B.: mit einem Drehschaltersymbol ein auf dem Frontpanel befindlicher Drehschalter dargestellt.  
“DBL” (64Bit; von + 1.79e+308 bis –1.79e+308  
siehe Eigenschaften eines Controls



# Datentypen in LabVIEW





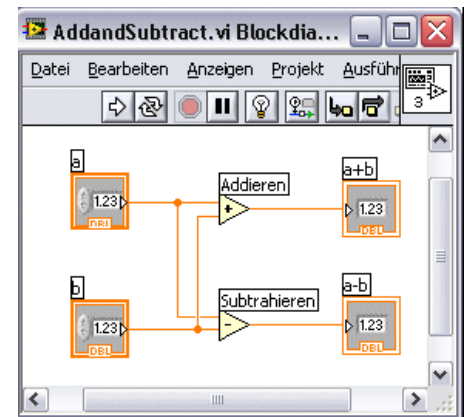
# Knoten

- sind Objekte im Blockdiagramm,
  - mit Ein- und/oder Ausgängen und
  - in einem laufenden VI bestimmte Funktionen ausführen.

## Knoten entsprechen

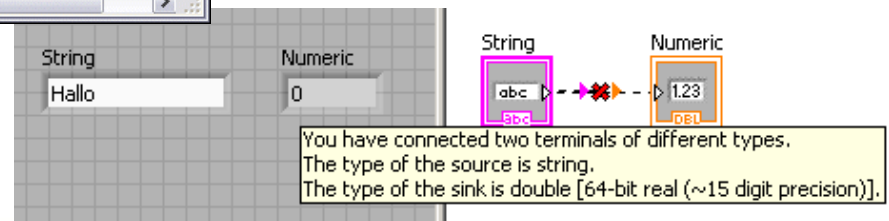
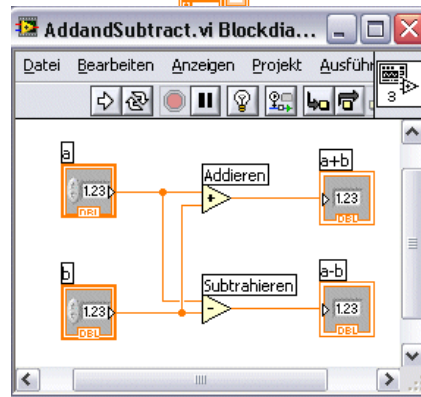
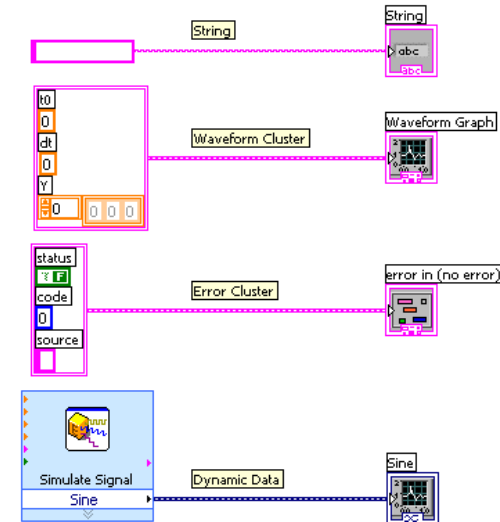
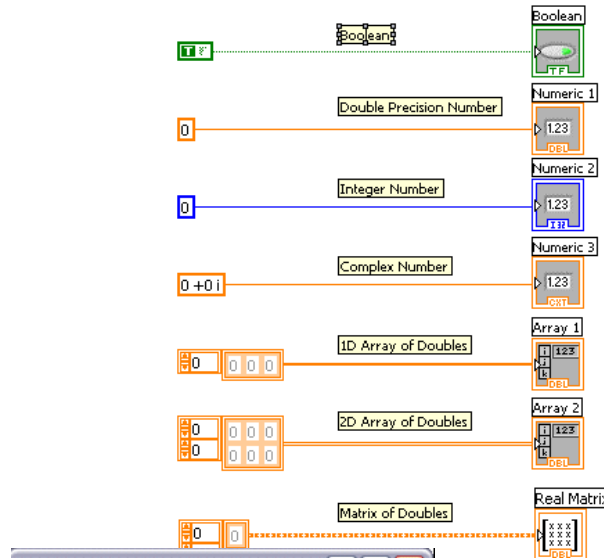
- Anweisungen, (z.B.: Zeitstempel)
- Operatoren,
- Funktionen und
- Subroutinen in textbasierten Programmiersprachen.

z.B.: Additions- und Subtraktionsfunktionen



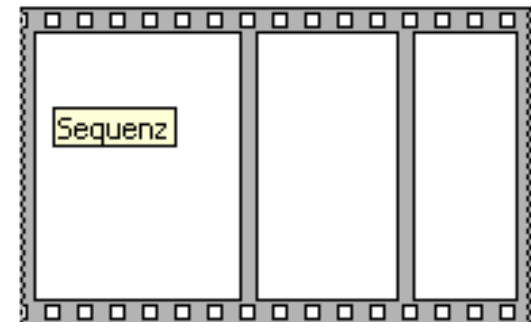
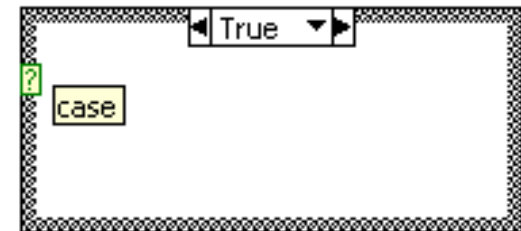
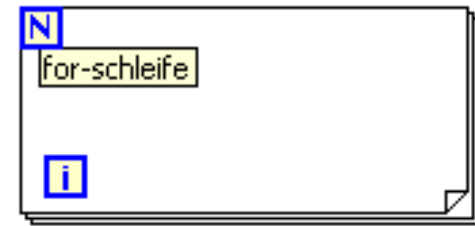
# Verbindungen

- Daten werden über Verbindungen ausgetauscht.
- Jede Verbindung hat eine Datenquelle, die Sie jedoch mit mehreren Datensenken verbinden können!
- Je nach Datentyp haben die Verbindungen unterschiedliche Farben, Formate und Linienstärken.
- Objekte mit inkompatiblen Datentypen



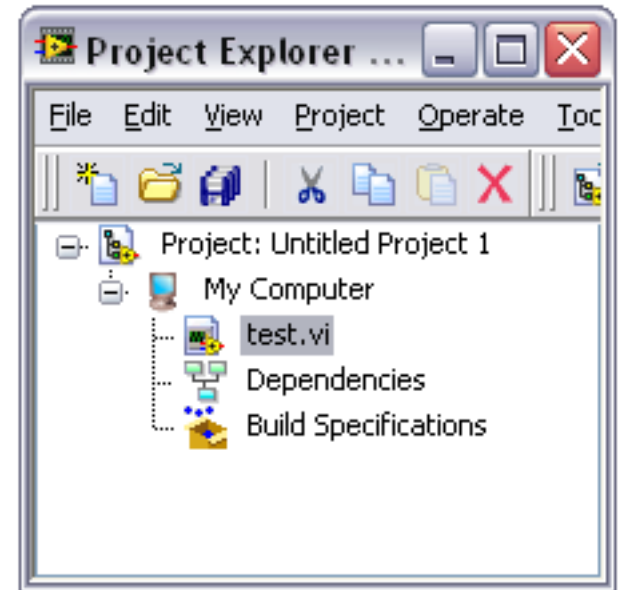
# Strukturen

- Strukturen sind grafische Darstellungen von Schleifen und Case-Anweisungen in textbasierten Programmiersprachen.
- Mit Strukturen werden Blockdiagramm-Abschnitte
  - wiederholt,
  - bedingungsabhängig ausgeführt oder
  - in einer bestimmten Reihenfolge ausgeführt.
- Weitere Informationen zu Strukturen finden Sie in Kapitel 8, *Schleifen und Strukturen*.



# LabVIEW Project

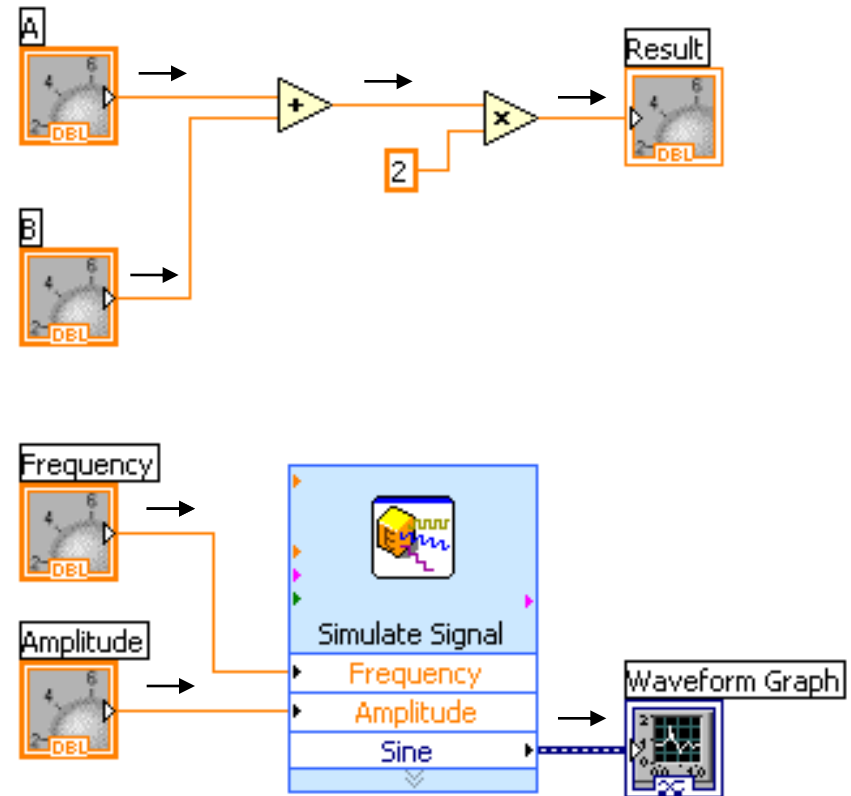
- Gruppieren und organisieren von VIs
- Hardware und I/O Kontrolle
- Erstellen von “libraries” und Stand alone Applikationen
- Steuern von großen LabVIEW Anwendungen
- Ermöglicht die Versionsverfolgung und deren Kontrolle



(LabVIEW»Project»New)

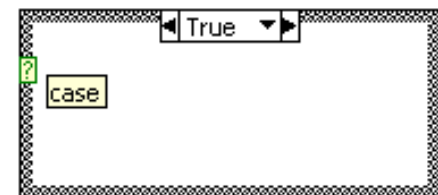
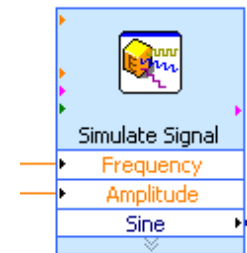
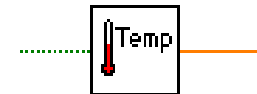
# Datenfluß Programmierung

- Blockdiagramm ausführen
  - Abhängig vom Fluß der Daten
  - Blockdiagramm wird nicht von links nach rechts ausgeführt
- Knoten werden ausgeführt wenn die Daten an allen Eingängen anliegen
- Daten liegen am Ausgang des Knotens (Ergebnis) erst nach der Knotenbearbeitung am Knoten an



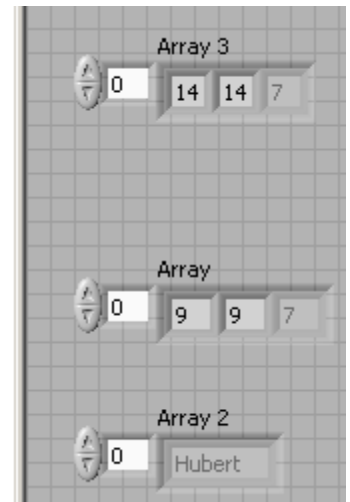
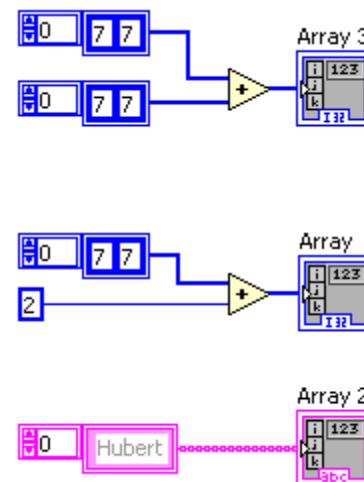
# LabVIEW umfasst die folgenden Arten von Knoten:

- **Funktionen** -Ausführungselemente, die mit einem Operator, einer Funktion oder einer Anweisung vergleichbar sind.
- **SubVIs** -VIs, die in einem Blockdiagramm von einem anderen VI verwendet werden, vergleichbar mit Unterprogrammen.  
Weitere Informationen Kapitels 7,
- **Express-VIs** -Auf Standardaufgaben im Bereich der Messtechnik zugeschnittene SubVIs. Die Konfiguration von Express-VIs wird in einem speziellen Dialogfeld durchgeführt.
- **Strukturen** -Ausführungssteuerelemente, wie zum Beispiel For-Schleifen, While-Schleifen, Case-Strukturen, flache und gestapelte Sequenzstrukturen, zeitgesteuerte Strukturen und Ereignisstrukturen.



# Polymorphe VIs und Funktionen

- Polymorphe VIs und Funktionen können an die Eingabewerte unterschiedlicher Datentypen angepasst werden.
  - es können entweder einige oder alle Eingänge polymorph sein.
  - einige für Zahlen oder booleschen Werte.
  - einige sind Zahlen oder Strings zulässig.
  - einige für skalare Zahlen,
  - einige auch für Arrays und Cluster mit numerischen Werten.

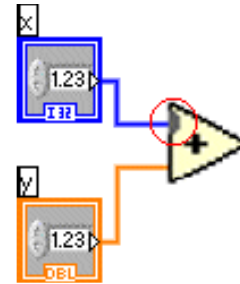




# Typumwandlungspunkte

- zwei unterschiedliche Datentypen  
(unterschiedliche Darstellung)
- Der Punkt bedeutet, dass der an den Knoten weitergeleitete Wert von LabVIEW in eine andere Darstellung konvertiert wurde.  
So sind zum Beispiel für die Funktion "Addieren" Fließkommazahlen mit doppelter Genauigkeit als Eingangswerte zulässig.
- Wenn ein VI Typumwandlungspunkte enthält, kann es langsamer werden und mehr Speicherplatz benötigen.

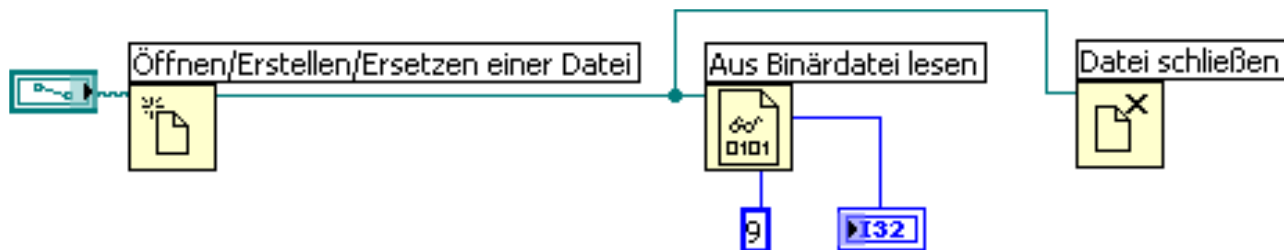
Nach Möglichkeit sollten Sie Datentypen durchgehend verwenden!!



Bis hier her am

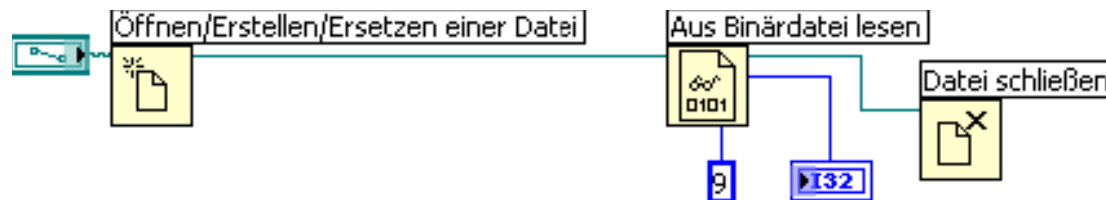
# Datenfluss im Blockdiagramm

- Fehlende Datenabhängigkeit
  - Gehen Sie nicht von einer Ausführungsreihenfolge von links nach rechts oder von oben nach unten aus, wenn es keine Datenabhängigkeit gibt.
- Legen Sie die Ausführungsreihenfolge der Objekte gegebenenfalls selbst fest, indem Sie den Datenfluss durch die entsprechenden Verbindungen vorgeben.
  - z.B.: Im folgenden Blockdiagramm besteht keine Abhängigkeit zwischen den Funktionen “Aus Binärdatei lesen” und “Datei schließen”,



# Datenfluss im Blockdiagramm

- Im folgenden Blockdiagramm wird eine Abhängigkeit erzeugt, indem ein Ausgang der Funktion “Aus Binärdatei lesen” mit der Funktion “Datei schließen” verbunden wird.
- Die Funktion “Datei schließen” wird erst ausgeführt, wenn die Ausgabe der Funktion “Aus Binärdatei lesen” empfangen wird.

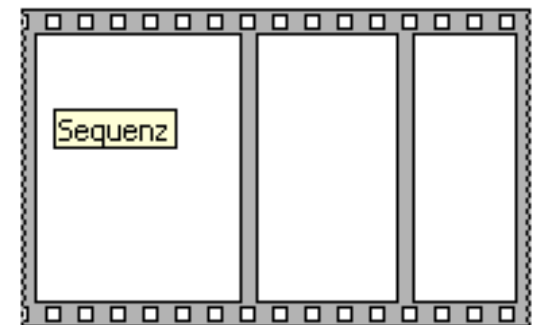
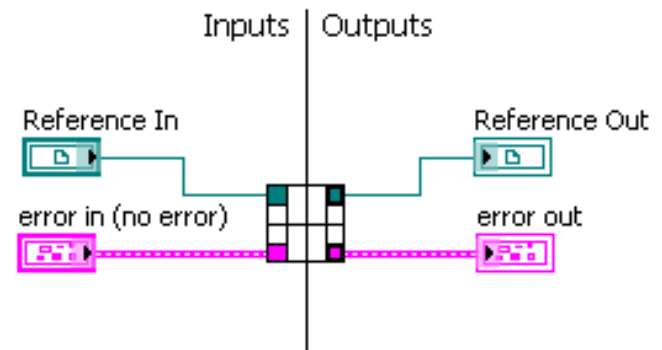


# Datenfluss im Blockdiagramm

- Durchflussparameter, gewöhnlich eine
  - Refnum oder ein
  - Fehler-Cluster, diese geben denselben Wert aus wie der entsprechende Eingangsparameter.

Diese Parameter sollten verwendet werden, um eine Ausführungsreihenfolge festzulegen, wenn keine Datenabhängigkeit vorliegt.

- Ansonsten müsste mit Sequenzstrukturen sichergestellt werden, dass Datenoperationen in der gewünschten Reihenfolge stattfinden.



# Speicherverwaltung

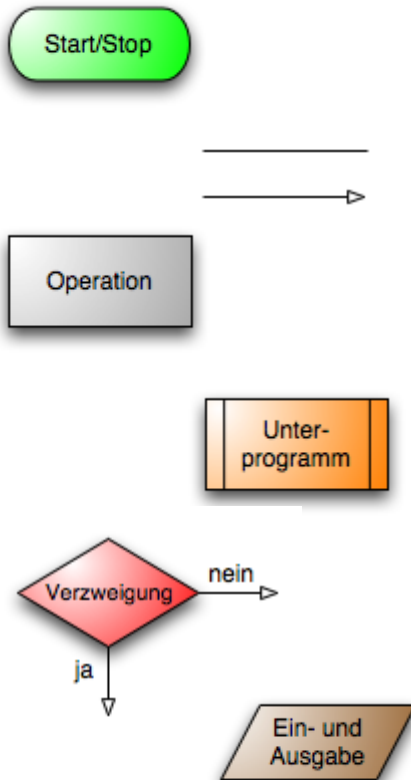
- VIs und Funktionen, die Daten generieren, reservieren auch automatisch den Speicher für die Daten.
- Wenn das VI oder die Funktion die Daten nicht mehr verarbeitet, hebt LabVIEW die Speicherzuweisung auf.
- Wenn Sie einem Array oder String neue Daten hinzufügen, wird zur Verwaltung der Daten erneut Speicher reserviert.

# Entwerfen des Blockdiagramms

- Gehen Sie beim Aufbau des Blockdiagramms immer von links nach rechts und von oben nach unten vor. (Struktur)
- VI s eine Bildschirmseite groß.
- Abschnitt des Blockdiagramms als logische Komponenten zusammen -> SubVIs bilden
- Verwenden Sie die Fehlerbehandlungs-VIs.
- Verbindungssegmente nicht überdecken.
- keine Objekte über Verbindungen.
- Verwenden Sie Kommentare
- Dokumentation

# Flussdiagramm

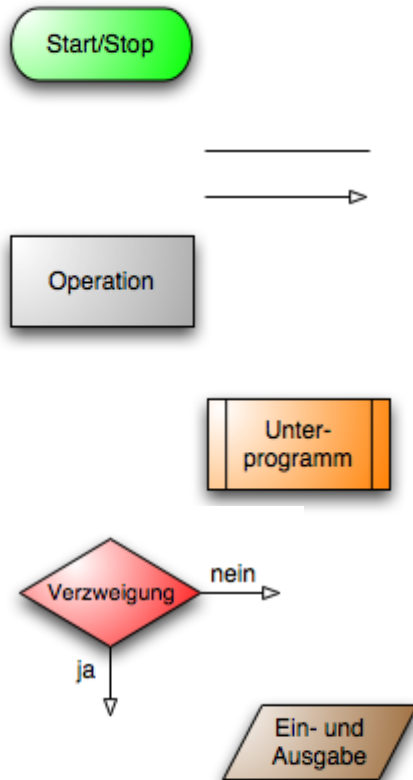
Ein F. ist eine graphische Darstellung zur Umsetzung eines Algorithmus in einem Programm und beschreibt die Folge von Operationen zur Lösung einer Aufgabe. DIN 66001



- Oval: Start, Ende, weitere Grenzpunkte
- Pfeil, Linie: Verbindung zum nächstfolgenden Element
- Rechteck: Operation (Eine Tätigkeit beschreibt einen bestimmten Vorgang, der innerhalb des jeweiligen Prozesses durchgeführt wird.)
- Rechteck mit doppelten, vertikalen Linien: Ein Unterprogramm das eine Teilaufgabe erarbeitet!
- Raute: Verzweigung Eine Entscheidung hat einen Eingang (oben) und mehrere Ausgänge (links, rechts und unten; Ja/Nein, usw.).
- Parallelogramm: Ein- und Ausgabe z.B.: Abfrage des Users; Plot



# Flussdiagramm

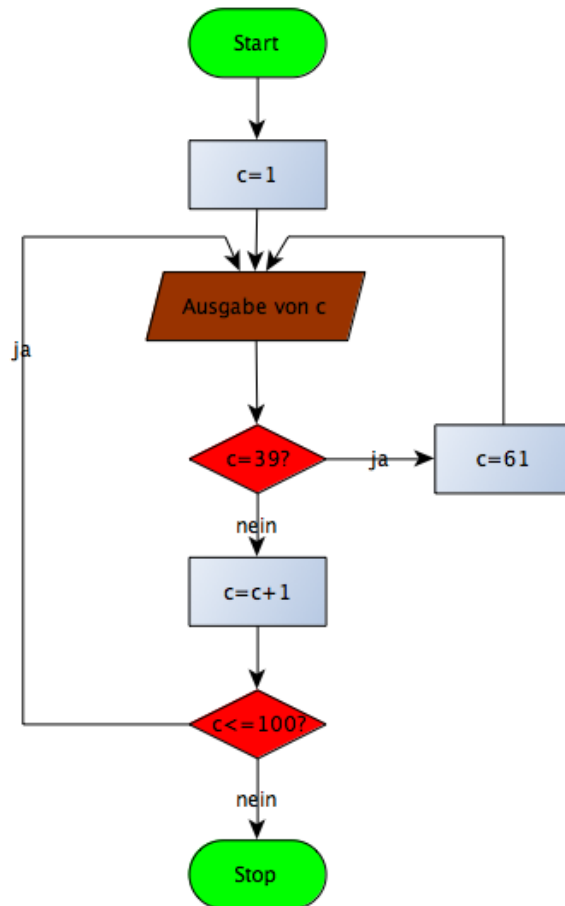


## Aufgabe Counter

Schreiben Sie ein Programm das c bis 101 hochzählt.  
Wenn c bei 39 angekommen ist – soll c auf den neuen Wert von 61 springen (geändert werden) und bis auf 101 weiterzählen.  
Zur Kontrolle soll bei c = 39 eine Lampe aktiviert werden.

c ist nicht der index-Zähler – c ist nur eine Variable

# Flussdiagramm



z.B.: Die nebenstehende Abbildung zeigt eine einfache Zählschleife von 1 bis 101.

Die Variable  $c$  wird vor Beginn der Schleife auf ihren Startwert  $c=1$  gesetzt.

Danach wird die erste Anweisung der Schleife, das Ausgeben der Variable  $c$ , ausgeführt.

Die nachfolgende zweite Anweisung ist eine einseitige Auswahl, die prüft, ob  $c$  den Wert 39 besitzt.

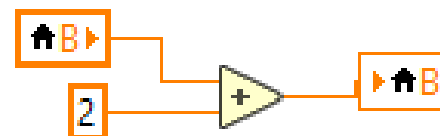
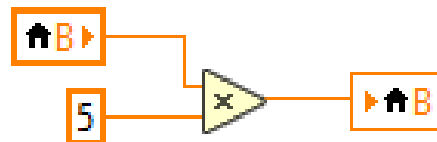
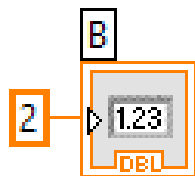
Falls dies der Fall ist, wird  $c$  auf den Wert 61 gesetzt und die Schleife beginnt mit dem nächsten Durchlauf.

Falls  $c$  nicht 39 ist, wird  $c$  in der nachfolgenden Anweisung um eins erhöht und anschließend geprüft, ob die Schleifenabbruchbedingung  $c \leq 100$  erreicht ist.

Falls nicht, erfolgt ein nochmaliger Schleifendurchlauf. Ausgegeben werden alle ganzen Zahlen von 1 bis 39 sowie 61 bis 100 (jeweils einschließlich).

# Race Conditions

Wer ist schneller?  
Was kommt raus?  
Kommt immer das gleiche raus?



# FGV

Am 16 race co

# Ampelprogramm



# Häufige Fehlerursachen

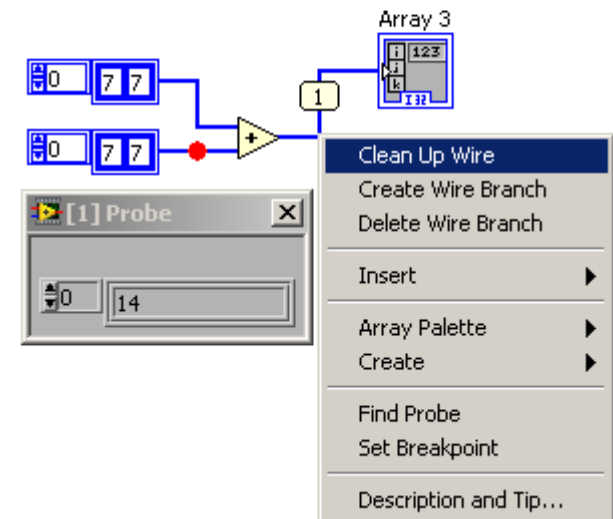
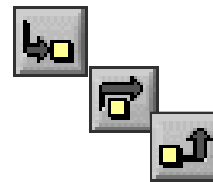
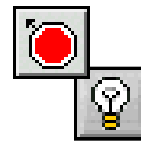


Die folgende Liste enthält häufige Ursachen, warum ein VI als nicht ausführbar gekennzeichnet wird, während Sie es bearbeiten:

- alle SubVIs, Funktionen und Strukturen → richtige Datentypen
- Verbindungssegment an einem Ende offen
- ein obligatorischer Blockdiagrammanschluss wurde nicht verbunden.
- ein SubVI ist nicht ausführbar oder
- das Anschlussfeld des SubVIs wurde zwischenzeitlich bearbeitet

# Fehlersuchmethoden

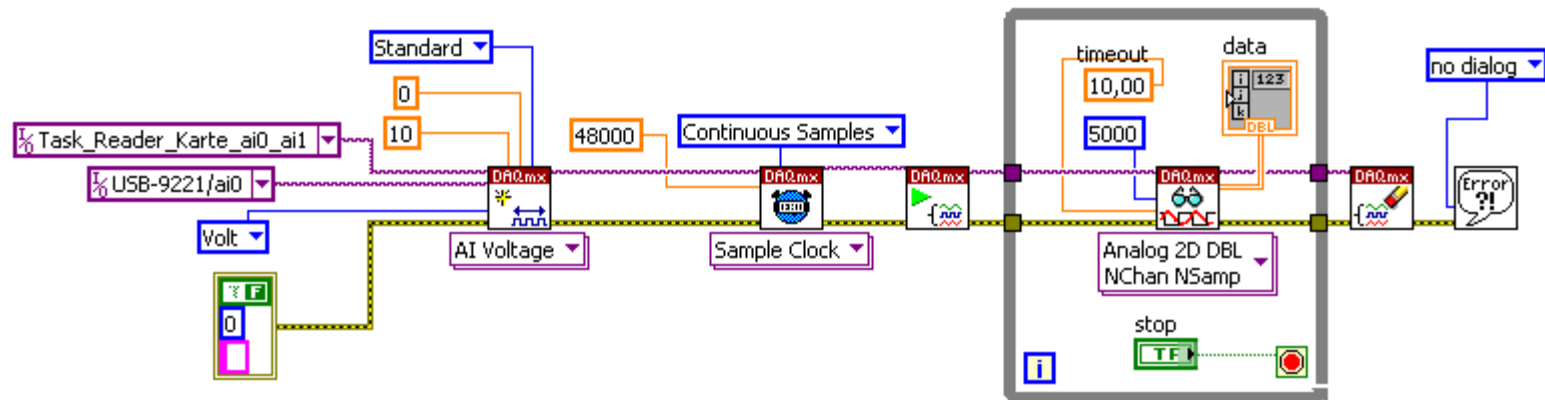
- Highlight-Funktion
- Einzelschrittausführung
- Haltepunkt-Werkzeug (Breakpoint)
- Sonden-Werkzeug (Probe)





# Fehlerprüfung

- folgt dem Datenflussmodell.
  - Fehlerein- und -ausgänge aller Knoten vom Anfang bis zum Ende des VIs.
  - am Ende des VIs ein Fehlerhandler-VI ein, um zu ermitteln, ob das VI fehlerlos ausgeführt werden konnte.
- Fehler-Cluster sind Durchflussparameter.
- Fehler-Cluster
  - **Status** ist ein boolescher Wert, (TRUE = Fehler)
  - **Code** ist ein 32-Bit-Integer mit Vorzeichen (Fehlerliste)
  - **Quelle** ist ein String (Fehlerstelle)

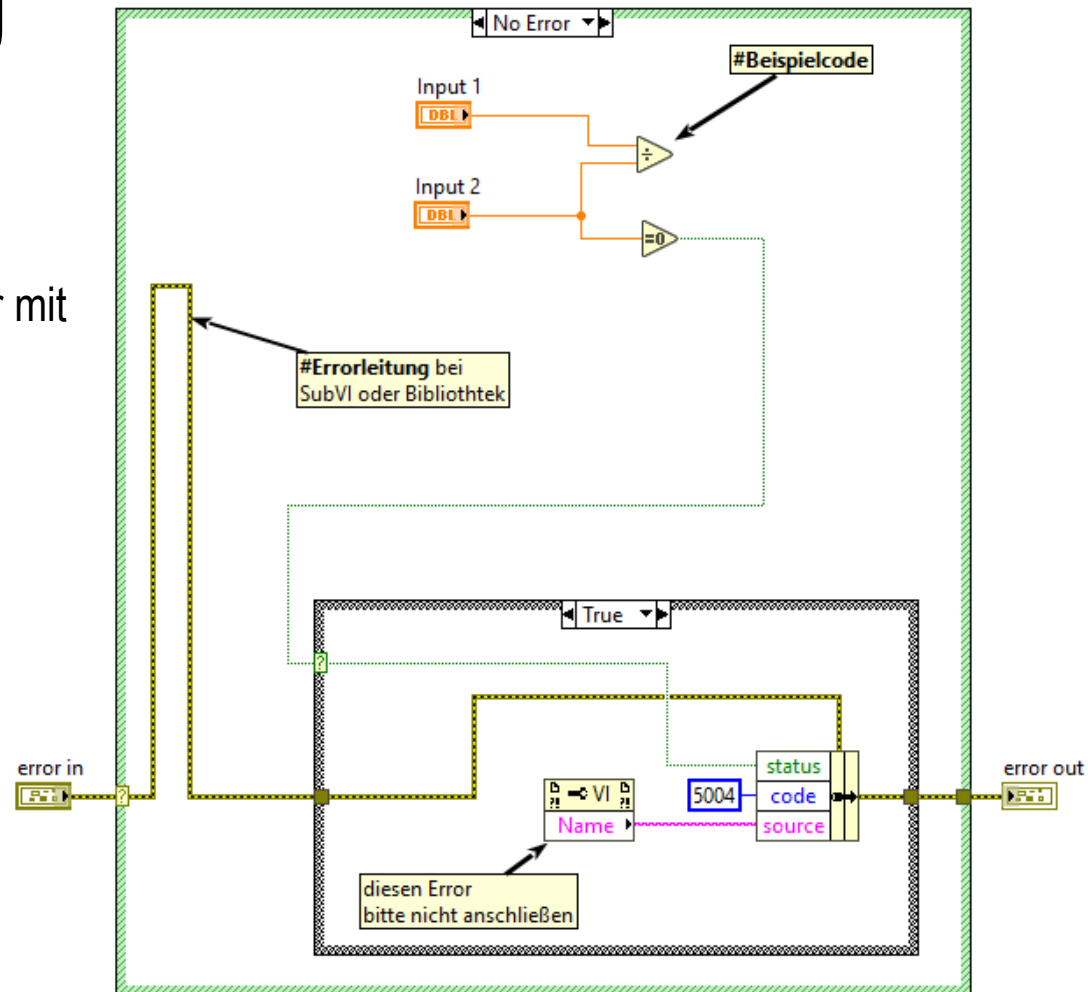


# Fehlerverarbeitung

- mit Case-Strukturen

Selektoranschluss einer Case-Struktur mit einem Fehler-Cluster

- Fehlercase „rot“ und
- kein Fehlercase „grün“.



# Fehlerverarbeitung

## Defining Custom Error Codes

1. Error codes 5000 through 9999 and -8999 through -8000 are reserved for you to define custom error messages.
  2. Complete the following steps to define custom error codes.
  3. Place the [General Error Handler](#) VI on the block diagram.  
Right-click the **user-defined codes** input and select **Create»Constant** from the shortcut menu. An array appears.
  4. Double-click the numeric constant and enter a number within the range of 5000 to 9999 or -8999 to -8000 in the array. For example, enter 5008.
  5. Right-click the **user-defined descriptions** input and select **Create»Constant** from the shortcut menu. Another array appears.
  6. Double-click the string constant and enter a description in the user-defined description array. For example, enter Ignore this message.
  7. Right-click the **error in** input and select **Create»Constant** from the shortcut menu. A cluster appears that contains a Boolean constant, a numeric constant, and an array.
  8. Use the Operating tool to set the Boolean control to TRUE.
  9. Double-click the numeric constant and enter the same number that appears in the **user-defined codes** constant.
  10. **Note** As an alternative to steps 6 through 8, you can wire a numeric constant with a value of 5008 to the [Error Cluster From Error Code](#) VI.
  11. Run the General Error Handler VI. The message box on the General Error Handler VI front panel displays the customized error code and description.
- You also can define custom error codes by [creating an XML-based text file](#) and adding the error codes and messages to the text file.

Bis  
2.6

# Fehlerverarbeitung

- Labview Template mit E
- [http://zone.ni.com/reference/de-XX/help/371361R-0113/lvhowto/creating\\_vi\\_templates/](http://zone.ni.com/reference/de-XX/help/371361R-0113/lvhowto/creating_vi_templates/)
- Demo-Programm mit Errorhandling erstellen

# Kapitel 7 Erstellen von VIs und SubVIs

Ein VI kann entweder als Benutzeroberfläche HMI dienen oder für häufig benötigte Operationen genutzt werden.

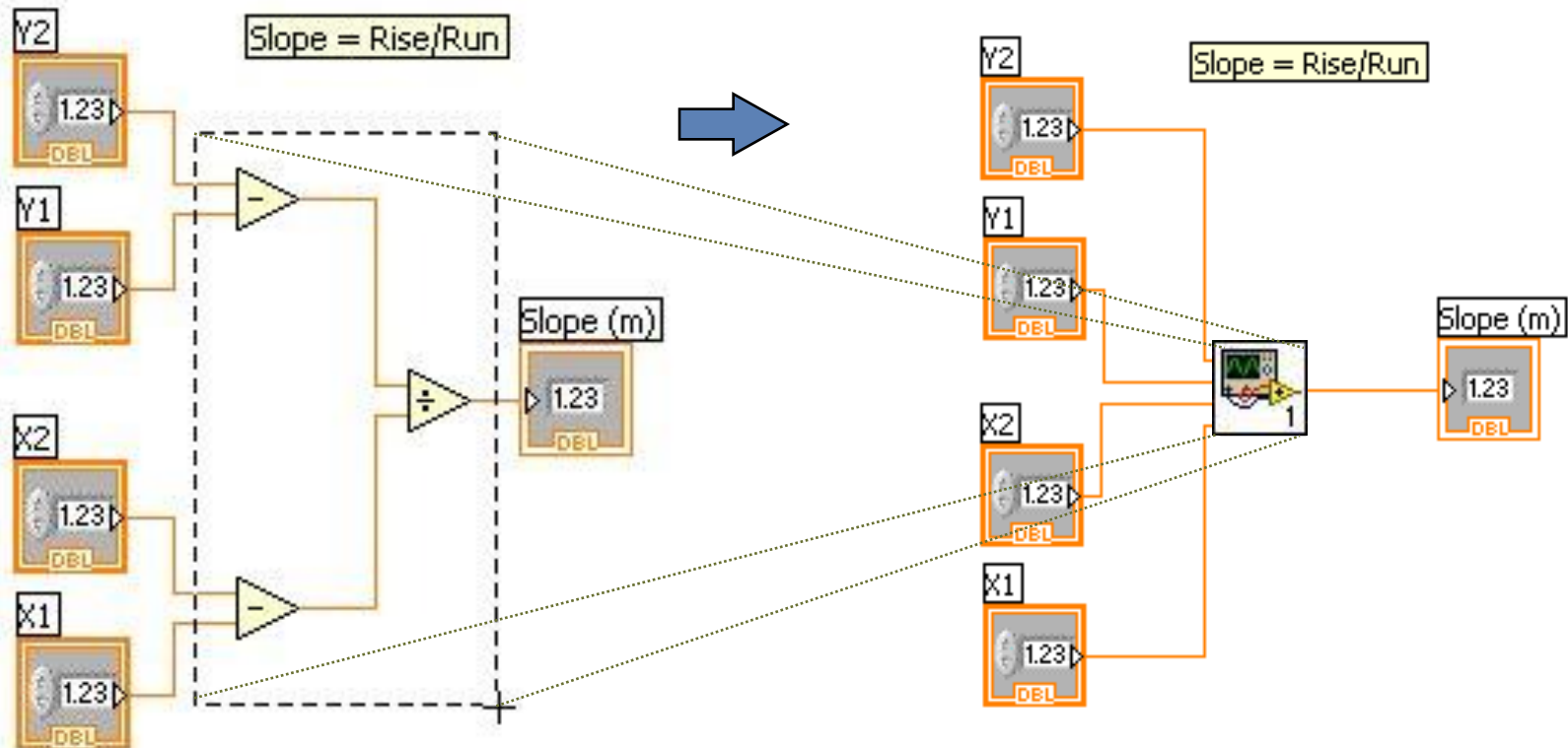
- passendes Beispiel-VI aus den Examples .
- Verwendung als SubVI oder HauptVI.
  - SubVI-Knoten entspricht einem Subroutinenaufruf in textbasierten Programmiersprachen.
- Symbol
- Anschlußfeld

# Erstellen von VIs und SubVIs

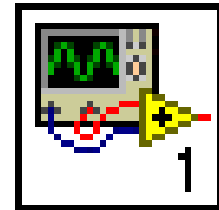
- Einfügen von Bedien- und Anzeigeelemente
- Bedienelemente links
- Anzeigeelemente rechts
- Cluster Fehlereingang
- Cluster Fehlerausgang
- Erstellen von SubVIs aus VI-Abschnitten

# Erstellen von SubVIs aus VI-Abschnitten

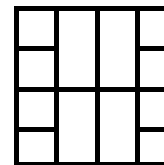
- Enclose area to be converted into a subVI.
- Select **Edit»Create SubVI** from the Edit Menu.



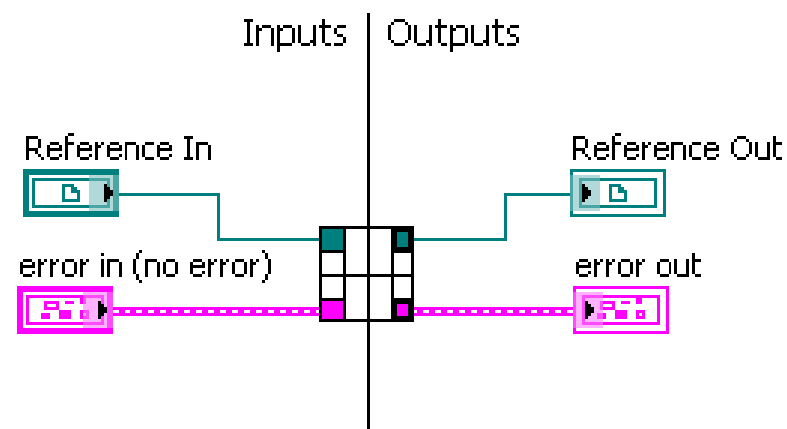
# Icon and Connector Pane



- Use this connector pane layout as a standard



- Top terminals are usually reserved for references, such as a file reference
- Bottom terminals are usually reserved for error clusters





# Benennen und Speichern von VIs

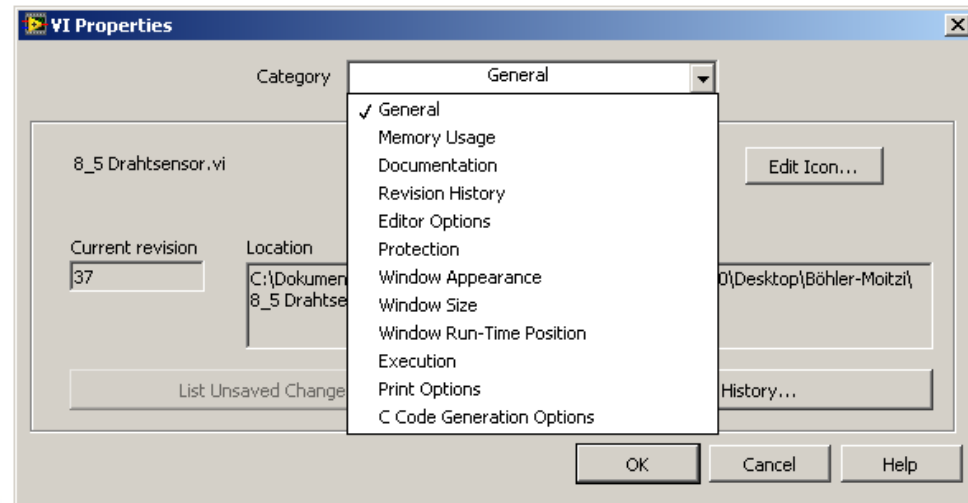
- Verwenden Sie aussagekräftige, beschreibende Namen, wie
  - Temperaturüberwachung.vi und
  - Seriell lesen und
  - Schreiben.vi

Vereinfachen die Wiedererkennung eines VIs und geben Auskunft über seinen Verwendungszweck.

- Kontexthilfe
- Speichern für die Vorgängerversion von LabVIEW VIs können für eine ältere Version von LabVIEW gespeichert werden.
  - Wählen Sie Datei»Für vorige Version speichern, um VIs für Vorgängerversionen von LabVIEW zu speichern.

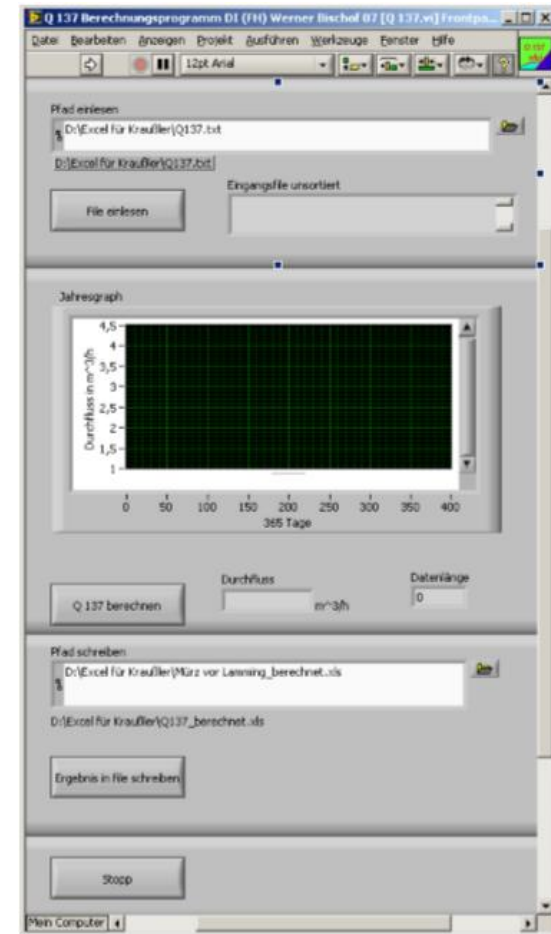
# Individuelle Anpassung von VIs

- Allgemein: aktuellen Pfad an, die Versionsnummer, die Versionshistorie
- Dokumentation: Beschreibung des VIs; Verknüpfung zu einer Hilfedatei.
- Sicherheit: VI sperren oder mit einem Kennwort schützen.
- Fenstererscheinungsbild: Erscheinungsbild; Fenstertitel und -stil ändern.
- Fenstergröße
- Ausführung:  
Optionen zur Ausführung eines VIs.
- Editor-Optionen:  
Gittergröße; Darstellungsart



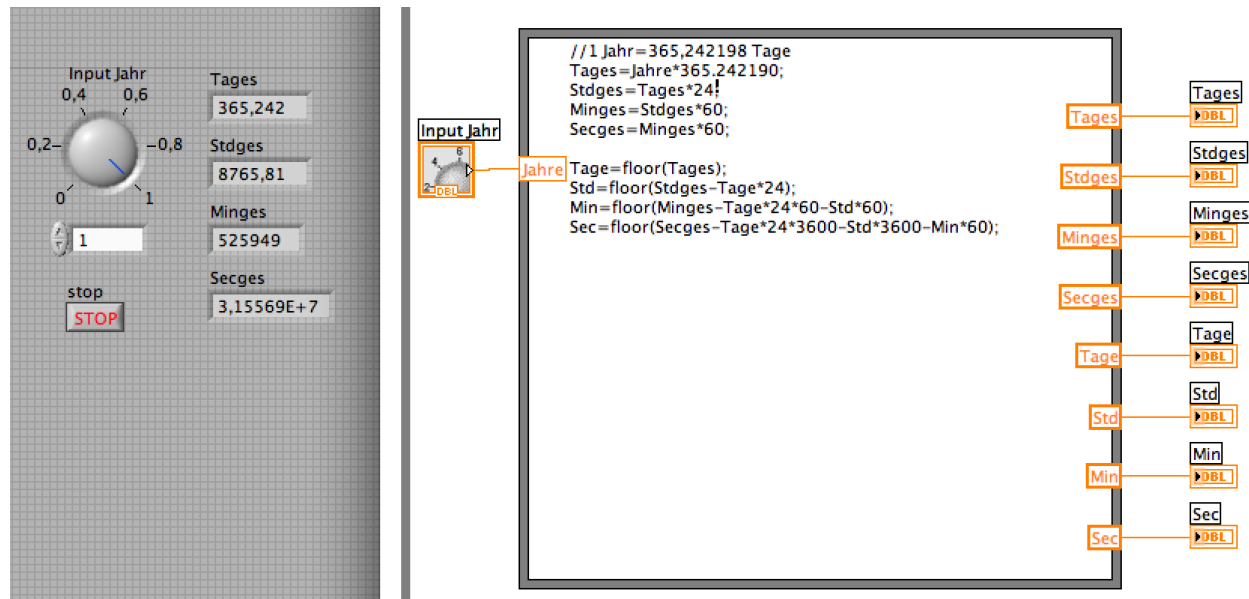
# Exe und Setup erstellen

- Q137 Event Handling als Basis



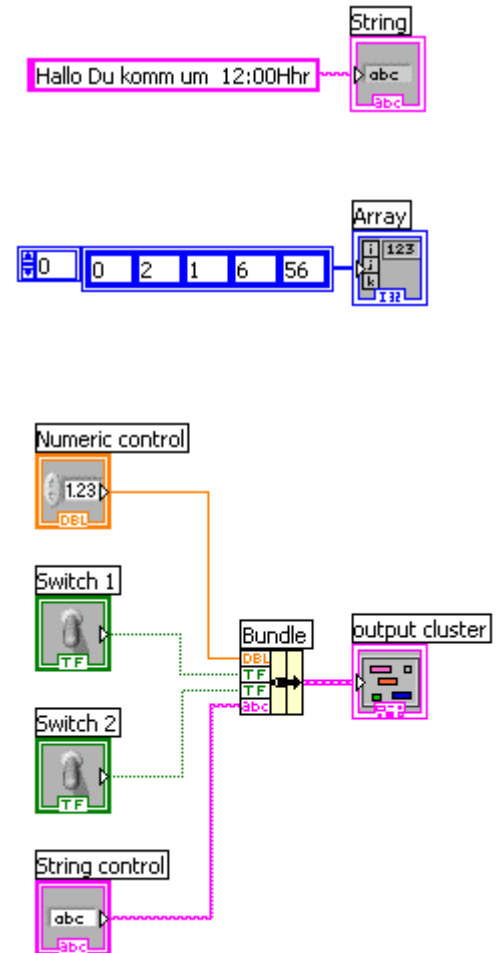
# Mathematik mit dem MathScript-Knoten

- Textbasierte Implementierung von Gleichungen und Algorithmen
- Variablen am Knotenrand
- M-File-Skriptsprache kompatibel
- mit Semikolon, um die sofortige Ausgabe zu deaktivieren



# Kapitel 9 Gruppieren von Daten mit Strings, Arrays und Clustern

- Mit Strings werden ASCII-Zeichen gruppiert.
- Mit Arrays werden Werte des gleichen Typs zusammengefasst.
- Cluster bieten Ihnen die Möglichkeit, Datenelemente unterschiedlichen Typs zu gruppieren.



# String

- darstellbare und nicht
- darstellbare ASCII-Zeichen (Druckersteuerung,...)
- Strings sind plattformunabhängig!!!!

Sie werden unter anderem in folgenden Anwendungsbereichen eingesetzt:

- Textmeldungen.
- numerischer Werte an Instrumente
- Speichern numerischen Daten auf Datenträger.
- Anweisungen und Aufforderungen für den Benutzer mit Hilfe von Dialogfeldern.

# ASCII Zeichen

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>	86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>	87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>	88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>	89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>	90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>;</b>	91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>	92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>	93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>	94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>	95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>

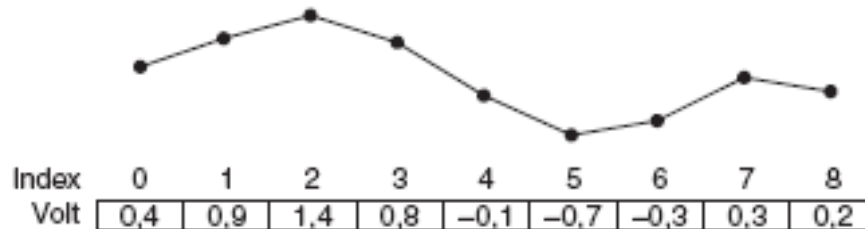
# Array

- Elemente (Werte) und Dimensionen (Länge, Höhe; Tiefe)
  - numerischen,
  - booleschen,
  - Pfad-, String-, Signalverlaufs- und
  - Cluster-Datentypen
- bei ähnlichen Daten
- wiederholende Berechnungen
- ideal zum Speichern von Signalverlaufsdaten oder zum
- Speichern von Daten, die in Schleifen erzeugt werden.



# Indizes

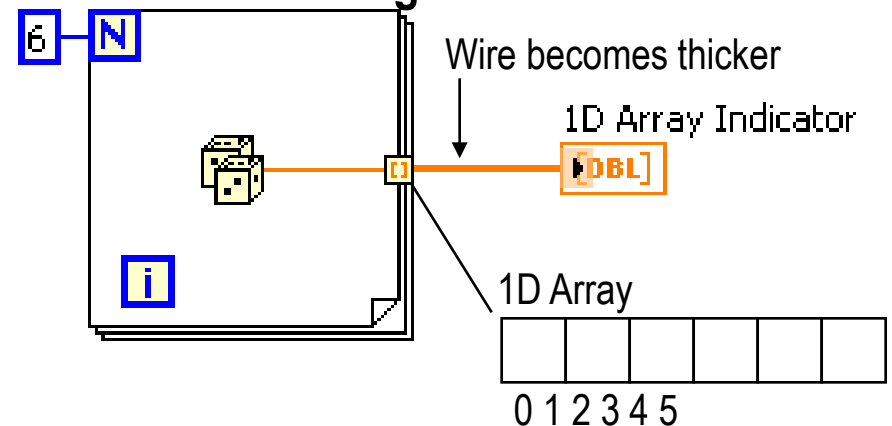
- Elemente finden
  - ein Index pro Dimension (Zeilen, Spalten und Seiten)



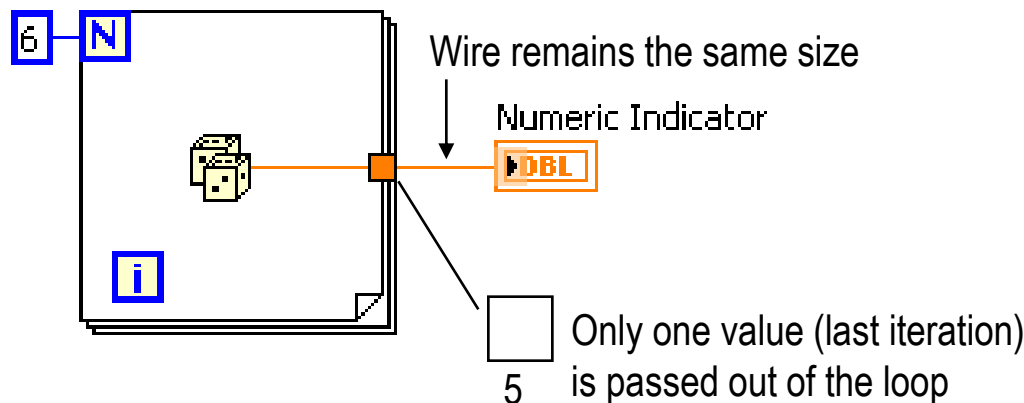
Beispiel für ein Array sind die Abtastwerte eines Signalverlaufs, die in einem numerischen Array abgelegt sind – jede Zelle enthält einen Abtastwert pro Abtastzeitpunkt

# Building Arrays with Loops (Auto-Indexing)

## Auto-Indexing Enabled



## Auto-Indexing Disabled

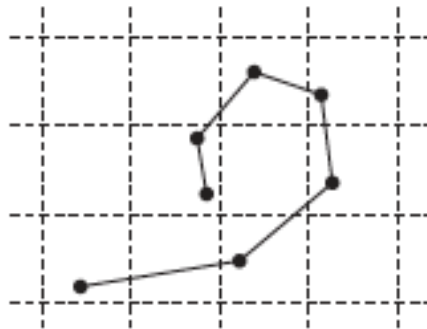


Bis hier !!!

Fehlerbehandl  
kommt später!

# 2D-Array

- Elemente in einem Raster gespeichert. → Spalten- und ein Zeilenindex ( $8 \times 8 = 64$  Elemente).



Index	0	1	2	3	4	5	6
x-Koordinate	0,4	2,2	3,3	3,2	2,4	1,8	1,9

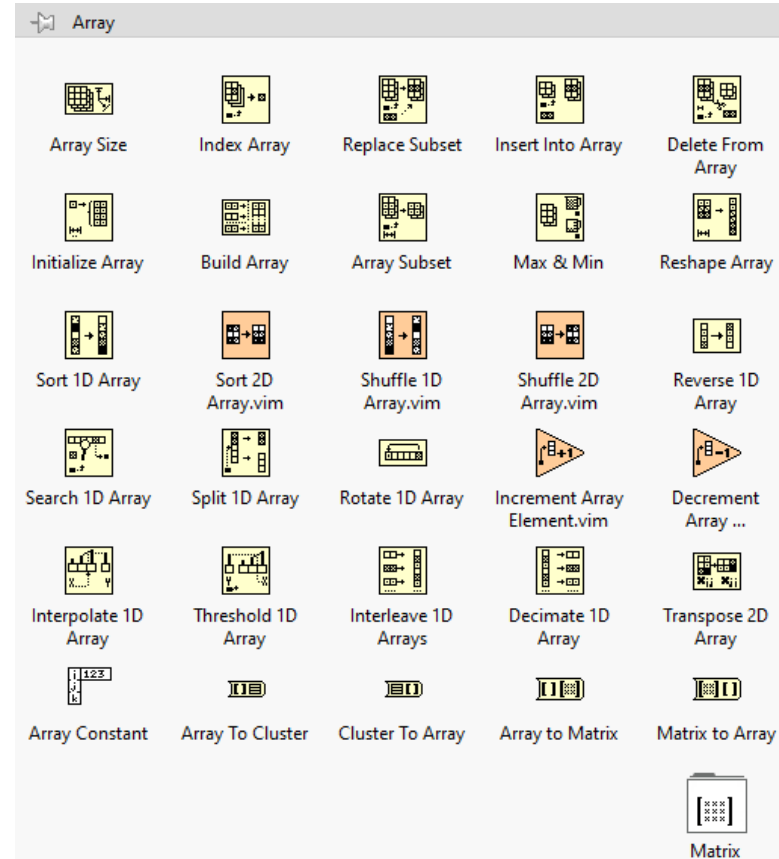
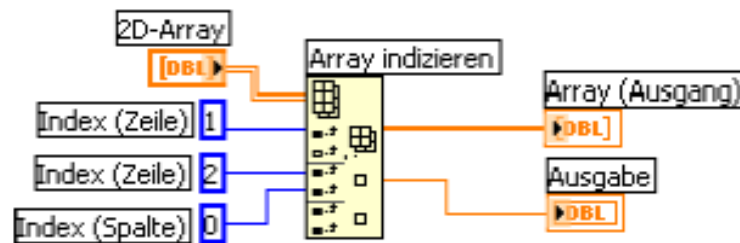
Index	0	1	2	3	4	5	6
y-Koordinate	0,2	0,5	1,3	2,3	2,6	1,9	1,2

	Spaltenindex							
	0	1	2	3	4	5	6	7
Zeilenindex	0							
	1							
	2							
	3							
	4							
	5							
	6							
	7							

Schachbrett enthält zum Beispiel acht Spalten und acht Zeilen für insgesamt 64 Positionen. Jede Position kann leer sein oder eine Schachfigur enthalten. Somit ließe sich ein Schachbrett beispielsweise als 2D-Array aus Strings darstellen, wobei jeder String der Name der Figur ist, die die entsprechende Position auf dem Schachbrett einnimmt, bzw. ein leerer String, wenn die Position leer ist.

# Array-Funktionen

- Arrays erstellen und bearbeiten
  - Extrahieren
  - Einfügen, Löschen oder Ersetzen
  - Teilen eines Arrays.



# Cluster

- Datenelemente unterschiedlichen Typs gruppieren.
- z.B.: LabVIEW-Fehler- Cluster,
- Ein Cluster entspricht einem Datensatz oder einer Struktur in textbasierten Programmiersprachen.
- übersichtlicher und es verringert sich die Anzahl der Anschlussfeldanschlüsse, die für SubVIs benötigt werden. (maximal 28 Anschlüsse)

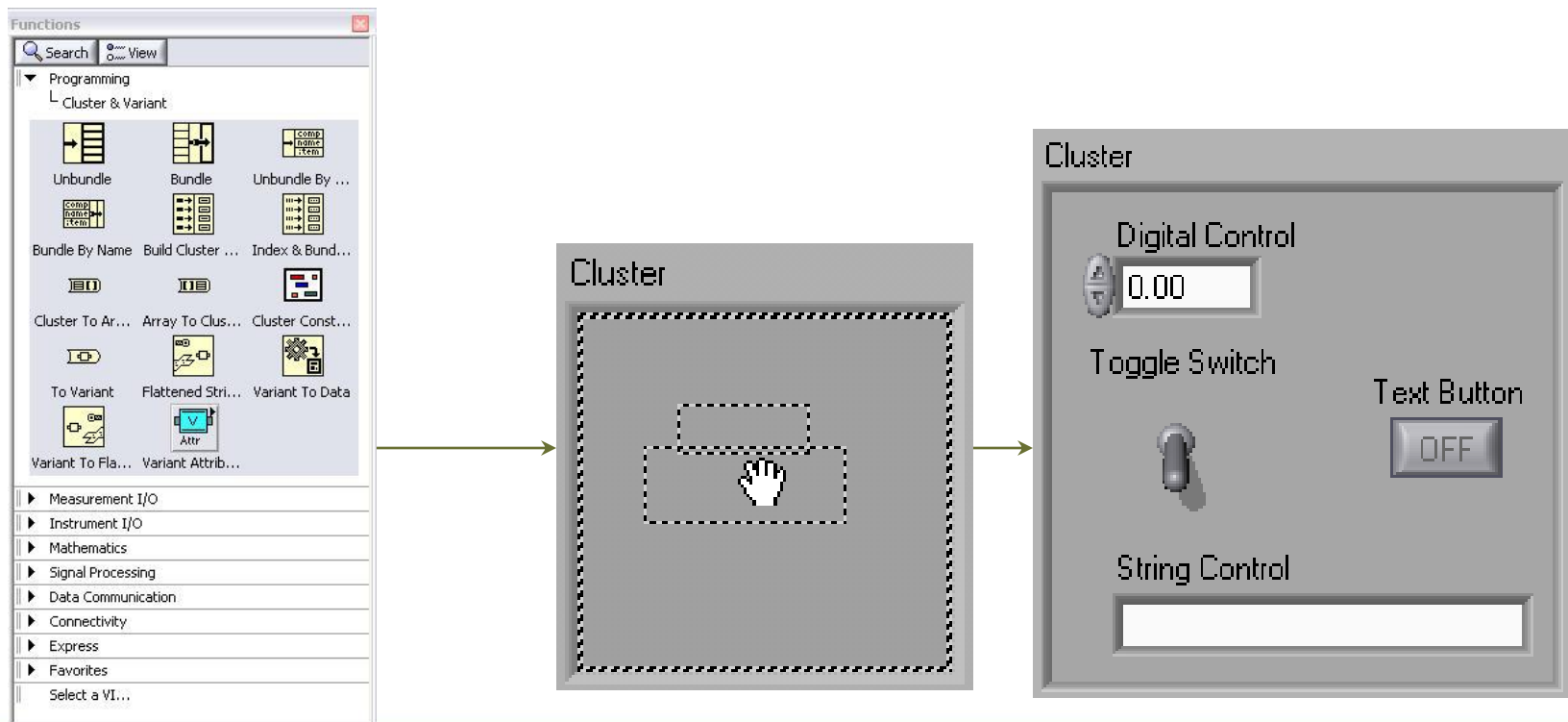
# Cluster

- Datenstruktur, die Daten gruppiert
- Daten können unterschiedlicher Art sein
- Entspricht *struct* in C
- Elemente müssen entweder nur Bedien- oder nur Anzeigeelemente sein
- Entspricht einem Bündel von Einzeldrähten
- Reihenfolge ist wichtig
- z.B.: LabVIEW-Fehler- Cluster,
- übersichtlicher (SubVIs maximal 28 Anschlüsse)

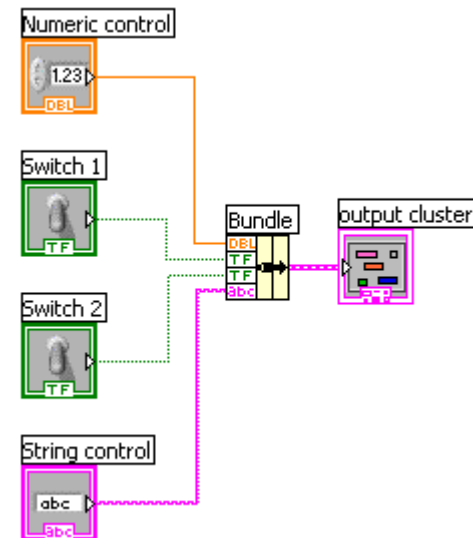
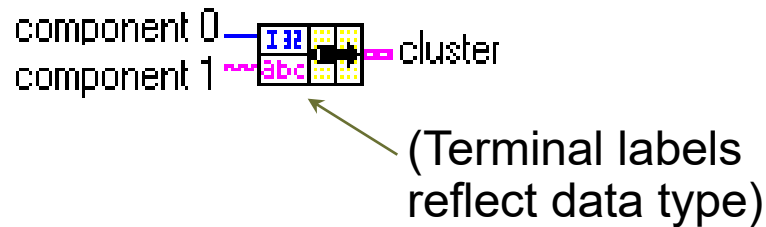


# Erstellen eines Cluster

- Auswählen eines **Cluster**-Containers.
- -> Objekte in den Container platzieren.
- **Bedienelemente» Modern» Array, Matrix & Cluster**

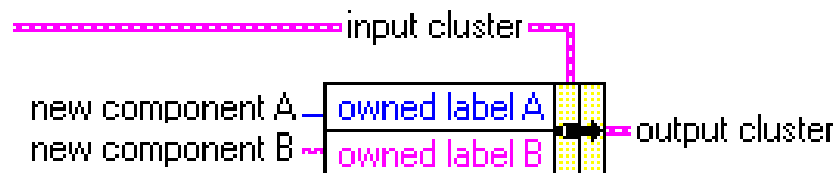


# Cluster Functions

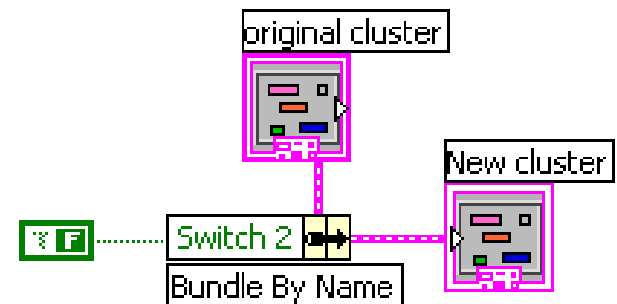


Bis hier  
und fgv.

## Bundle



## Bundle By Name




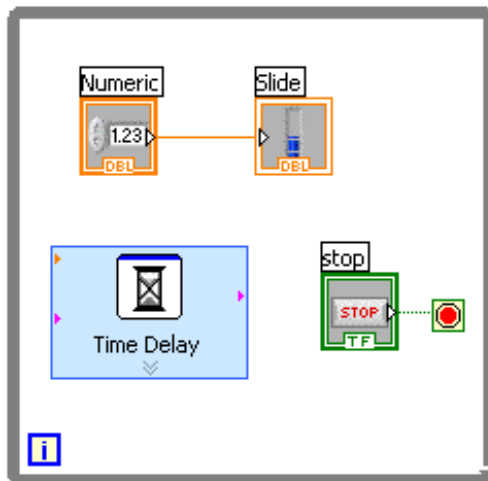


# Kapitel 8 Schleifen und Strukturen

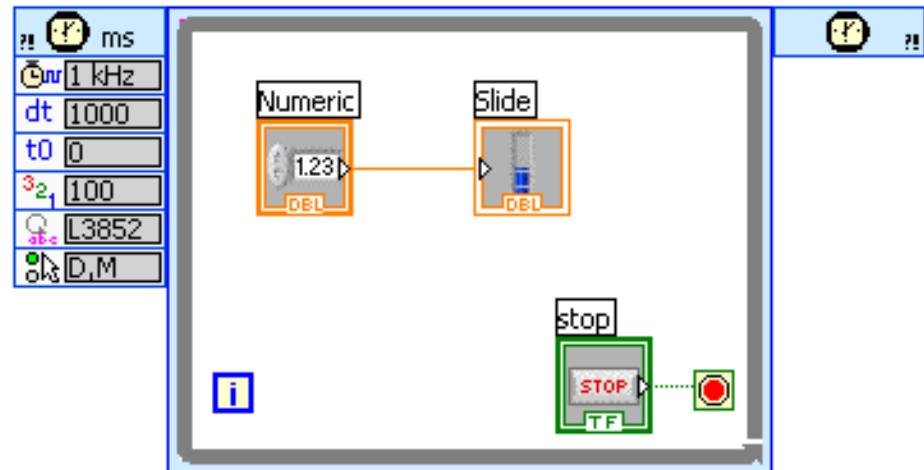
- For-Schleife (N-mal)
- While-Schleife (bis Bedingung erfüllt)
- Case-Struktur (mehrere Rahmen)
- Sequenzstruktur (sequenzieller Reihenfolge)
- Ereignisstruktur (mehrere Rahmen, vom Benutzer bestimmte Ereignisse ausgelöst werden)
- Zeitgesteuerte Strukturen (zeitlichen Vorgaben)

# Wie wird eine Schleife getaktet?

1. Zeitverzögerung bei der Schleife 
2. Konfigurieren Sie das Express-VI Verzögerung (möglich bei For- und While-Schleifen).
3. Zeitgesteuerte Schleifen  
Konfigurieren Sie eine spezielle zeitgesteuerte While-Schleife für das gewünschte *delta t*.



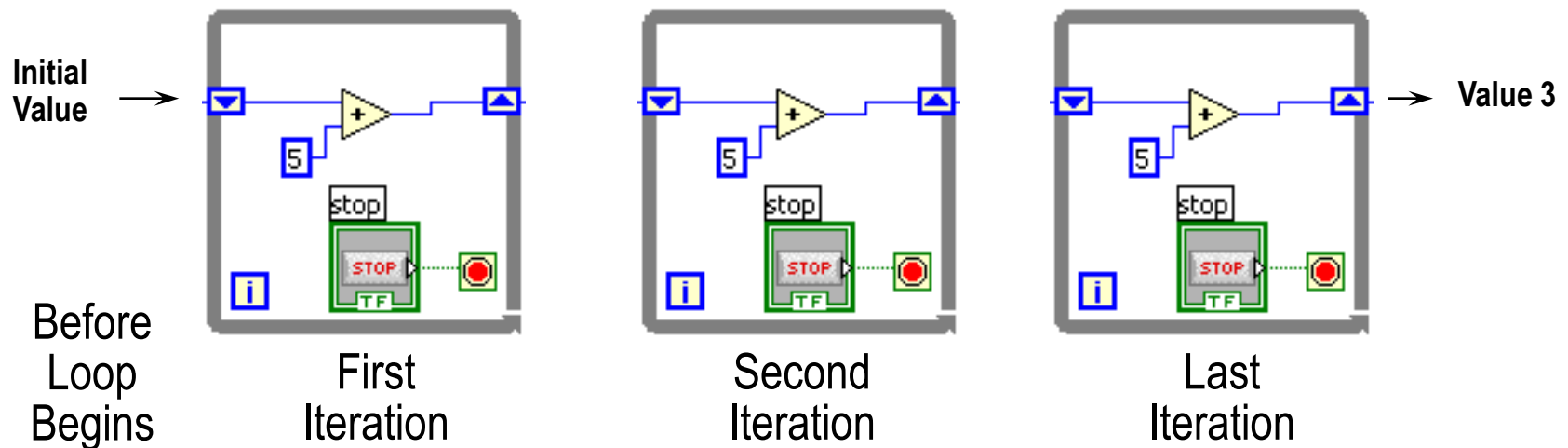
Time Delay



Timed Loop

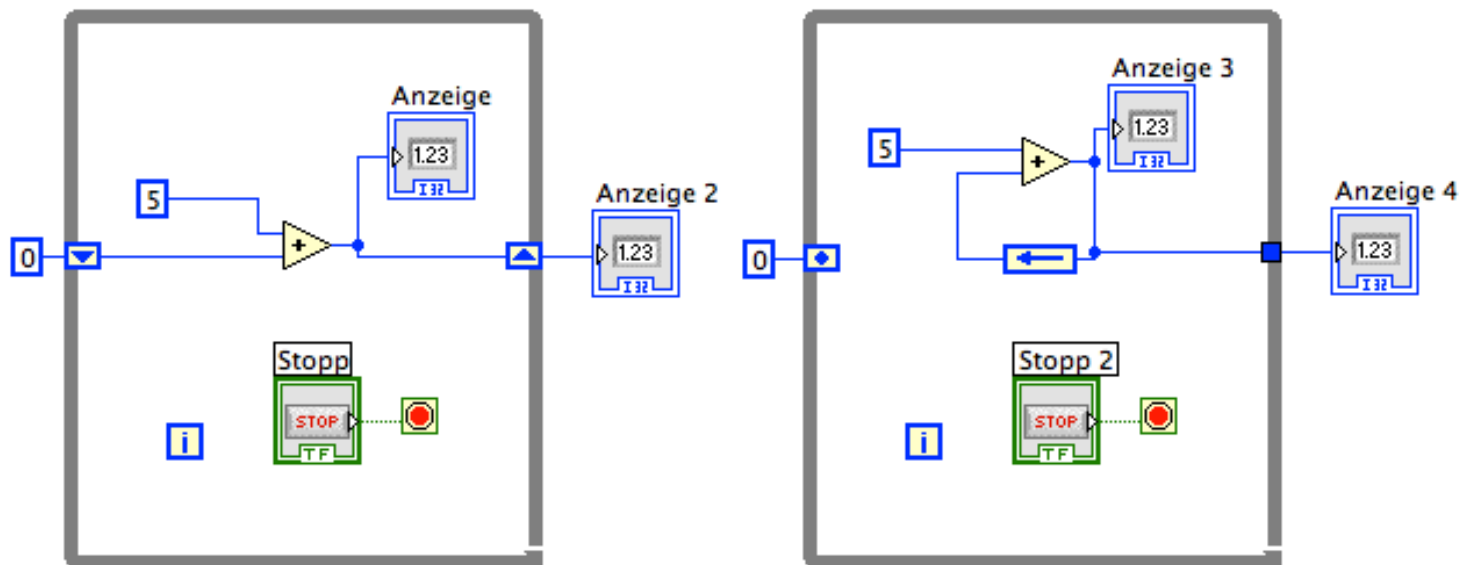
# Schieberegister – Zugriff auf Daten aus vorheriger Schleife

- Rechter Anschluss speichert Daten bei Beendigung eines Schleifendurchlaufs.
- Linker Anschluss liefert gespeicherte Daten zu Beginn des nächsten Schleifendurchlaufs.



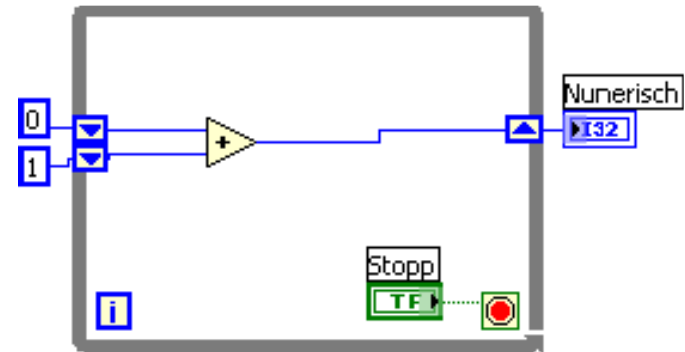
# Schieberegister

Eine Eingabe von 0 würde beim ersten Schleifendurchlauf eine Ausgabe von 5 ergeben, von 10 beim zweiten und 15 beim dritten Schleifendurchlauf. Anders ausgedrückt: Schieberegister werden verwendet, um Werte von einem Schleifendurchlauf zum nächsten beizubehalten. Der Rückkopplungsknoten ist eine weitere Darstellungsweise desselben Konzepts. Beide unten abgebildete Programme verhalten sich gleich.



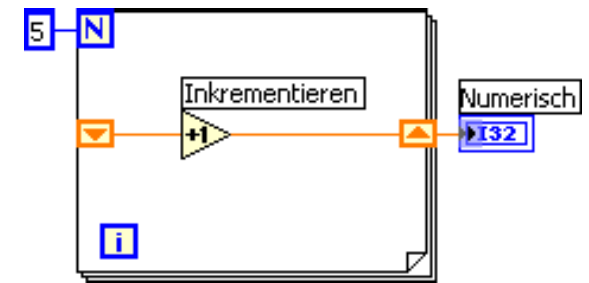
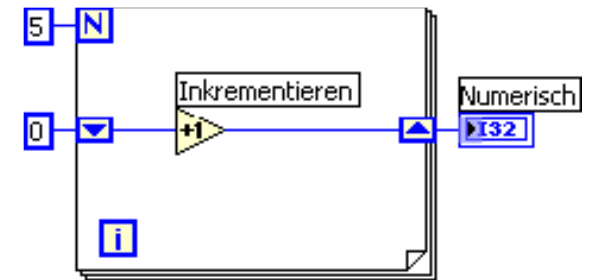
# For- und While-Schleifen

- Eine Schleife kann auch mehrere Schieberegister enthalten.
- Funktionale globale Variablen
- Initialisierung des Schieberegisters
- gestapelte Schieberegister → auf vergangene Schleifendurchläufe zuzugreifen.  
z.B.: Mittelwert



# For-Schleifen

- Die rechte Abbildung zeigt eine For-Schleife, die fünfmal ausgeführt wird. Der Wert des Schieberegisters wird dabei jeweils um eins erhöht. Nach fünf Durchläufen der For-Schleife gibt das Schieberegister den Endwert 5 an das Anzeigeelement weiter und das VI wird abgebrochen. Bei jeder VI-Ausführung beginnt das Schieberegister mit dem Wert 0.
- Wenn Sie das Schieberegister nicht initialisieren, verwendet die Schleife den Wert der letzten Schleifenausführung oder den Standardwert für den entsprechenden Datentyp, wenn die Schleife noch nicht ausgeführt wurde. Auf diese Weise können zum Beispiel Statusinformationen für die nachfolgende Ausführung des VIs gespeichert werden. In der rechten Abbildung sehen Sie ein nicht initialisiertes Schieberegister.

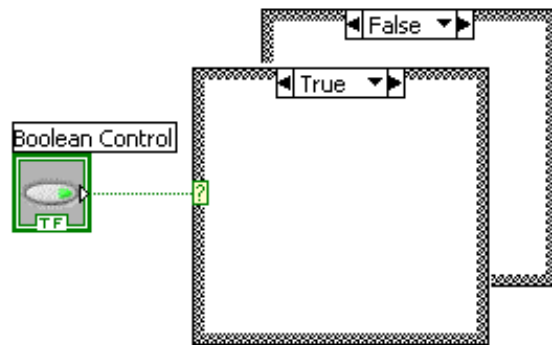


# Case-, Sequenz- und Ereignisstrukturen

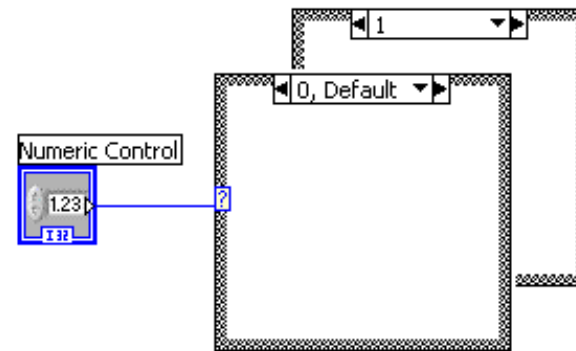
- immer mehrere Unterdiagramme
- Eine Case-Struktur führt je nach Eingabewert einen anderen Blockdiagrammabschnitt (Unterdiagramm) aus.
- Sequenzstruktur werden die darin enthaltenen Unterdiagramme nacheinander ausgeführt.
- Ereignis-Struktur hängt die Ausführung des Unterdiagramms davon ab, wie der Benutzer auf das VI Einfluss nimmt.

# How Do I Make Decisions in LabVIEW?

## 1. Case Structures

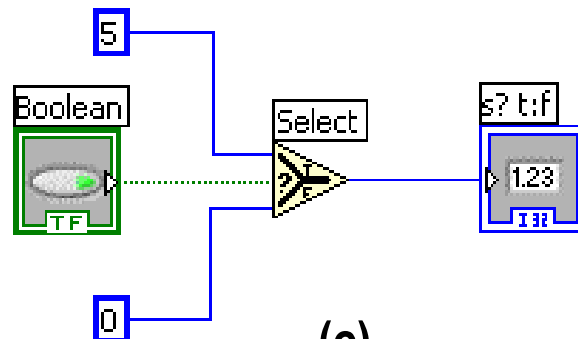


(a)



(b)

## 2. Select

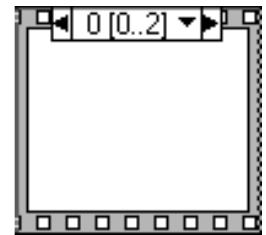
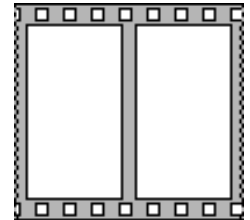


(c)



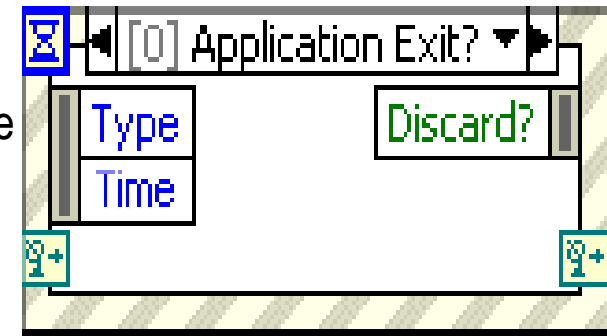
# Sequenzstrukturen

- in sequentieller Reihenfolge
- auch nur aus einem Unterdiagramm
- Ausführungsreihenfolge durch Datenabhängigkeit bestimmt.
- flache und gestapelte



# Ereignisstruktur

- ein oder mehrere Unterdiagramme (Cases),
- wobei immer eines der Unterdiagramme ausgeführt wird, wenn die Struktur ausgeführt wird.
- wartet, bis ein Ereignis auf dem Frontpanel auftritt
- Ereignis kann über:
  - die Benutzeroberfläche,
    - zum Beispiel Mausklicks oder Tastenbetätigungen.
  - eine externe Quelle oder über
  - andere Programmbestandteile hervorgerufen werden.
    - Andere Ereignistypen können programmatisch erzeugt und für die Kommunikation mit verschiedenen Teilen der Applikation verwendet werden.

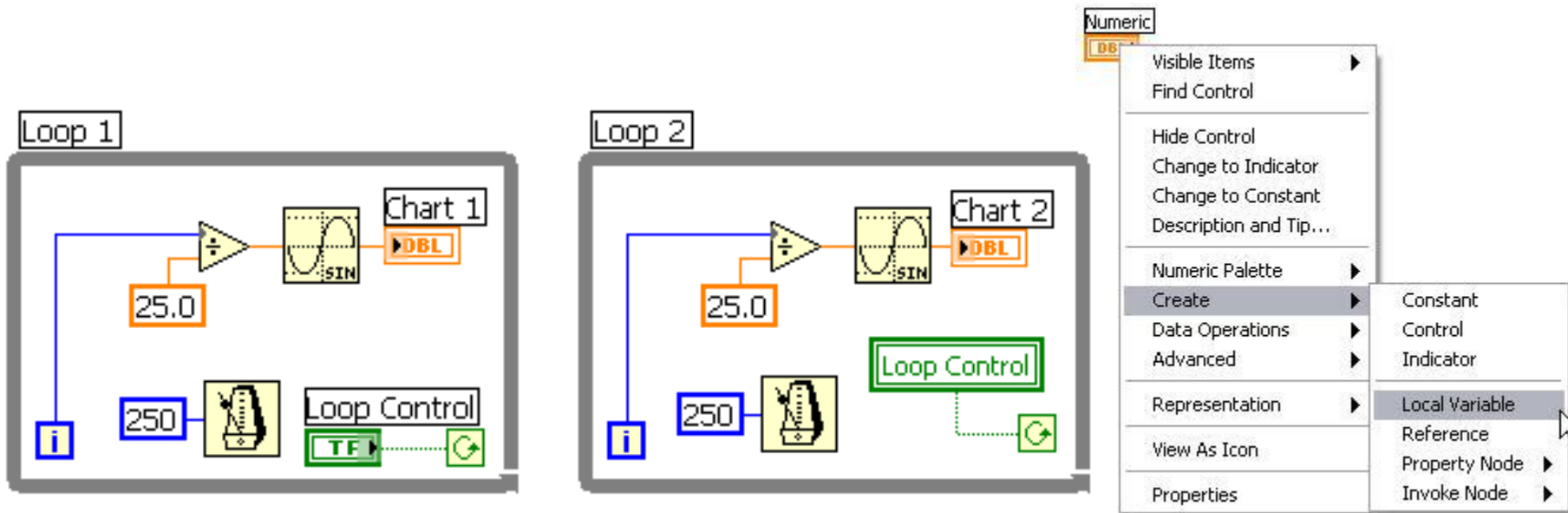


In LabVIEW werden nur programmatisch erzeugte bzw. an der Benutzeroberfläche auftretende Ereignisse unterstützt, jedoch keine Ereignisse, die über externe Quellen ausgelöst werden.

# Lokale Variablen

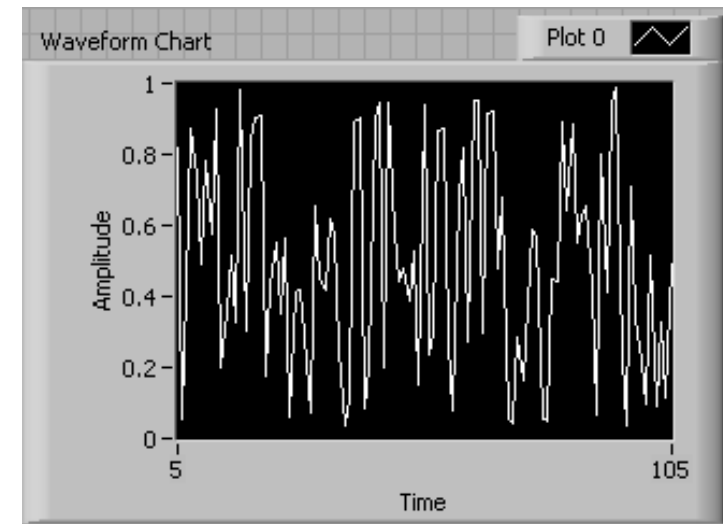
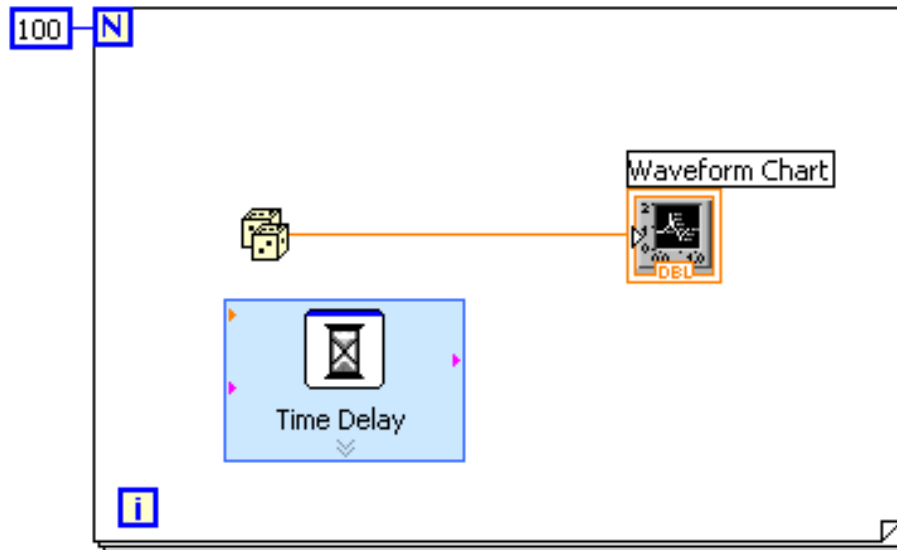
- für Daten zwischen parallelen Schleifen.
- lesen – schreiben an vers. Stellen
- unterbrechen das Datenflussparadigma → Vorsicht!!!

Bis hier



# Kapitel 10 Graphen und Diagramme (Chart)

- Diagramm:
  - Datenpunkten werden immer neu hinzugefügt, → Daten-Historie entsteht.
  - bei langsamen Prozessen, einige wenige Werte pro Sekunde

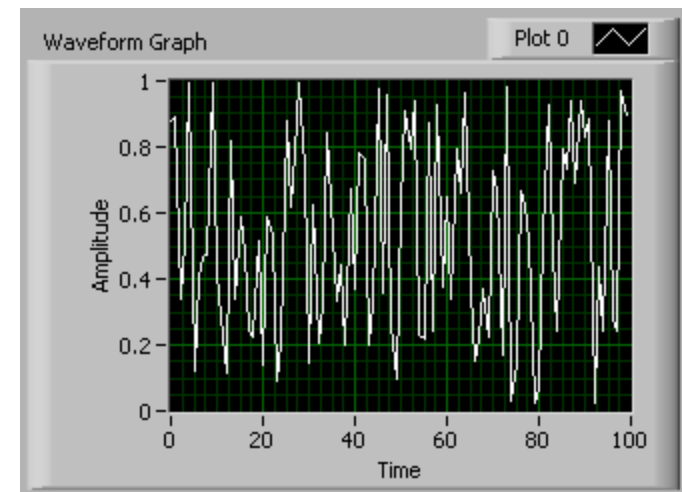
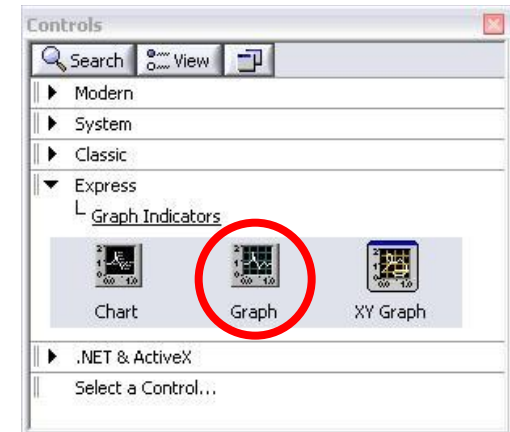
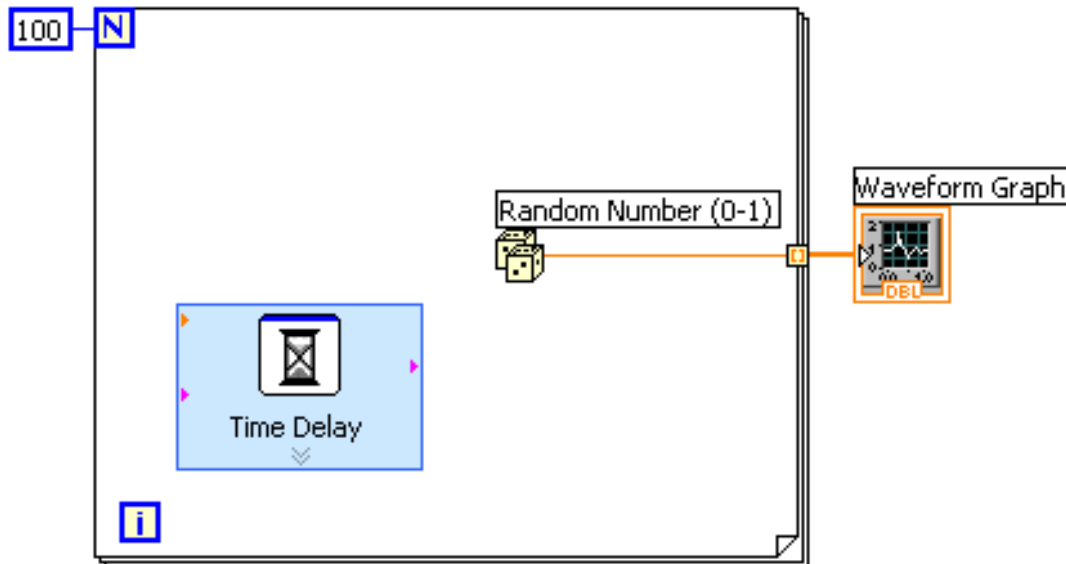


# Graphen und Diagramme

- Graphen:

- Datenarray nötig

Graphen werden normalerweise bei schnellen Prozessen mit kontinuierlicher Datenerfassung eingesetzt.

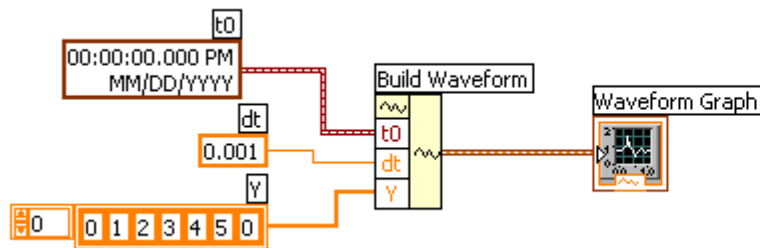


# Using Arrays and Clusters with Graphs

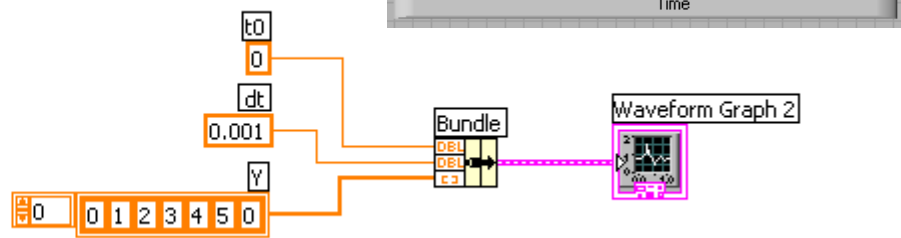
The Waveform Datatype contains 3 pieces of data:

- $t_0$  = Start Time
- $dt$  = Time between Samples
- $Y$  = Array of  $Y$  magnitudes

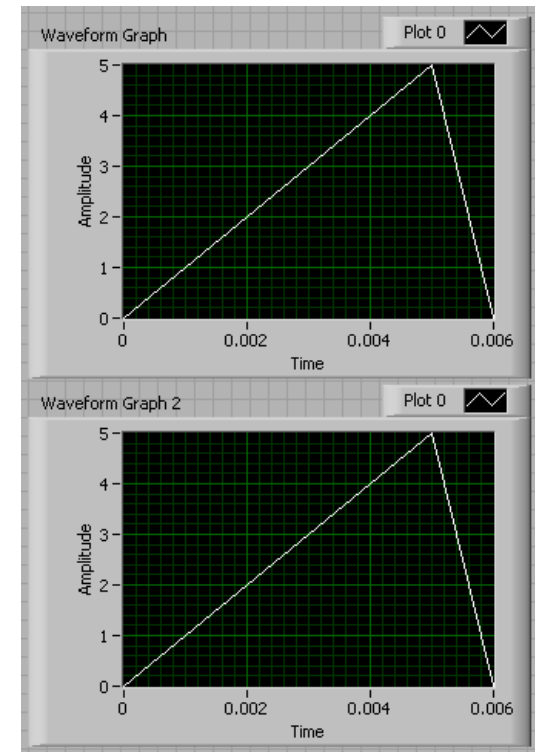
Two ways to create a Waveform Cluster:



**Build Waveform (absolute time)**

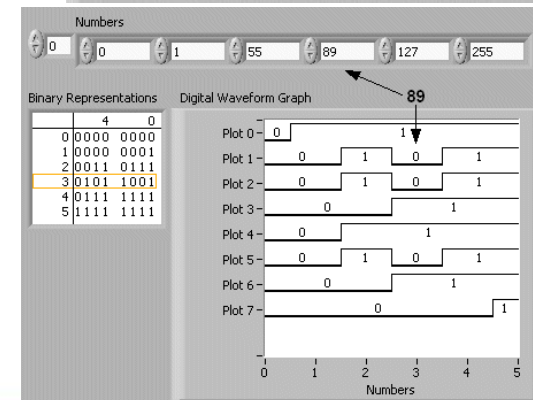
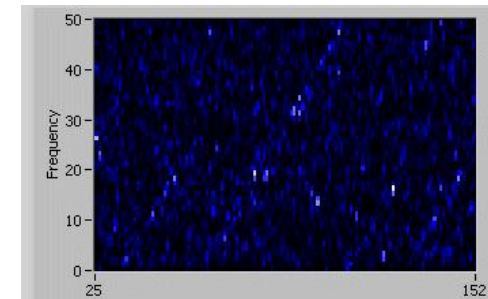
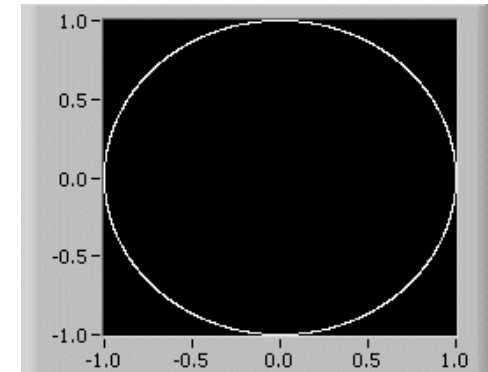


**Cluster (relative time)**

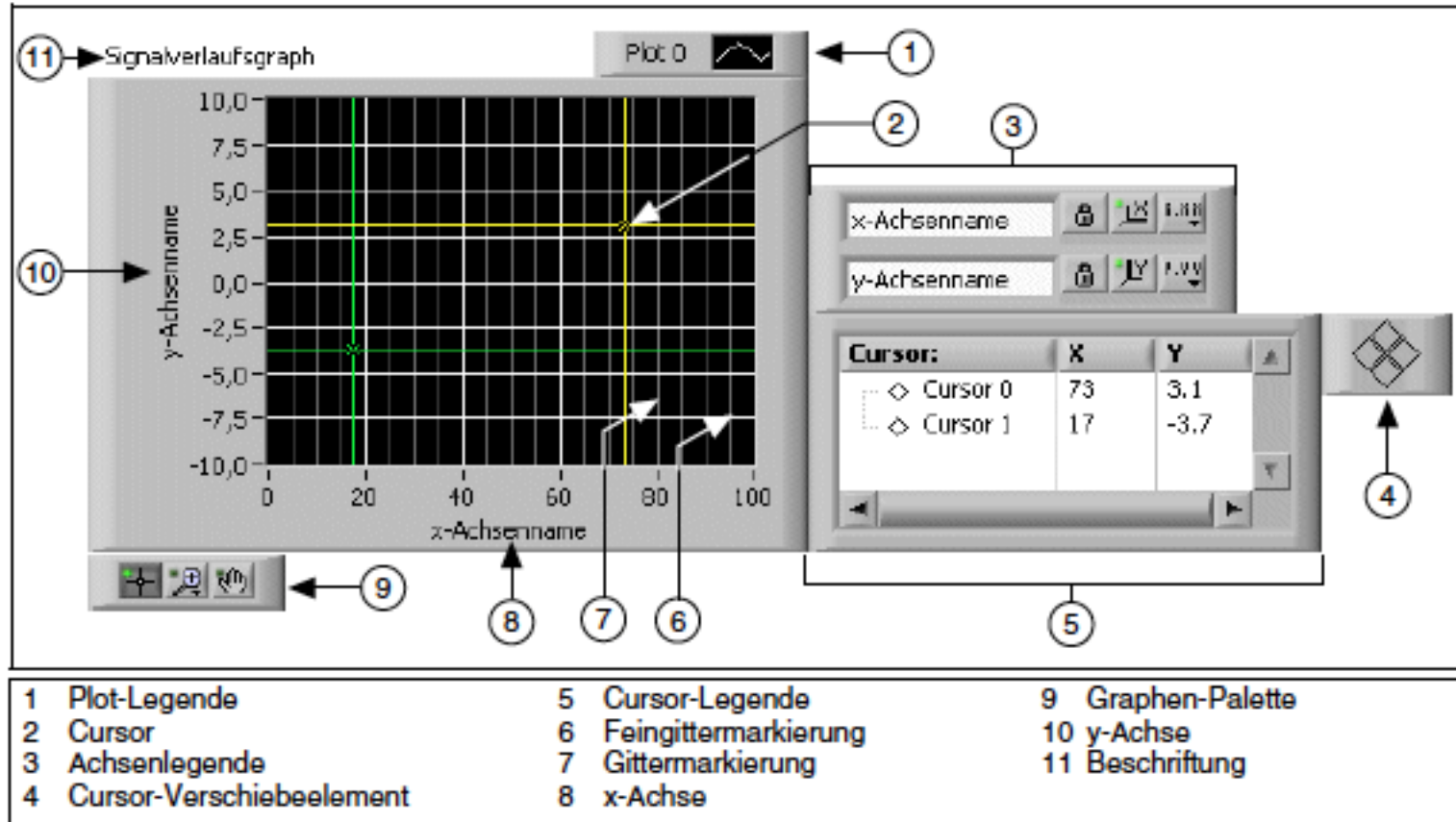


# Arten von Graphen und Diagrammen:

- Signalverlaufsdigramme und Signalverlaufsgraphen  
Zeigen die mit einer konstanten Rate erfassten Werte an.
- XY-Graphen  
Zeigen Werte von Funktionen mit mehreren Werten und Daten an, die nicht mit einer konstanten Rate erfasst wurden.
- Intensitätsdiagramme und Intensitätsgraphen  
Zeigen Werte von drei Dimensionen in einem zweidimensionalen Plot an, wobei die Werte der dritten Dimension anhand von Farben dargestellt werden.
- Digitale Signalverlaufsgraphen  
Zeigen Daten als Impulse oder als Gruppen digitaler Signalformen an.



# Anpassen der Darstellung von Graphen und Diagrammen



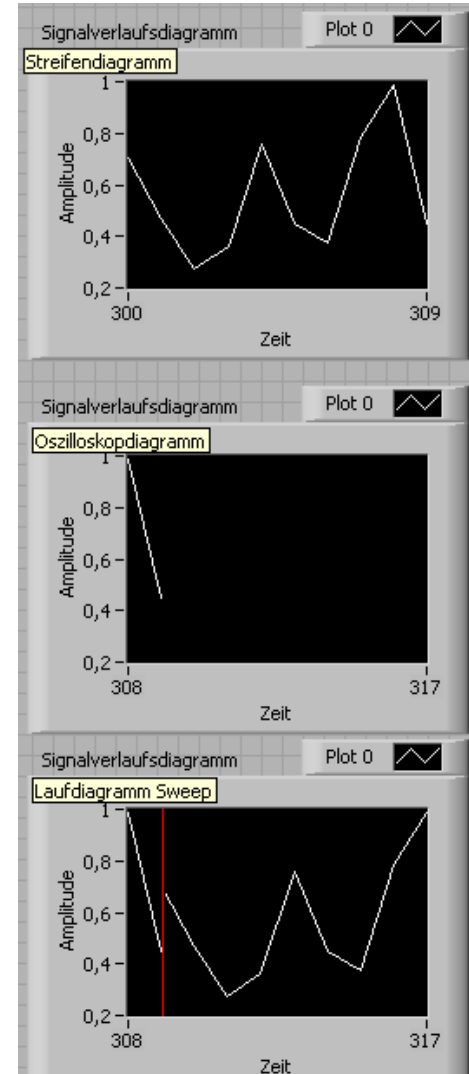
- Die meisten der in der oben dargestellten Legende angezeigten Objekte lassen sich einblenden, indem Sie den Graphen mit der rechten Maustaste anklicken, aus dem Kontextmenü die Option Sichtbare Objekte und dann das entsprechende Element auswählen.



# Konfigurieren des Aktualisierungsmodus von Diagrammen

- Streifendiagramm
- Oszilloskopdiagramm (Impuls oder eine Kurve)
- Laufdiagramm (wie ein Oszilloskopdiagramm, jedoch wenn die Kurve den rechten Rand des Darstellungsbereiches erreicht hat, wird sie nicht gelöscht, sondern läuft weiter).

*Siehe Graph einfach*



# Kapitel 11 Datei-I/O

- Daten aus Dateien ausgelesen oder in Dateien geschrieben.

beispielsweise:

- Öffnen und Schließen von Dateien
- Lesen von Daten aus und Schreiben von Daten in Dateien
- Lesen von Daten aus und Schreiben von Daten in Dateien im Tabellenkalkulationsformat
- Verschieben und Umbenennen von Dateien und Verzeichnissen
- Ändern von Dateieigenschaften
- Erstellen, Ändern und Lesen einer Konfigurationsdatei

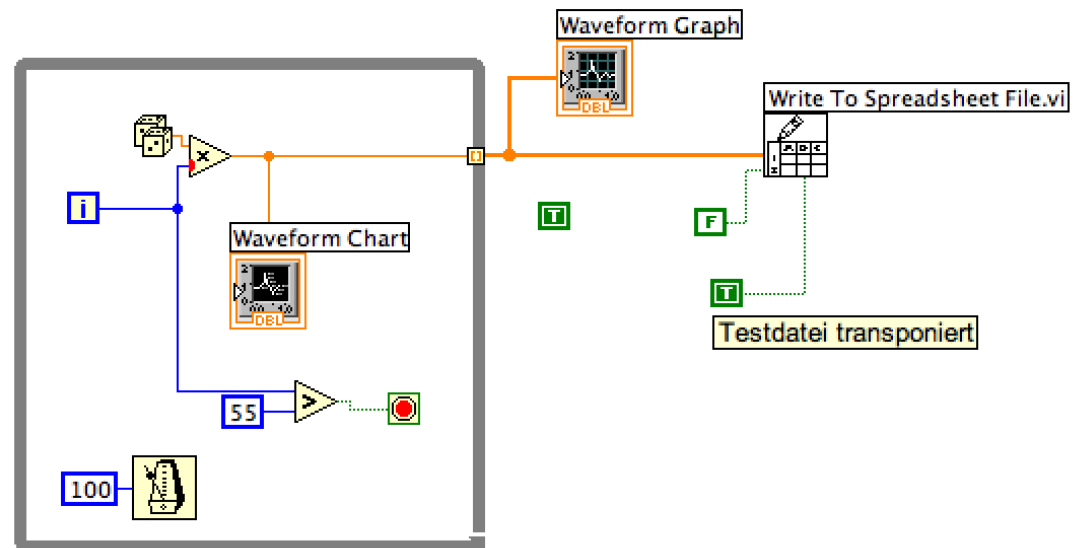
# Auswahl eines Datei-I/O-Formats

Text-, Datenprotokoll- und im Binärformat.

- meist Textdateien, da diese am häufigsten verwendet werden und am einfachsten portiert werden können. (nicht in Microsoft Excel – im ASCII Zeichensatz sind auch Buchstaben)
- Binärdateien,
  - wenn in unregelmäßiger Folge Schreib- oder Lesezugriffe durchgeführt werden müssen oder
  - Geschwindigkeit und
  - Festplattenspeicherplatz von Bedeutung sind
- Datenprotokolldateien bei komplexe Datensätze oder unterschiedliche Datentypen

# In Tabellenkalkulationsdatei schreiben

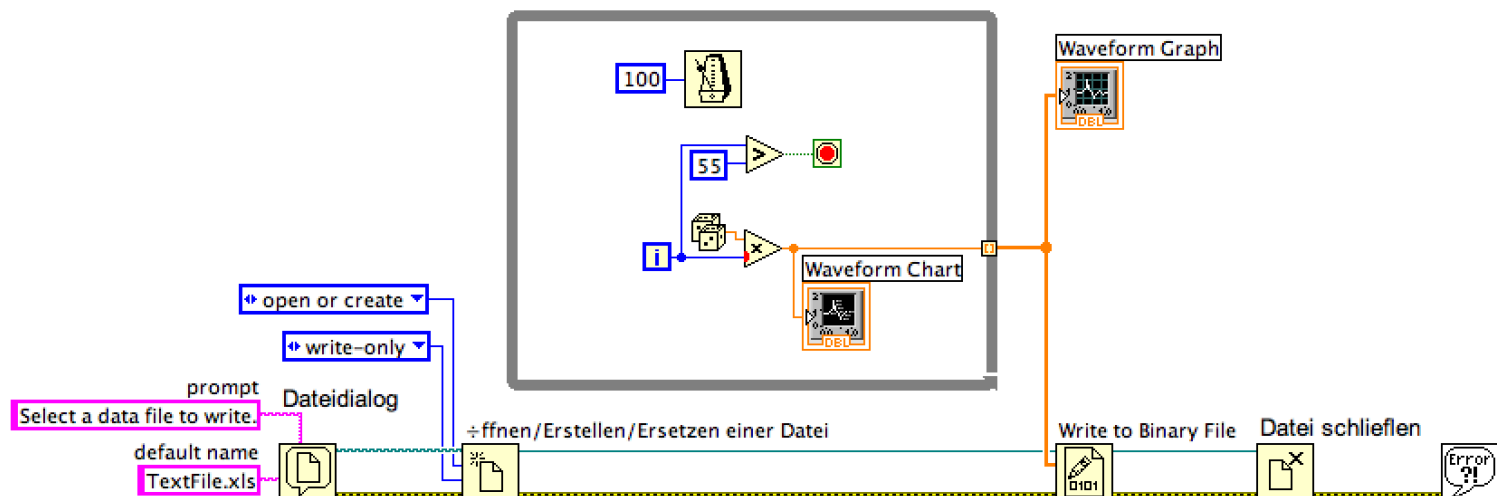
- Beispiel für die Anwendung des VIs “In Tabellenkalkulationsdatei schreiben”, um Zahlen an eine durch Tabulatoren unterteilte Tabellenkalkulationsdatei zu übergeben.
- Bei Ausführung dieses VIs werden Sie von LabVIEW aufgefordert, die Daten in eine vorhandene Datei zu schreiben oder eine neue Datei zu erstellen.



# ...oder auch so...

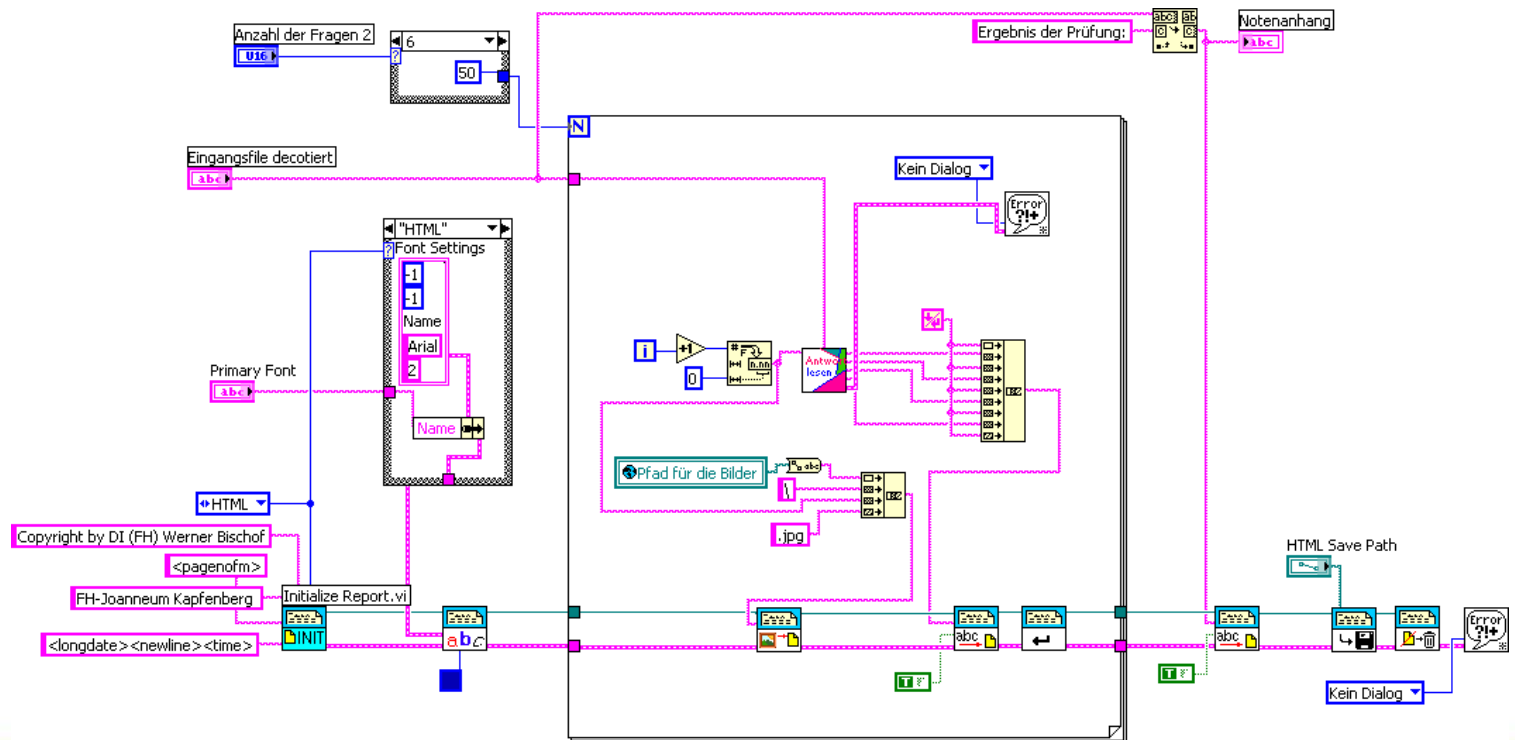
- Vergleichen Sie dieses Blockdiagramm mit dem VI “In Tabellenkalkulationsdatei schreiben”, welches dieselbe Aufgabe durchführt.

Im vorherigen Blockdiagramm werden für jede Dateioperation einzelne Funktionen (einschließlich "Array in Tabellenstring") verwendet, um das Array aus Zahlen in einen String umzuwandeln. Das VI "In Tabellenkalkulationsdatei schreiben" führt mehrere Dateioperationen aus. Dazu gehören das Öffnen der Datei, die Konvertierung des Zahlen-Arrays in einen String und das Schließen der Datei.

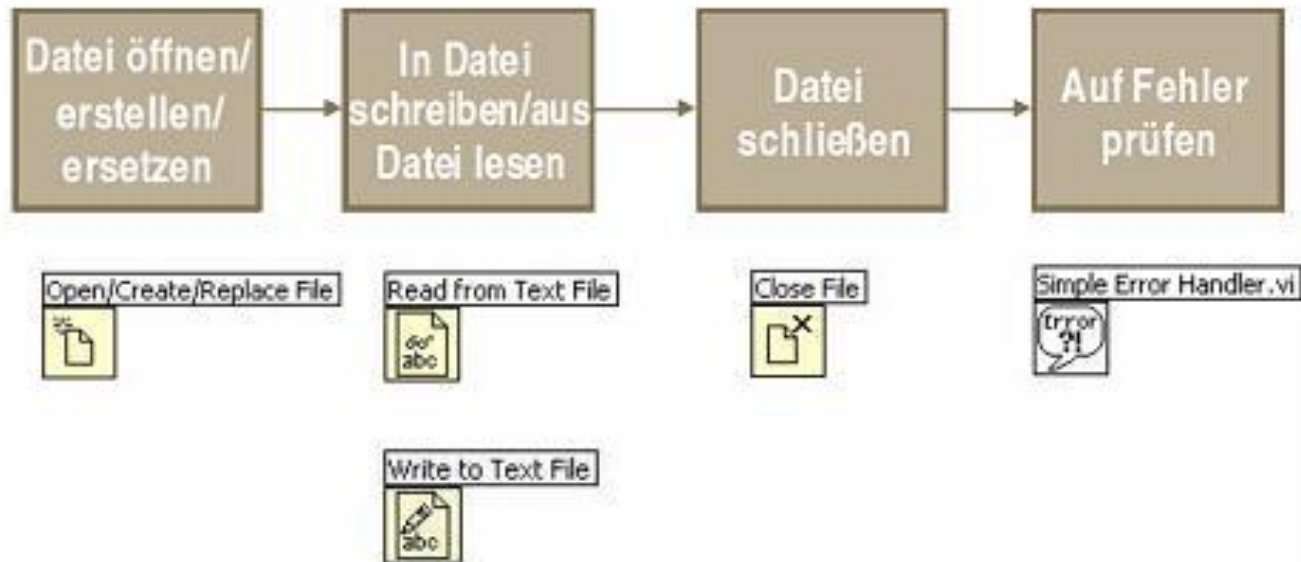


# Datenträger-Streaming

- Datei-I/O-Funktionen auch zum Datenträger-Streaming verwenden, wodurch Speicherressourcen durch weniger häufiges Öffnen und Schließen eingespart werden.
- Dabei handelt es sich um ein Verfahren, bei dem Dateien geöffnet bleiben, wenn mehrere Schreiboperationen (beispielsweise in einer Schleife) durchgeführt werden.



# Datei-I/O



# Kapitel 12 Dokumentieren und Drucken von VIs

- Informationen zum Blockdiagramm oder Frontpanel in jeder beliebigen Entwicklungsphase
- Drucken oder Reports für die ausgegebenen Daten und Ergebnisse.  
Die Vorgehensweise richtet sich nach mehreren Faktoren, zum Beispiel danach,
  - ob manuell oder
  - programmatisch gedruckt werden soll,
  - wie viele Optionen für das Berichtsformat benötigt werden,
  - ob diese Funktion auch in der entsprechenden ausführbaren Anwendung benötigt wird
  - oder auf welchen Plattformen die VIs laufen sollen.



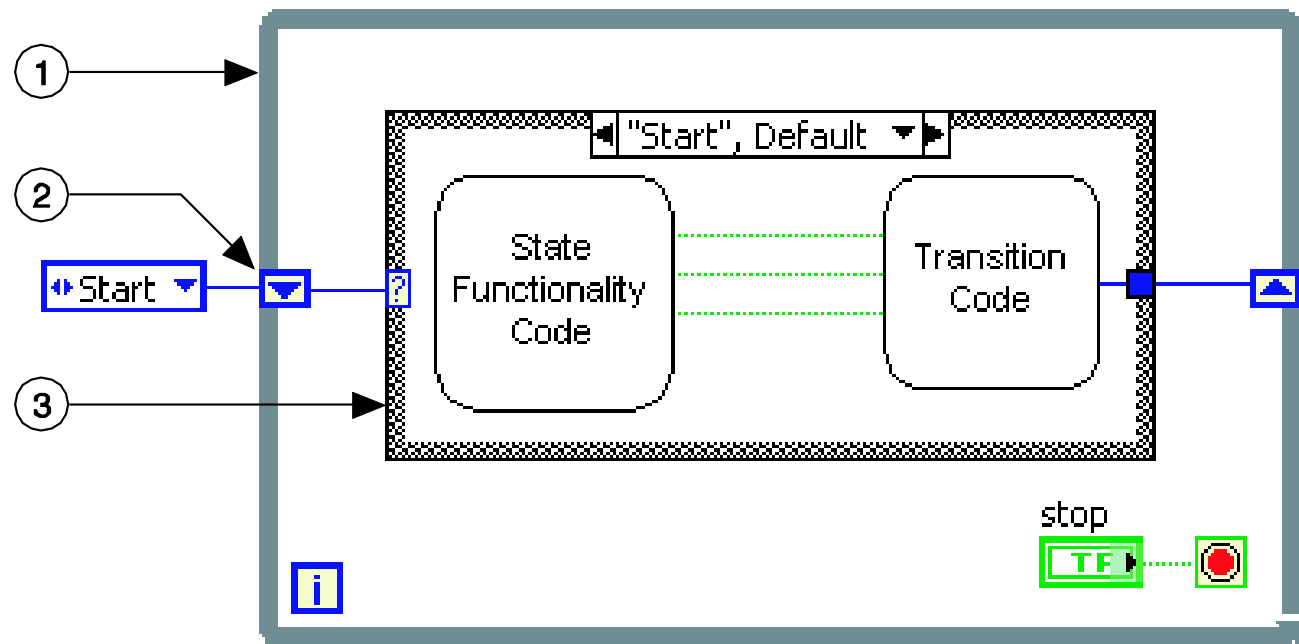
# VI-Beschreibungen, Hinweisstreifen

- Pulldown- Menü Kategorie die Option Dokumentation an.  
Wenn Sie den Cursor über das VI-Symbol bewegen, wird die Objekt- bzw. VI-Beschreibung im Fenster Kontexthilfe angezeigt.
- Hinweisstreifen sind kurze Beschreibungen, die angezeigt werden, wenn während der VI-Ausführung der Cursor über ein Objekt bewegt wird.

Den Inhalt des Hinweisstreifens legen Sie im Dialogfeld Beschreibung und Tipp fest. Wenn Sie keinen Tipp eingeben, erscheint kein Hinweisstreifen.

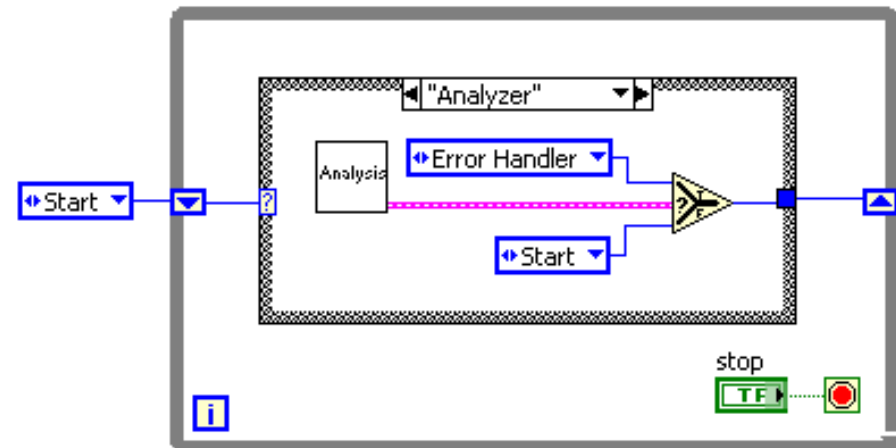
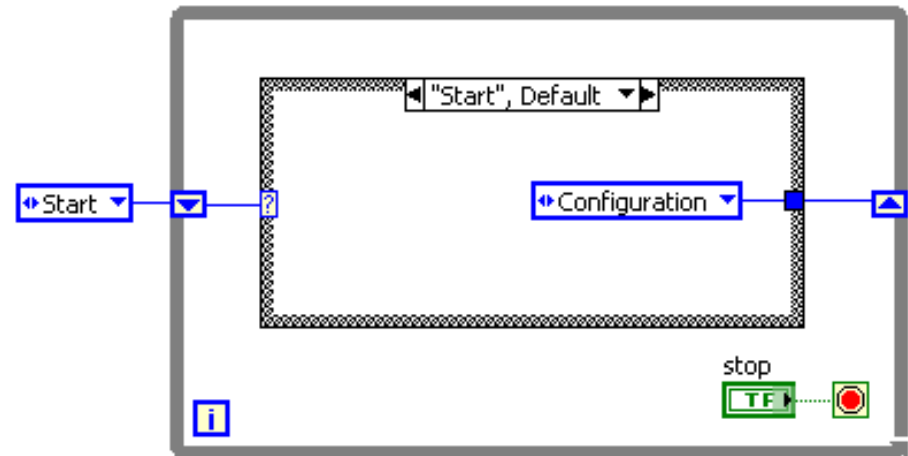
# State Machines

- While Loop
- Case Structure
- Shift Register



# State Machines Transitions

- Several programming techniques exist for transitioning from state to state in LabVIEW using State Machines
- Default transition implies that after one state, another state always follows
- Transitions between two potential states can be handled by a Select Function



# Section VI - Instrument Control

- A. Overview of Instrument Control
- B. GPIB
- C. Serial
- D. Instrument I/O Assistant
- E. VISA
- F. Instrument Drivers

# What Types of Instruments Can Be Controlled?

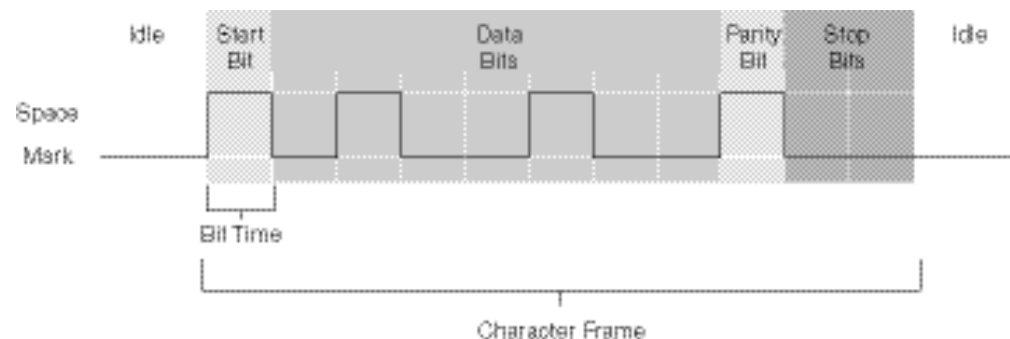
- GPIB
- Serial
- Modular Instruments
- PXI Modular Instruments
- Image Acquisition
- Motion Control
- USB
- Ethernet
- Parallel Port
- CAN

# GPIB

- General Purpose Interface Bus (GPIB)
- GPIB is usually used in stand alone bench top instruments to control measurements and communicate data
- Digital 8-bit parallel communication interface
- IEEE 488.1 and 488.2 define standards for GPIB

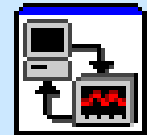
# Serial

- Serial communication transmits one bit at a time over a transmission line
- Usually does not require external hardware
- Four parameters: baud rate, data bits, parity bit, stop bits



# Instrument I/O Assistant

- LabVIEW Express VI used to communicate with message-based instruments
- Communicate with an instrument that uses a serial, Ethernet, or GPIB interface
- Use the Instrument I/O Assistant when an instrument driver is not available



Instrument I/O  
Assistant



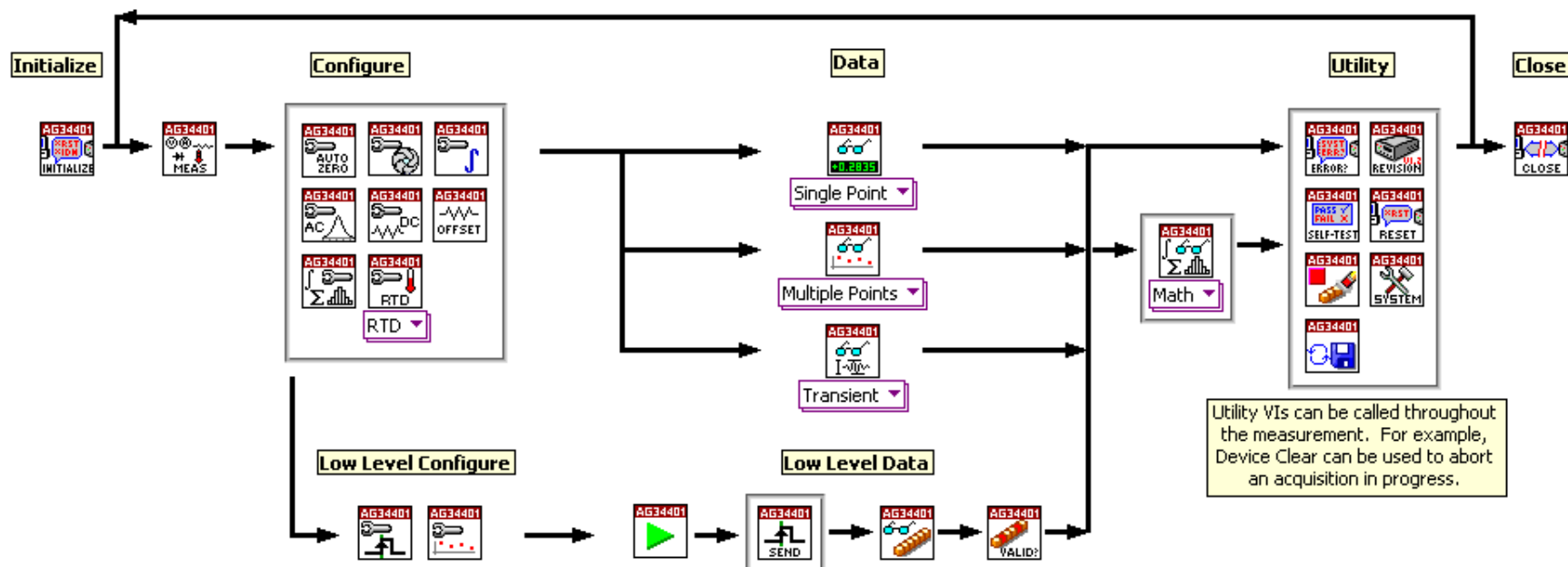
# VISA

- Virtual Instrumentation Software Architecture (VISA)
- High-level API that calls low-level drivers
- Can control VXI, GPIB, serial, or computer-based instruments
- Makes appropriate driver calls depending on the instrument used.

# Instrument Drivers

## Agilent 34401 VI Tree

Use the Example Finder to find examples demonstrating the usage of this instrument driver.  
To launch Example Finder, select "Find Examples..." from the LabVIEW Help menu.



**Triggers:** Immediate, External, Internal  
**Samples:** 1 - 50000 per trigger (34401, 34410)  
 1 - 1000000 per trigger (34411)

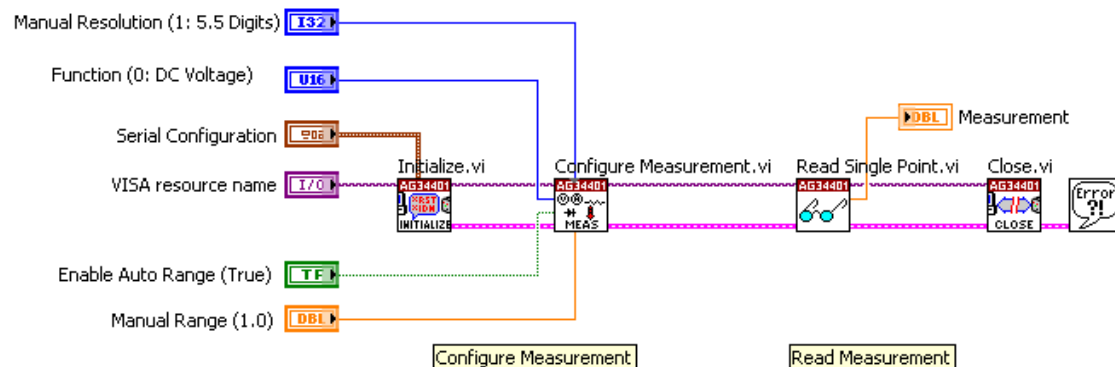
**Triggers:** Software  
**Samples:** 1 - 512 total (34401)  
 1 - 50000 total (34410)  
 1 - 1000000 total (34411)

Utility VIs can be called throughout the measurement. For example, Device Clear can be used to abort an acquisition in progress.

**VIs in gray boxes are optional**

# Instrument Drivers

- Plug and Play drivers are a set of VIs that control a programmable instrument
- VIs correspond to instrument operation: configuring, triggering, and reading measurements
- Help getting started since programming protocol for each instrument is already known



# The LabVIEW Certification Program

## Architect

- Mastery of LabVIEW
- Expert in large application development
- Skilled in leading project teams

Certified  
LabVIEW  
Architect

## Developer

- Advanced LabVIEW knowledge and application development experience
- Project management skills

Certified LabVIEW  
Developer

## Associate Developer

- Proficiency in navigating LabVIEW environment
- Some application development experience

Certified LabVIEW Associate  
Developer

## Fundamentals Exam

- Pre-Certification Skills Test

**Free On-Line Fundamentals Exam**

# Deutsche Fachbücher mit der Studentenversion von LabVIEW



**LabVIEW für Studenten**  
Autor: R. Jama/A. Hagedorn  
Verlag: Pearson Studium, 08/2004  
576 Seiten;  
mit CD-ROM (LabVIEW7);  
4., veränderte Ausgabe  
Preis: 49,95 €  
ISBN 3-8273-7154-6



**Einführung in LabVIEW**  
Autoren: W. Georg/E. Metin  
Verlag: Hanser Fachbuch-verlag  
Leipzig, 03/2006  
(2. Ausgabe 09/2006)  
328 Seiten; broschiert  
Preis: 39,90 €  
ISBN 3-446-40400-7



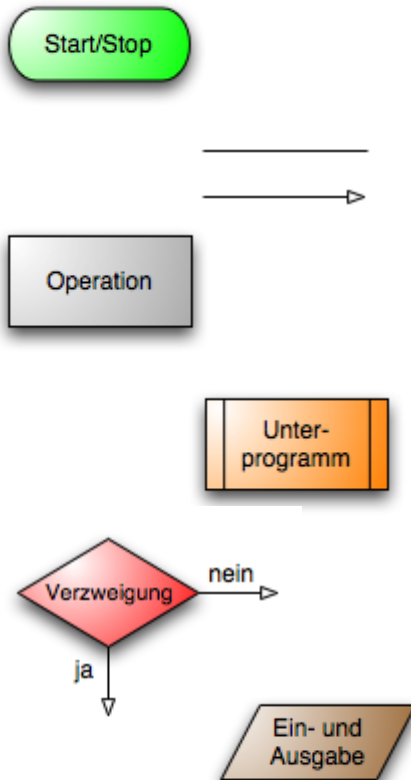
**Elektrische Messtechnik**  
Autor: R. Lerch  
Verlag: Springer Verlag 09/2006  
600 Seiten  
Preis: 42,95 €  
ISBN 3-540-34055-6



**Handbuch für die Programmierung mit LabVIEW**  
Autor: B. Müttelein  
Verlag: Elsevier Verlag,  
ab Jan 2007; 460 Seiten  
Preis: 49,50 €  
ISBN (978)3-8274-1761-9

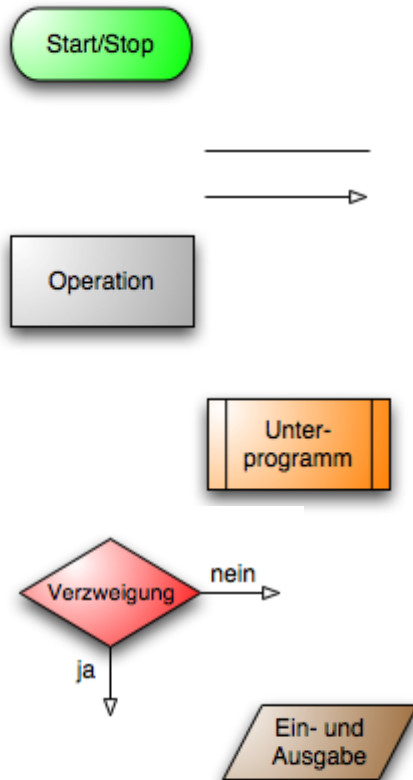
# Flussdiagramm

Ein F. ist eine graphische Darstellung zur Umsetzung eines Algorithmus in einem Programm und beschreibt die Folge von Operationen zur Lösung einer Aufgabe. DIN 66001



- Oval: Start, Ende, weitere Grenzpunkte
- Pfeil, Linie: Verbindung zum nächstfolgenden Element
- Rechteck: Operation (Eine Tätigkeit beschreibt einen bestimmten Vorgang, der innerhalb des jeweiligen Prozesses durchgeführt wird.)
- Rechteck mit doppelten, vertikalen Linien: Ein Unterprogramm das eine Teilaufgabe erarbeitet!
- Raute: Verzweigung Eine Entscheidung hat einen Eingang (oben) und mehrere Ausgänge (links, rechts und unten; Ja/Nein, usw.).
- Parallelogramm: Ein- und Ausgabe z.B.: Abfrage des Users; Plot

# Flussdiagramm

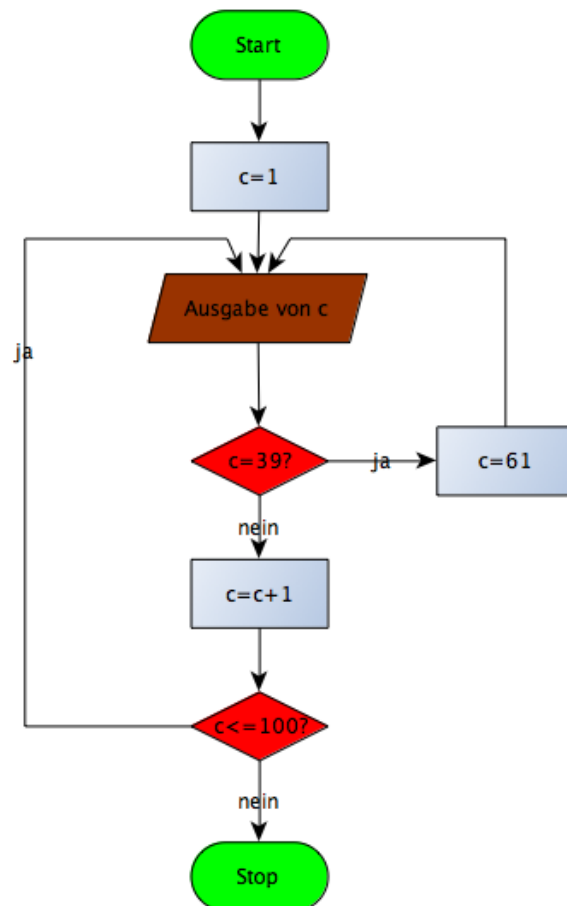


## Aufgabe Counter

Schreiben Sie ein Programm das c bis 101 hochzählt.  
Wenn c bei 39 angekommen ist – soll c auf den neuen Wert von 61 springen (geändert werden) und bis auf 101 weiterzählen.  
Zur Kontrolle soll bei c = 39 eine Lampe aktiviert werden.

c ist nicht der index-Zähler – c ist nur eine Variable

# Flussdiagramm



z.B.: Die nebenstehende Abbildung zeigt eine einfache Zählschleife von 1 bis 101.

Die Variable  $c$  wird vor Beginn der Schleife auf ihren Startwert  $c=1$  gesetzt.

Danach wird die erste Anweisung der Schleife, das Ausgeben der Variable  $c$ , ausgeführt.

Die nachfolgende zweite Anweisung ist eine einseitige Auswahl, die prüft, ob  $c$  den Wert 39 besitzt.

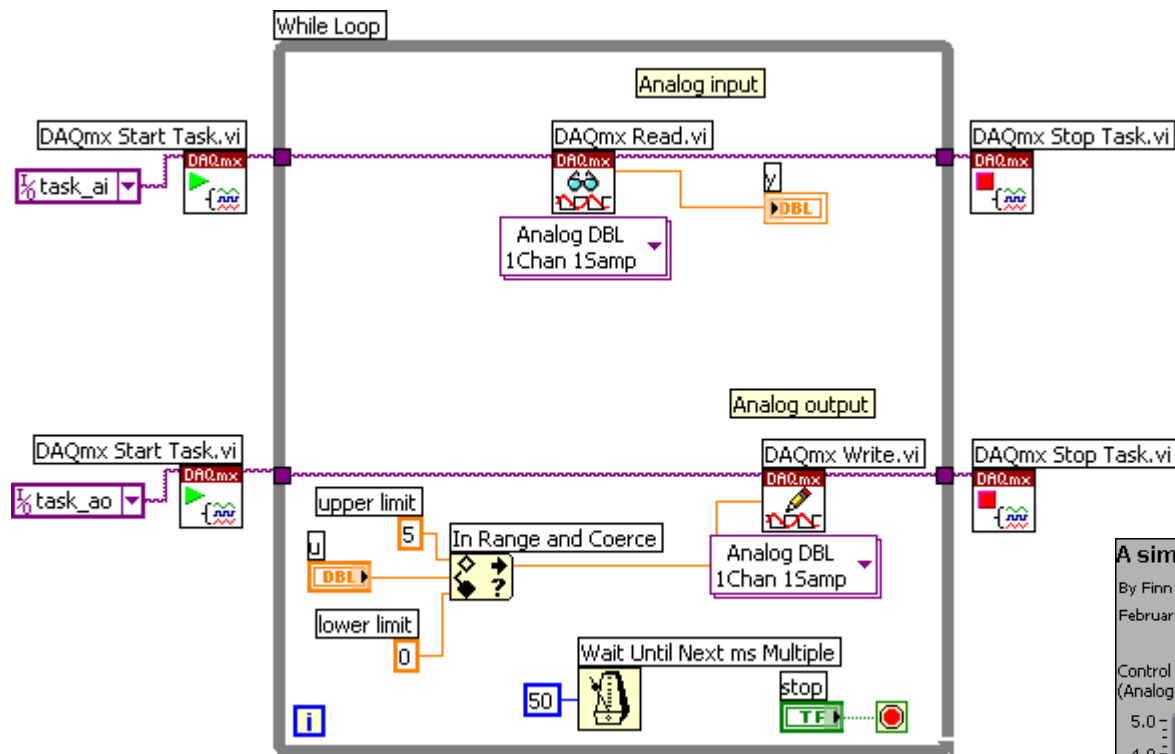
Falls dies der Fall ist, wird  $c$  auf den Wert 61 gesetzt und die Schleife beginnt mit dem nächsten Durchlauf.

Falls  $c$  nicht 39 ist, wird  $c$  in der nachfolgenden Anweisung um eins erhöht und anschließend geprüft, ob die Schleifenabbruchbedingung  $c \leq 100$  erreicht ist.

Falls nicht, erfolgt ein nochmaliger Schleifendurchlauf. Ausgegeben werden alle ganzen Zahlen von 1 bis 39 sowie 61 bis 100 (jeweils einschließlich).



# USB 6008

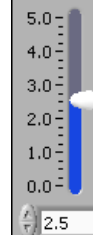


## A simple VI realizing analog output (AO) and analog input (AI)

By Finn Haugen (finn@techteach.no)

February 20 2006

Control signal, u [V]  
(Analog output)

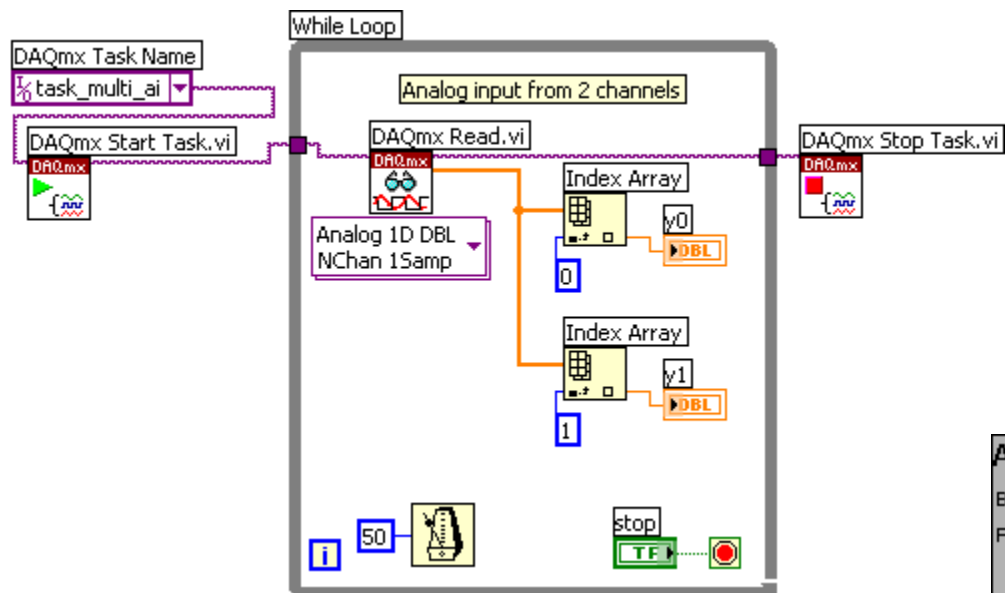


Measurement, y [V]  
(Analog input)



STOP

# USB 6008 Multi Read



## A simple VI realizing multiple analog input (AI)

By Finn Haugen (finn@techteach.no)

February 20, 2005

Measurement, y0 [V]  
(Analog input)



2.0

Measurement, y1 [V]  
(Analog input)

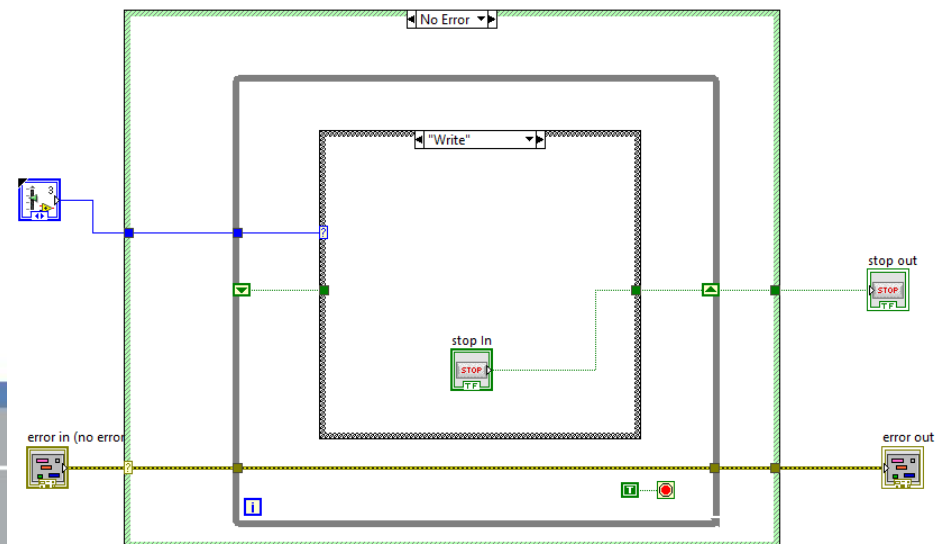
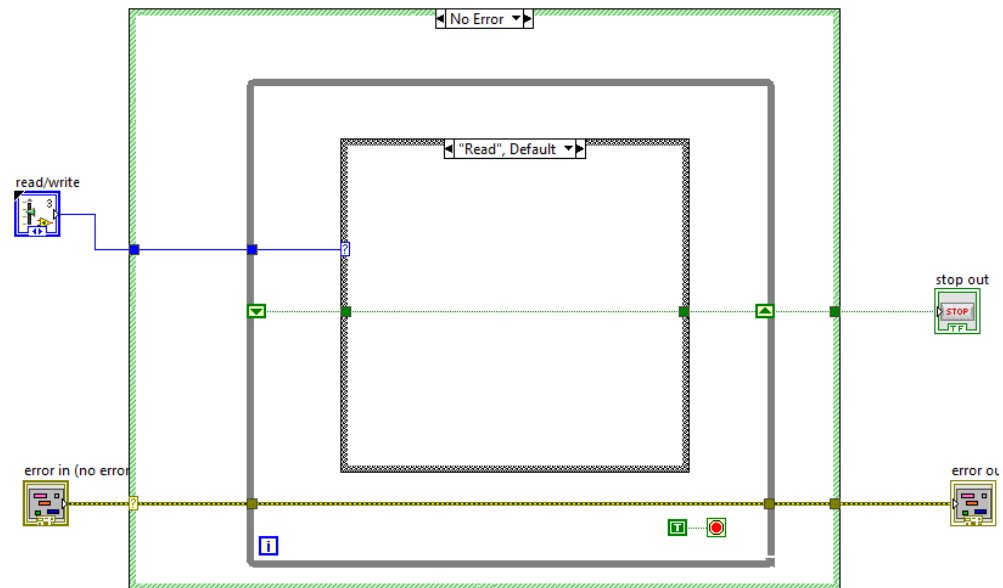
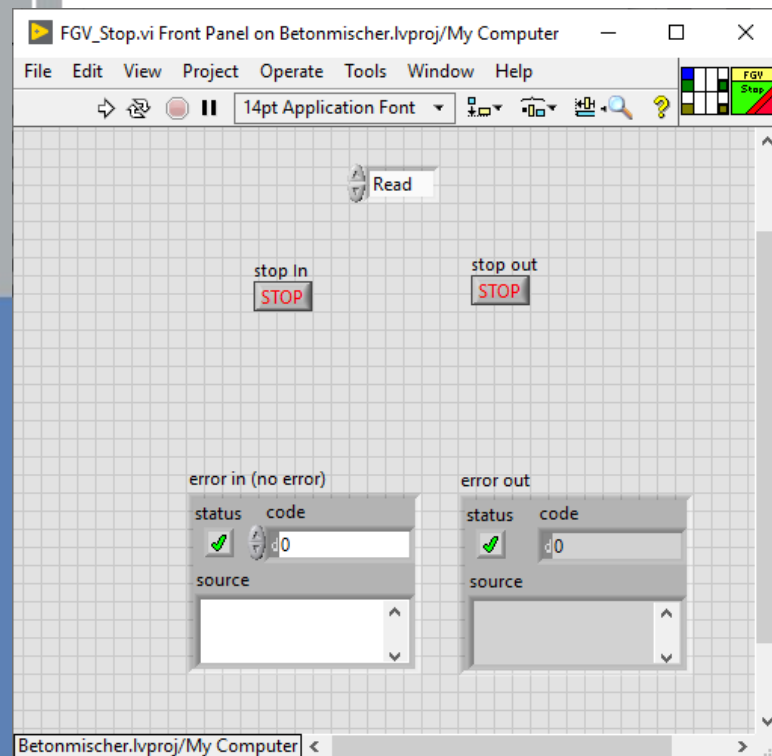


0.0

STOP

# FGV → Funktionale Globale Variable

[https://learn-ni.com/teach/riodevguide/code/rt\\_functional-global-variable.html](https://learn.ni.com/teach/riodevguide/code/rt_functional-global-variable.html)



# FGV → Funktionale Globale Variable

Vorteil:

- globaler Zugriff ohne Race Conditions
- einfach
- auch als Zähler mit Increment und Decrement
- mit Cluster beliebig erweiterbar

Wichtig:

Type Definition

