

*Bernd vom Berg  
Peter Groppe*

# *LabVIEW für den Praktiker*

## **LabVIEW-Projekt**

# **„Erfassung und Darstellung eines Temperaturmesswertes“**

(Stand: 23.05.2012, V1.0)



## Worum geht's ?

Dieses Projekt soll der erste Einstieg in die Verwendung von LabVIEW in Zusammenhang mit (beliebigen) Mikrocontroller-Systemen sein.

Die bidirektionale Kommunikation zwischen beiden Systemen erfolgt dabei zunächst ganz einfach über eine serielle COM-Schnittstelle, wobei auch USB-Schnittstellen über entsprechende USB/Seriell-Adapter zum Einsatz kommen können.

In diesem Projekt wird zunächst eine kurze Einführung in die Möglichkeiten und in die Anwendung von LabVIEW gegeben.

Danach wird über eine serielle Schnittstelle ein 8051er-Mikrocontroller-System angeschlossen, das einen Temperaturmesswert an den PC/LapTop übermittelt, der dann dort mittels LabVIEW empfangen, ausgewertet und auf dem Monitor auf verschiedene Art und Weise dargestellt wird.

Da das Datenübertragungsprotokoll sehr einfach aufgebaut ist und offen gelegt wird, kann aber auch jedes beliebige Mikrocontroller-System (AVR, PIC, ARM, ...), das über eine serielle UART-Schnittstelle verfügt, angeschlossen werden und Messwerte an LabVIEW senden.

---

Bei der Zusammenstellung von Texten, Abbildungen, Stückteillisten, u.s.w. wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden.

Die Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Für die Mitteilung eventueller Fehler sind die Autoren dankbar.

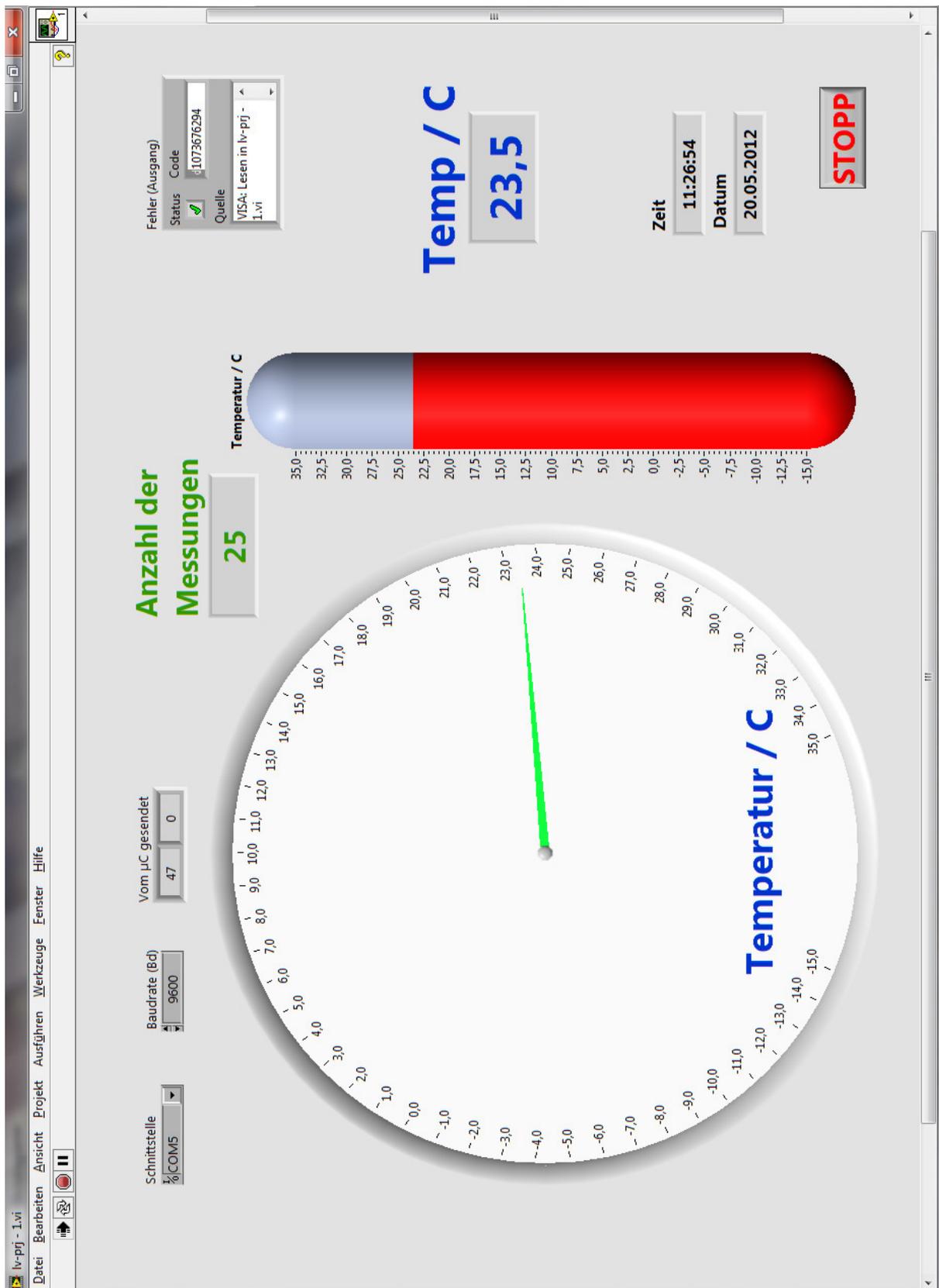
Alle von uns im Rahmen des Projektes erstellten Programme sind zur freien, nicht kommerziellen Verwendung freigegeben, sofern die Rechte Dritter (z.B. von Seiten der Firma National Instruments (NI)) nicht betroffen sind.

# Inhalt

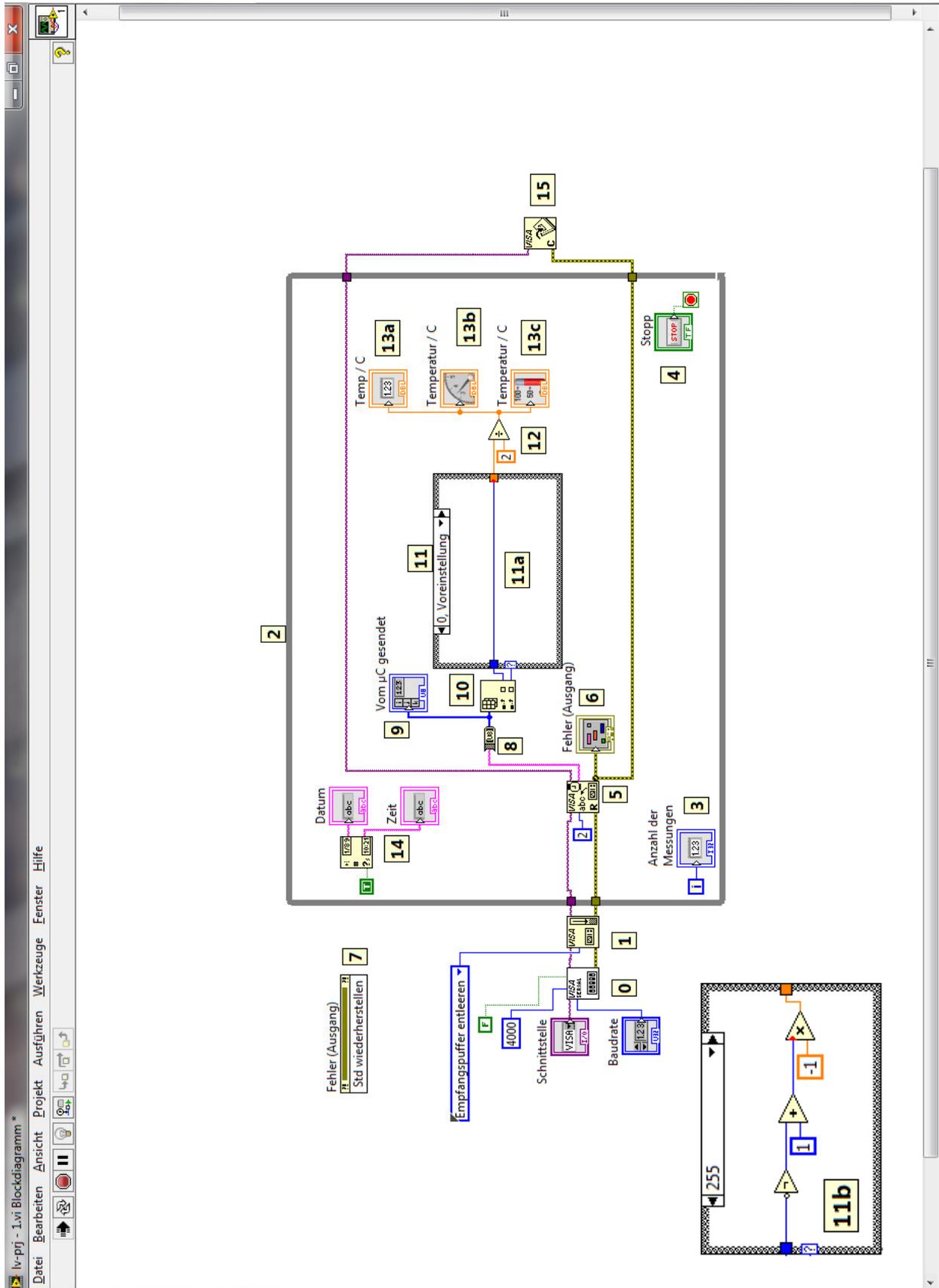
(Stand: 23.05.2012)

- 1. Einleitung**
- 2. Das Projekt-Szenario**
  - 2.1 Das Mikrocontroller-System als Datenquelle
- 3. Die Installation und der Start von LabVIEW**
- 4. LabVIEW kompakt**
  - 4.1 Allgemeines
  - 4.2 Der Entwurf des Blockdiagramms
  - 4.3 Die Gestaltung des Frontpanels
  - 4.4 Der Start des VIs
- 5. Das Projekt**
  - 5.1 Das Blockdiagramm
  - 5.2 Das Frontpanel
  - 5.3 Test und Ergebnis
- 6. Und wie geht's weiter ?**
- 7. Anhänge**
  - 7.1 Liste der Shortcuts
- 8. Literatur, Seminare und Bezugsquellen**
- 9. Versions History**

## Ergebnis: Das Frontpanel



# Ergebnis: Das Blockdiagramm



## 1. Einleitung



Viele Entwickler und Anwender von Mikrocontroller-Systemen kennen das Problem: die kompakte Rechner-Einheit ist fertig und sie arbeitet genau so wie gewünscht – der Mikrocontroller erfasst Messwerte, berechnet Ausgangsgrößen, führt eine Steuerung und/oder eine Regelung durch und überträgt Daten.

**Aber ein Punkt bleibt sehr oft ungelöst** bzw. wird wenig zufrieden

stellend realisiert: die ansprechende Visualisierung von wichtigen Daten.

Oder ganz allgemein: **die Schaffung einer optisch gut gestalteten Benutzeroberfläche** auf der Messdaten graphisch dargestellt werden, über die interaktiv Benutzereingaben erfolgen können, auf der alle System- und Prozesskenndaten auf einen Blick übersichtlich abgebildet werden, ist und bleibt ein Problem.

Oft sind zwar PC oder Laptop mit ihren vollfarb/vollgraphischen Monitoren vorhanden und damit könnten alle nur denkbare Darstellungswünsche erfüllt werden, aber: wie programmiert man solch eine „optische Mensch/Maschine-Schnittstelle“ für den Rechner?

Mit den klassischen Programmiersprachen wie Visual Basic, C++, C# oder Java ist das sicherlich kein Problem, aber doch recht aufwendig und vor allen Dingen fehlt manchmal auch das notwendige Wissen, um solche Softwarepakete zu bedienen und einzusetzen.

An dieser Stelle ermöglicht nun das „**Laboratory Virtual Instrument Engineering Workbench**“ oder kurz das Programmpaket **LabVIEW**, hervorragend gestaltete Messgeräte- und Prozesssteueroberflächen in kürzester Zeit auf dem PC/Laptop zu entwerfen und zum Laufen zu bringen.

### **Jedoch:**

Der Einsteiger bzw. Neuanwender von LabVIEW steht am Anfang einer unübersichtlichen Vielfalt von hunderten von Anzeige- und Bedienelementen und Verarbeitungs-, Auswert- und Darstellungsfunktionen gegenüber, und er versucht meistens recht aufwendig und teilweise unstrukturiert den ersten Einstieg zu finden.

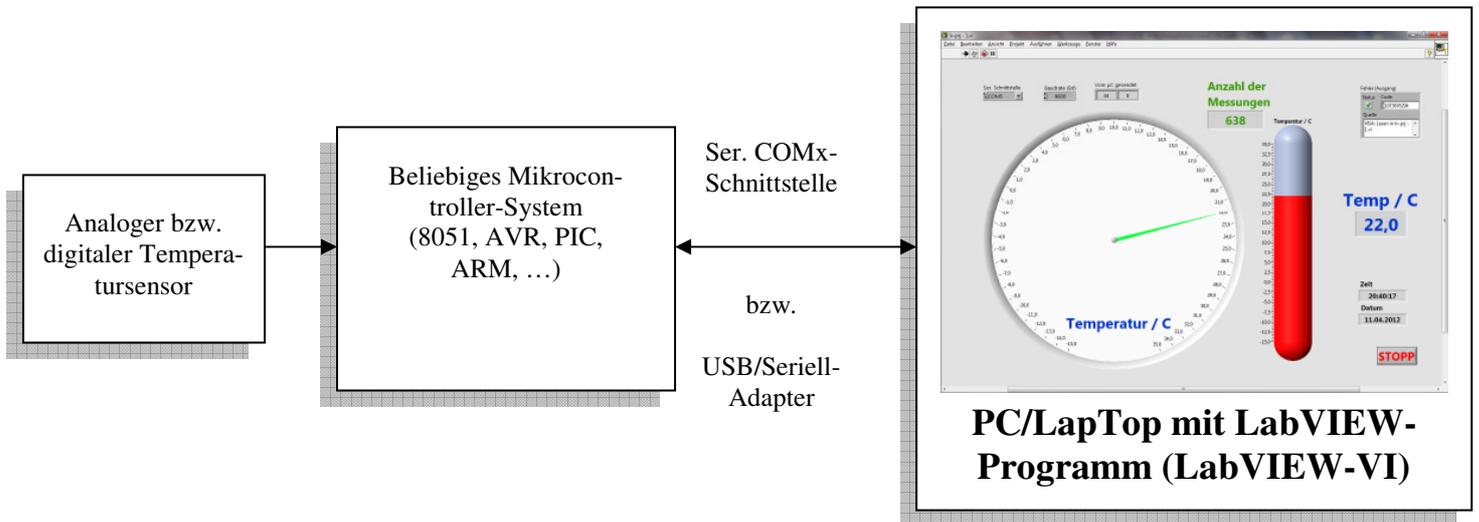
Und an dieser Stelle setzt dieses kleine Einsteiger-Projekt an:

An einem einfachen Praxisbeispiel: „**Erfassung und Darstellung eines Temperaturmesswertes**“ wird in leicht nachvollziehbaren Schritten der Aufbau, die grundlegende Struktur und die Verwendung von LabVIEW dargestellt, erläutert und realisiert.

Die hierzu notwendigen Vorkenntnisse für den ersten Einstieg in LabVIEW sind äußerst gering, die bereits erreichbaren Ergebnisse dagegen äußerst sehenswert.

## 2. Das Projekt-Szenario

Die **Abb.2.1** zeigt das diesem Projekt zugrund liegende Szenario:

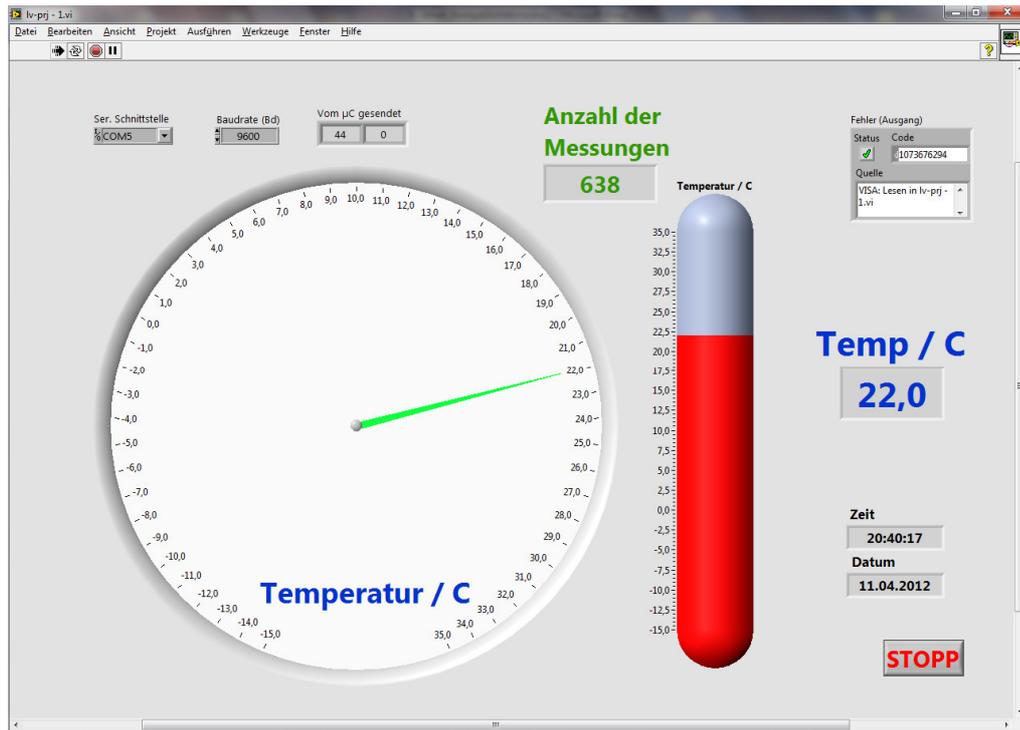


**Abb.2.1: Das Projekt-Szenario**

Ein beliebiges Mikrocontroller-System erfasst mit Hilfe eines analogen oder digitalen Sensors die aktuelle Umgebungstemperatur und sendet diesen Messwert in Form von zwei einzelnen Bytes über eine serielle UART-Schnittstelle an den PC bzw. LapTop.

Alternativ kann auch eine USB-Schnittstelle benutzt werden, wenn ein entsprechender USB/Seriell-Adapter zum Einsatz kommt.

Auf dem PC/LapTop wird der Messwert dann von einem LabVIEW-Programm (LabVIEW-VI) empfangen, aufbereitet und „sehr schön“ dargestellt, s. **Abb.2.2**:



**Abb.2.2: Das Endergebnis auf dem Monitor**

Dieses komplette LabVIEW-Programm soll nun in diesem Projekt Schritt-für-Schritt einfach nachvollziehbar entwickelt werden.

## 2.1 Das Mikrocontroller-System als Datenquelle

In unserer konkreten Anwendung kommen die folgenden Hardware-Komponenten zum Einsatz:

- 8051er-Mikrocontroller-System mit dem  $\mu C$  AT89C51CC03 der Firma Atmel: „**PT-First-Step-Board**“ für Lehre und Ausbildung (s. Kap.8).
- Digitaler 1-Wire-Bus-Temperatursensor **DS1820**.  
(s. dazu auch unser bereits veröffentlichtes Projekt „**1-Wire-Bus-Projekt**“).
- Standard-PC/LapTop entweder mit „echten“ seriellen COMx-Schnittstellen oder mit USB-Schnittstellen und USB/Seriell-Adapter.

Natürlich können auch **andere Mikrocontroller-Systeme** verwendet werden (AVR, PIC, ARM, ...), solange diese den Temperaturmesswert im nachfolgend beschriebenen Format über eine serielle UART-Schnittstelle an den PC/LapTop aussenden.

### Das verwendete Datenformat

Das LabVIEW-Programm erwartet den aktuellen Temperaturmesswert in Form von zwei hintereinander ausgesandten Bytes, die den folgenden Aufbau haben:

1. Byte							
x	x	x	x	x	x	x	x

2. Byte							
x	x	x	x	x	x	x	x

### **1. Byte: Messwert**

In diesem Byte befindet sich der eigentliche **Messwert** und dabei gilt:

- Ist der Messwert (also die Temperatur) **positiv**, so steht hier direkt der Wert der Temperatur und dieser Wert kann sofort entsprechend ausgewertet bzw. dargestellt werden.  
Die Auflösung, also der kleinste noch erkennbare Temperaturschritt, beträgt hierbei  $0,5^{\circ}\text{C}$ , d.h. die Wertigkeit eines Bits ist  $0,5^{\circ}\text{C}$ .
- Ist der Messwert (also die Temperatur) **negativ**, so steht hier der Wert der Temperatur in der **2er-Komplement-Darstellung**, d.h. der Wert muss erst noch umgeformt werden, um die korrekte negative Temperatur zu erhalten:  
Bitweise Invertierung des Wertes, hinzuaddieren von '1' und Multiplikation mit '-1'.

## 2. Byte: Vorzeichenbyte

In diesem Byte befindet sich die Information über das Vorzeichen des Temperaturmesswertes: hat dieses Byte den Wert '0', so stellt der Wert im ersten Byte eine positive Temperatur dar, das erste Byte braucht also nicht umgerechnet zu werden.

Hat dieses zweite Byte dagegen den Wert '255', so stellt der Wert im ersten Byte eine negative Temperatur dar, das erste Byte muss also wie zuvor beschrieben erst einmal umgerechnet werden.

### Beispiele:

**+68,5°C**  $\equiv$

1. Byte							
1	0	0	0	1	0	0	1

2. Byte							
0	0	0	0	0	0	0	0

**-11,5°C**  $\equiv$

1. Byte							
1	1	1	0	1	0	0	1

2. Byte							
1	1	1	1	1	1	1	1

### Der Kommunikationsablauf

Das Mikrocontroller-System sendet nun diese beiden Bytes zyklisch ca. jede Sekunde aus, das LabVIEW-Programm empfängt diese und wertet sie aus.

Die Datenübertragungsparameter dabei sind:

- 9.600 Bd.
- 8 Datenbits
- Keine Parität
- 1 Stoppbit
- Keine Flußsteuerung.

Natürlich können so auch beliebige andere Messwerte übertragen und dargestellt werden: man muss nur dafür sorgen, dass das Datenformat stimmt, d.h. dass die beiden gesendeten Bytes entsprechend ausgebaut sind und dass die Auswertung im LabVIEW-Programm passend ist.

### 3. Die Installation und der Start von LabVIEW

#### Die LabVIEW-Versionen

Das Programmpaket LabVIEW wird von National Instruments in der aktuellen **Version LabVIEW 2011** in verschiedenen, **kostenpflichtigen** Ausführungen angeboten (Stand: 03.2012, [www.ni.com](http://www.ni.com))

- LabVIEW Base
- LabVIEW Full
- LabVIEW Professional
- LabVIEW Developer Suite.

Daneben gibt es noch:

- Die **Studentenversion** für Schüler, Studenten und Auszubildende. Diese Version wird sehr preiswert vertrieben, entspricht in ihrem Umfang der **Full-Version**, darf allerdings nur vom zuvor erwähnten Personenkreis verwendet werden.
- Die **kostenfreie „30-Tage-Evaluierungsversion“**, die in ihrem Umfang der **Professional-Version** entspricht.  
Downloadbar unter: <http://www.ni.com/trylabview/d/>

Bei der nachfolgend beschriebenen Installation (unter Windows XP) stützen wir uns auf die „30-Tage-Evaluierungsversion“ ab, wobei die Installationen der anderen Versionen ähnlich abläuft.

#### Die Installation von LabVIEW 2011 unter Windows XP

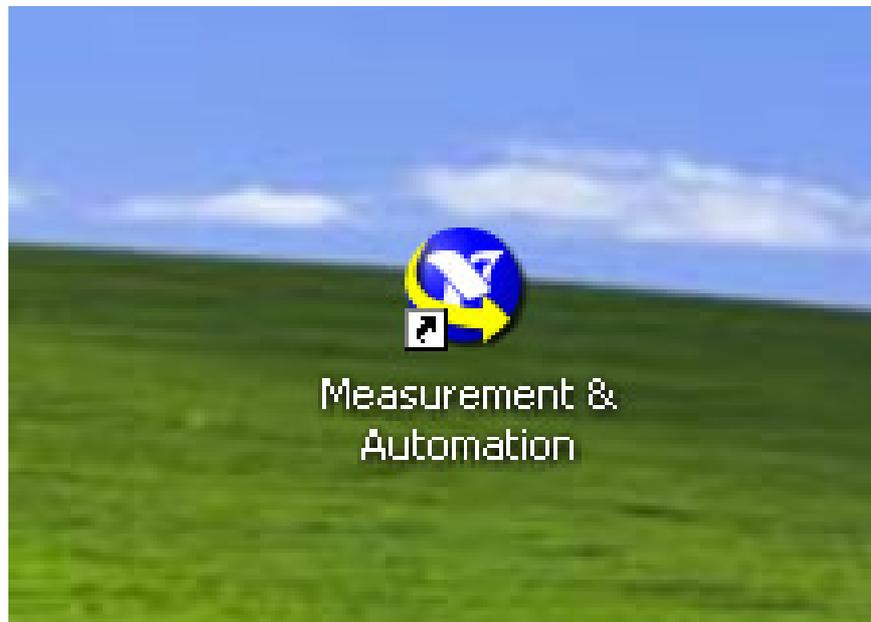
Laden Sie sich die „30-Tage-Evaluierungsversion“ aus dem Internet herunter (‘**2011LV-WinGer.exe**’) und starten Sie diese.

- Bestätigen Sie das Entpacken der einzelnen Files mit ‘OK’.
- Das Unzip-Programm wird aufgerufen; wählen Sie ‘**Unzip**’
- Nun werden die Dateien entpackt, insgesamt 638 Files.
- Nach dem Entpacken startet die Installation automatisch.

Folgen Sie einfach, wie bei Windows-Programmen gewohnt, den Installationshinweisen. Seriennummern brauchen nicht eingegeben zu werden, da es sich um die Demo-Version handelt.

Zum Schluss der Installation starten Sie den PC/LapTop neu.

LabVIEW ist nun einsatzbereit und auf dem Desktop sehen Sie **das neue Ikon von National Instruments 'Measurement & Automation'**, Abb.3.1:

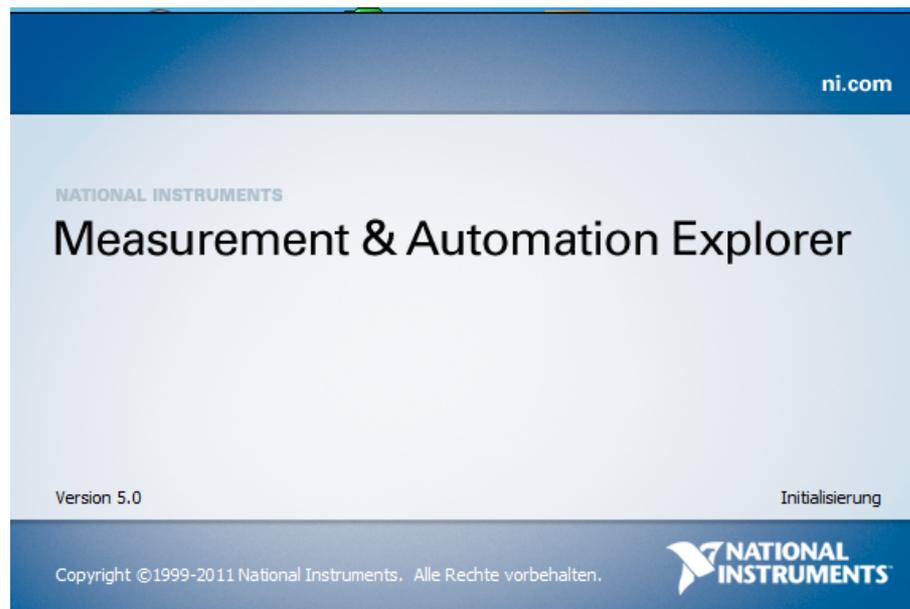


**Abb.3.1: Das Startfenster zu LabVIEW**

### **Der Start von LabVIEW**

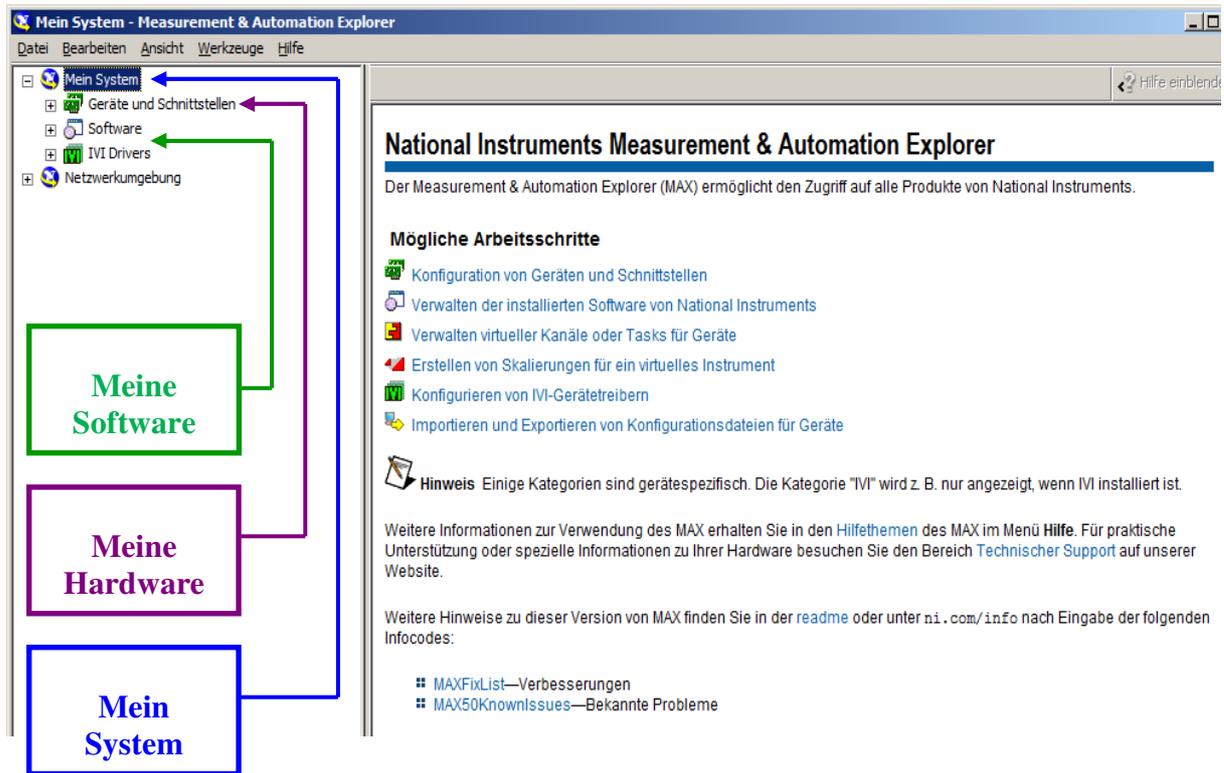
Klicken Sie zum Start von LabVIEW auf das neue Ikon, Abb.3.1.

Es öffnet sich der so genannte **'Measurement & Automation Explorer (MAX)'**, der das **zentrale Verwaltungs- und Kontroll-Zentrum** für alle National Instruments-Produkte (Hard- und Software) darstellt, **Abb.3.2:**



### **Ab.3.2: Der Start des 'Measurement & Automation Explorers MAX'**

Auf diesem zentralen Bildschirm sehen Sie auf einem Blick **alle wichtigen installierten bzw. angeschlossenen (National Instruments)Komponenten**, die der MAX auf ihrem Rechner erkannt hat, **Abb.3.3:**



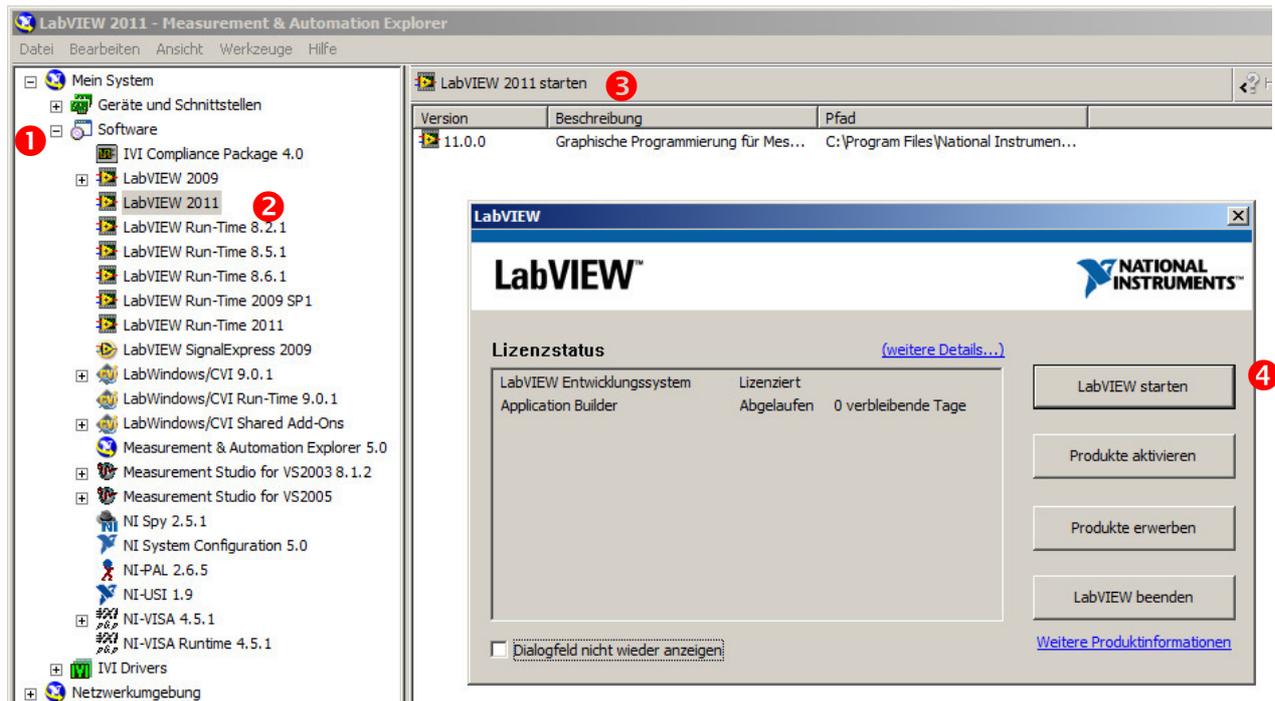
**Abb.3.3: Der MAX als zentrales Informationszentrum**

‘**Mein System**’ besteht u.a. aus:

- Der **angeschlossenen Hardware** (‘**Geräte und Schnittstellen**’): zur Zeit ist jedoch keine besondere National Instruments Hardware angeschlossen. Allerdings werden unter diesem Punkt auch die an bzw. in ihrem Rechner vorhandenen seriellen und parallelen Schnittstellen angezeigt. Daher können Sie dort feststellen, welche COMx-Schnittstellen bzw. welche USB/Seriell-Adapter (mit ihren COMx-Zuordnungen) der MAX ‘entdeckt’ hat, und die somit unter LabVIEW verfügbar sind.
- Der installierten (National Instruments) **Software**. Unter diesem Punkt ist nun LabVIEW zu finden.

Erweitern Sie daher den Menüpunkt ‘**Software**’ durch Klicken auf das ‘+’-Zeichen, **Abb.3.4**,

**1**:



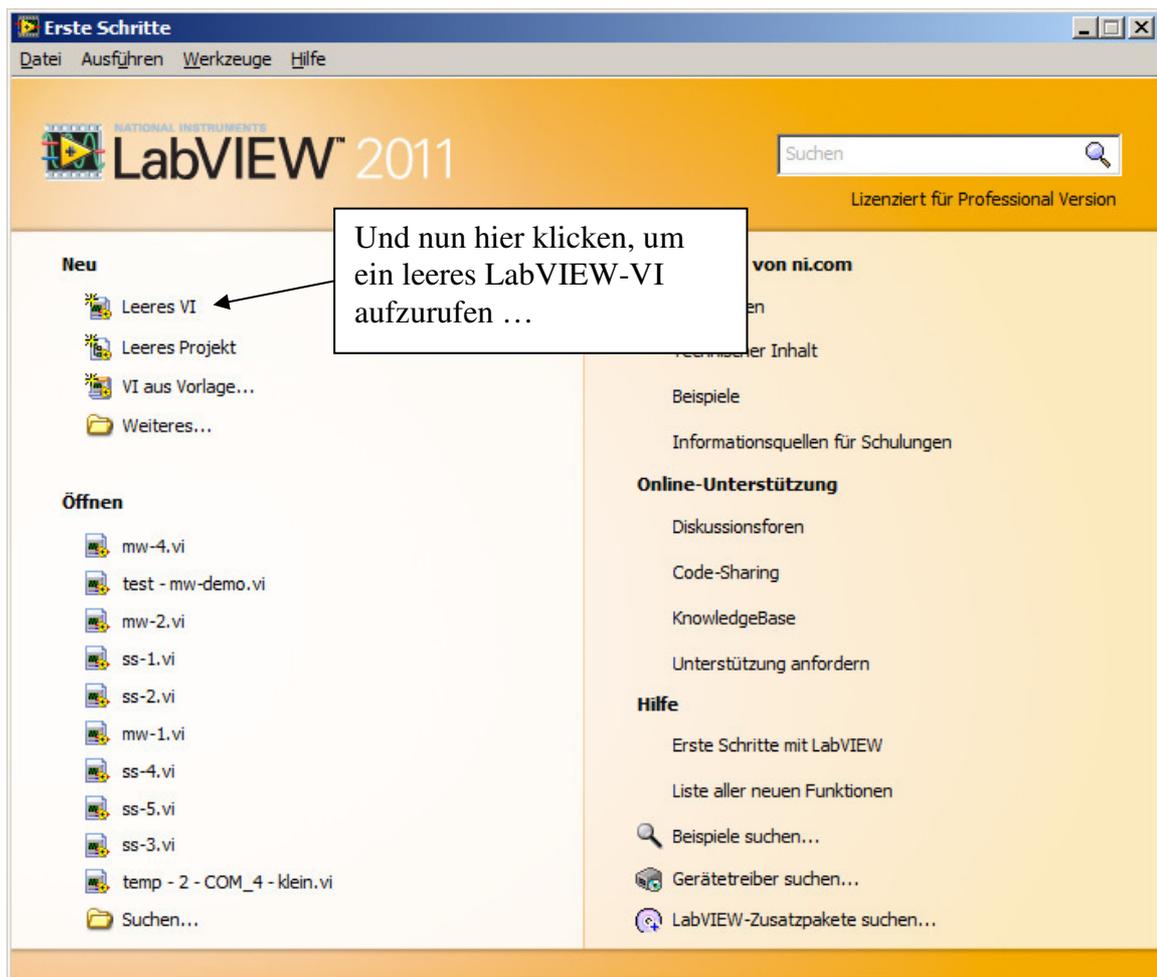
**Abb.3.4: Die Erweiterung des Menüpunktes 'Software'**

Es erscheint ein Pull-Down-Menü mit allen auf ihrem Rechner installierten Software-Komponenten von National Instruments (Hinweis: die Darstellung auf Ihrem Rechner sieht sicherlich etwas anders aus).

Klicken Sie nun auf den Menüpunkt 'LabVIEW 2011', 2: die Darstellung im rechten Teilfenster ändert sich und Sie klicken dort auf 'LabVIEW 2011 starten', 3.

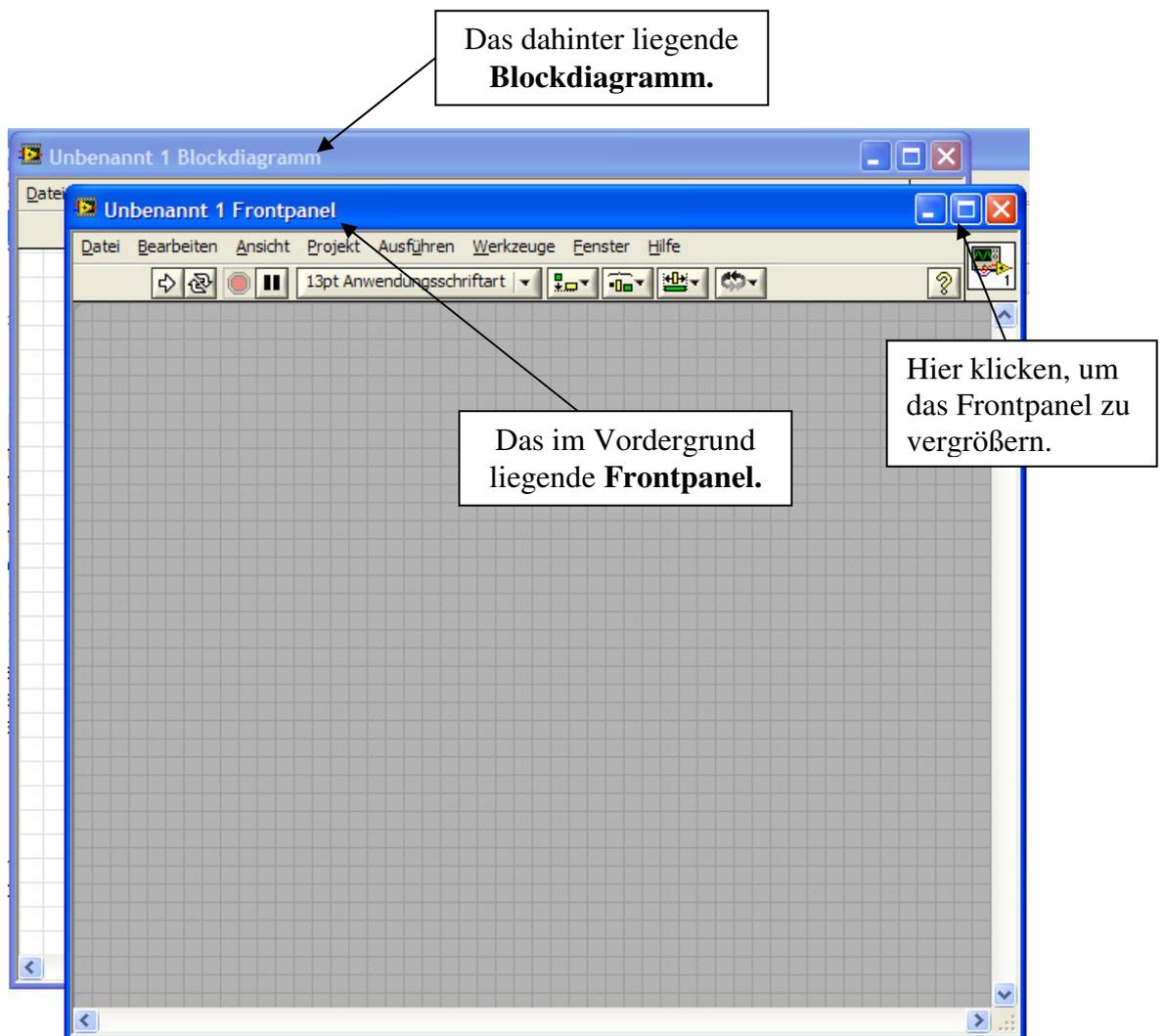
Es erscheint ein kleines neues Fenster, in dem Sie auf den Button 'LabVIEW starten' klicken, 4.

Nun endlich startet LabVIEW, Abb.3.5:



**Abb.3.5: LabVIEW wurde erfolgreich gestartet (LabVIEW-Start-Bildschirm)**

Klicken Sie nun auf **‘Leeres VI’** und ein noch leeres LabVIEW-VI ( $\equiv$  LabVIEW-Programm), bestehend aus **Frontpanel** und **Blockdiagramm** erscheint, **Abb.3.6:**



**Abb.3.6: Das noch leere LabVIEW-VI mit Frontpanel und Blockdiagramm**

Und bevor Sie nun richtig einsteigen können, wollen wir Ihnen in einer **äußerst kurzen Kompaktbeschreibung** einige wesentliche Kerneigenschaften von LabVIEW näher bringen.

#### **4. LabVIEW kompakt**

In diesem Kapitel möchten wir eine sehr knappe und kompakte Einführung in einige Grundprinzipien von LabVIEW geben.

Dieses Wissen reicht dann aus, um ein erstes eigenes LabVIEW-Programm zu entwickeln und zu testen.

Und natürlich soll das auch Spaß auf mehr machen.

Wenn das dann gelungen ist, können Sie sehr einfach weiter in die Welt von LabVIEW einsteigen und finden dazu die entsprechenden Hinweise in Kapitel 6.

## 4.1 Allgemeines

LabVIEW ist, vereinfacht ausgedrückt, zunächst ein Software-Programm-Paket für den PC/LapTop, mit dem man auf dem Monitor und mit der PC-Hardware bzw. einer entsprechenden Zusatzhardware beliebige Messgeräte selber entwickeln und betreiben kann. Der große Vorteil dabei ist, dass man sich solch ein Messinstrument fast rein graphisch, durch Erstellung und Verdrahtung von intelligenten „**Funktionsbausteinen**“ zusammen bauen kann.

Leistungsfähige Ergebnisse unter LabVIEW sind daher mit einer verblüffenden Schnelligkeit erreichbar.

Sehr treffendes Zitat eines LabVIEW-Einsteigers: „**LabVIEW ist wie Lego !**“

Das komplizierte Erlernen und das Arbeiten mit einer höheren PC-Programmiersprache ist gar nicht erforderlich.

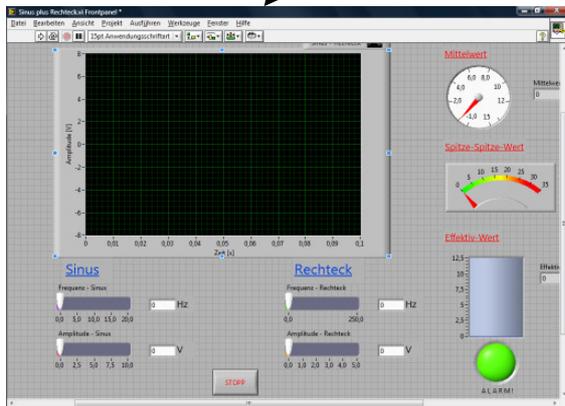
Ein Messgerät unter LabVIEW oder genauer gesagt: ein **LabVIEW-Programm** wird daher auch als „**Virtuelles Instrument**“ oder einfach als „**VI**“ bezeichnet.

### **Merke: Das Virtuelle LabVIEW-Instrument (VI)**

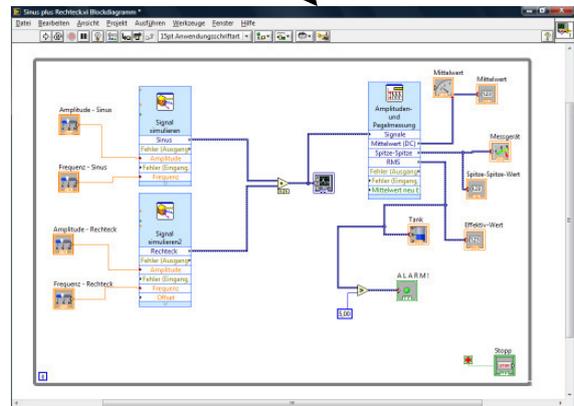
Ein **VI** besteht immer aus genau **zwei großen Kernkomponenten**, nämlich aus:

- dem **Frontpanel** oder auch kurz Panel genannt
- und aus
- dem **Blockdiagramm**, Abb.4.1.

**Das LabVIEW-Programm**  
 $\equiv$   
**LabVIEW-Virtuelles Instrument VI**



**Frontpanel**



**Blockdiagramm**

**Abb.4.1.1: Das LabVIEW-VI**

**Das Frontpanel**

bildet die **Benutzerschnittstelle** (Benutzeroberfläche, „HMI  $\equiv$  Human Maschine Interface“ bzw. „Mensch-Messgeräte-Schnittstelle“) für alle Eingaben/Einstellungen und für alle Ausgaben/Anzeigen.

Das Frontpanel ist daher das, was der Anwender später, beim Start bzw. Ablauf des LabVIEW-Programms (des LabVIEW-VIs) als Ergebnis „zu sehen bekommt“.

Wie bei einem „richtigen“ Messgerät findet man daher auf dem Monitorbild alle Arten von **Anzeige- und Bedien-Elementen**:

- Knöpfe, Taster, Regler,
- LEDs, Zeigerinstrumente, Ziffernzeigen,
- graphische Kurvendarstellungen,
- Beschriftungs- und Dekorations-Elemente,
- etc.

## Das Blockdiagramm

beinhaltet das eigentliche **LabVIEW-Programm** (den LabVIEW-Programmcode, die interne Intelligenz des Messgerätes) in Form von einzelnen **Funktionen**:

- Erfassung der Messwerte,
- Verarbeitung der Eingaben vom Frontpanel,
- Verknüpfung von Werten,
- Ansteuerung der Anzeigen auf dem Frontpanel,
- Ausgabe von Daten für die Dokumentation, Ausdruck, Speicherung,
- etc.

Die Vielfalt der bereits in LabVIEW vorhandenen Funktionen ist fast unbeschränkt, zumal der Anwender natürlich auch eigene Funktionen entwickeln bzw. zusammenstellen kann.

### **Merke: Die wesentlichste und bestechenste Eigenschaft von LabVIEW**

Man stellt sich das VI komplett auf graphischem Wege, d.h.

- unter Verwendung von **graphischen Symbolen** für das Frontpanel
- und
- durch die Verknüpfung von **graphischen Funktionsblöcken** bei der Erstellung des Blockdiagramms

zusammen.

(Die sich hinter all dem versteckende Programmiersprache heißt 'G', und wurde von NI eigens für LabVIEW entwickelt. Aber damit kommt der Anwender eigentlich gar nicht in Berührung.)

Da die Erstellung des Wunsch-Messgerätes jetzt nun rein Software-mäßig erfolgt, lassen sich natürlich unzählige Messinstrumente der verschiedensten Art und mit den unterschiedlichsten Eigenschaften, für die unterschiedlichsten Aufgaben entwickeln bzw. per Mausklick zusammenstellen.

Und **die Basis** dabei bleibt immer die Gleiche: der PC mit LabVIEW als einzige, alles umfassende Entwicklungsoberfläche und natürlich die benötigte eigene Messwerterfassungshardware bzw. ganz allgemein: das selbst entwickelte Mess-, Steuerungs- oder Automatisierungssystem, das an LabVIEW angekoppelt wird.

Nachfolgend werden wir nun zunächst allgemein einige Konzepte und Lösungsvorschläge zur Erstellung von Blockdiagramm und Frontpanel vorstellen und das dann im Kapitel 5 für die Realisierung unseres ersten Projektes anwenden.

## 4.2 Der Entwurf des Blockdiagramms

Starten Sie nun LabVIEW und öffnen Sie ein leeres VI so dass das „Grundbild“ eines VIs gemäß der Abb.3.6 erscheint.

Schalten Sie nun um auf das Blockdiagramm-Fenster und vergrößern Sie dieses auf die volle Bildschirmgröße.

### **Merke: Die Shortcuts**

Das schnelle und effektive Arbeiten mit LabVIEW wird durch so genannte ‘**Shortcuts**’ unterstützt.

Das sind spezielle Tastenkombinationen, die bestimmte Funktionen bzw. Funktionsabläufe bewirken.

Solche Tastendrucke sind oft schneller ausgeführt als die Auswahl der entsprechenden Aktionen mit der Maus: Menü bzw. Untermenü auswählen und bestimmte Felder anklicken.

Die Shortcuts sind oft als Kombination der ‘**Strg-Taste**’ mit einer weiteren Taste zusammengesetzt.

Eine Übersicht über einige wichtige Shortcut-Kombinationen, die Sie ab jetzt täglich benutzen werden, finden Sie im Anhang ‘**Wichtige Shortcuts**’.

Weitere sind in der Dokumentation zu LabVIEW aufgeführt.

### **Tipp: Umschalten Blockdiagramm <---> Frontpanel**

Das Umschalten zwischen Blockdiagramm- und Frontpanel-Fenster lässt sich sehr einfach mit dem **Shortcut** ‘**Strg+E**’ durchführen.

### **Die Funktionspaletten des Blockdiagramms**

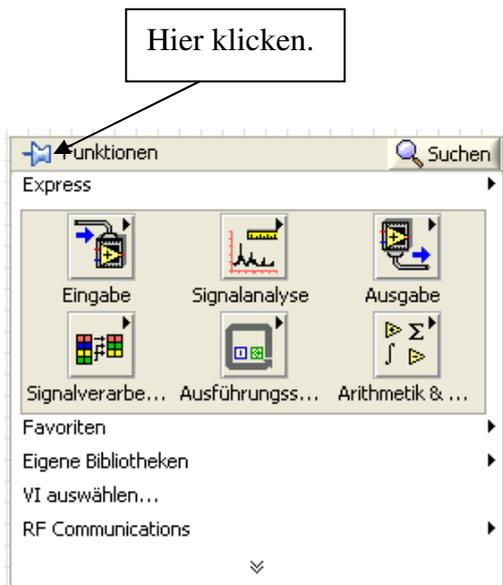
Wie bereits erwähnt, steckt die eigentliche Intelligenz eines LabVIEW-VIs im Blockdiagramm und wird durch die dort verfügbaren Funktionen „zusammengebaut“.

### **Tipp: Die Funktionen auf dem Blockdiagramm**

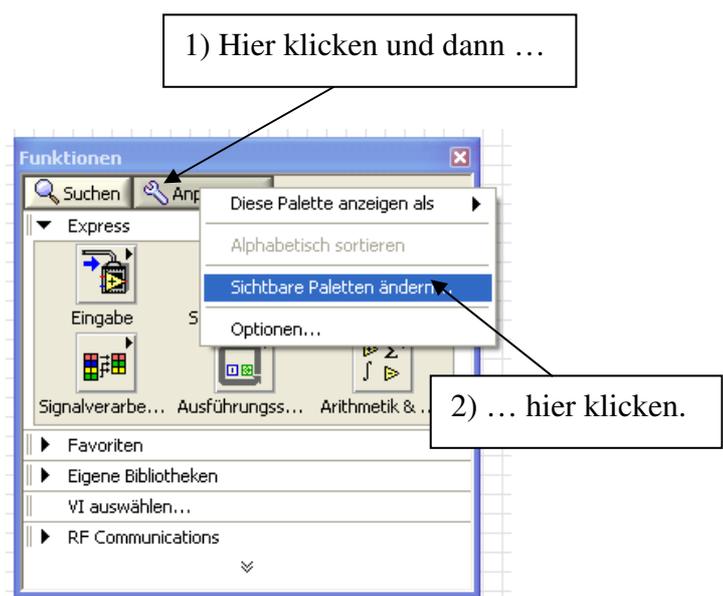
Zu den verfügbaren Funktionen gelangen Sie, in dem Sie mit der rechten Maustaste auf eine freie Stelle im Blockdiagramm klicken.

Alle hier nun verfügbaren Funktionen sind in so genannten **Funktionspaletten** zusammengefasst, deren für uns zunächst Wichtigste die Palette ‘**Programmierung**’ ist.

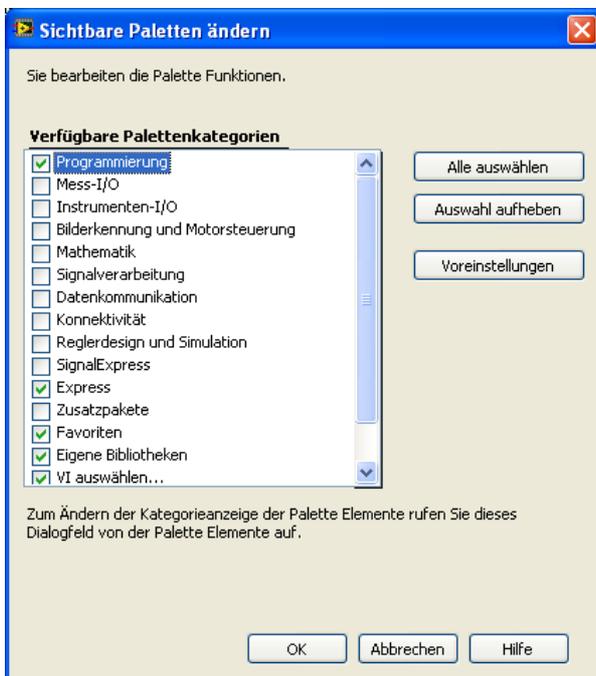
Diese Zusammenstellung müssen wir uns als Erstes einmal **dauerhaft sichtbar machen**, **Abb.4.2.1:**



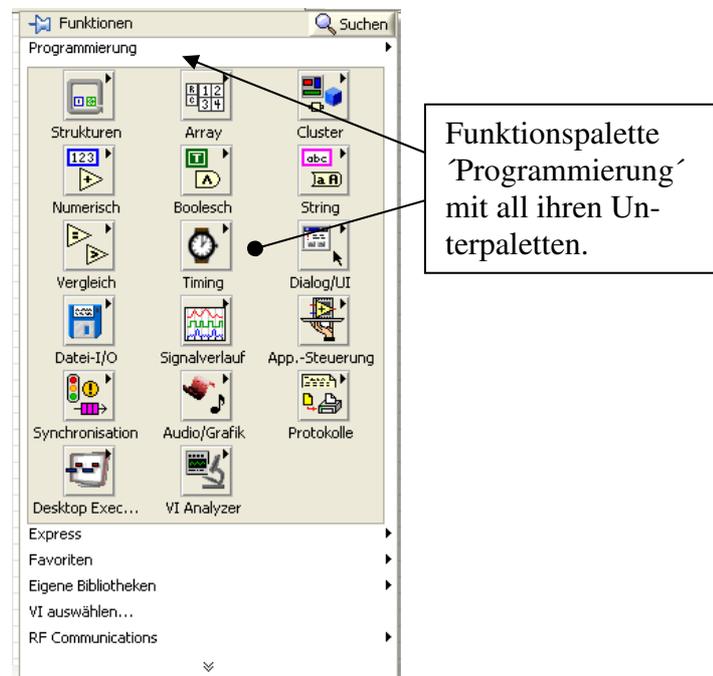
**Bild 1):** Mit der rechten Maustaste auf eine freie Stelle im Blockdiagramm klicken und dann auf die Heftzwecke klicken.



**Bild 2):** Auf den Button 'Anpassen' klicken und dann auf 'Sichtbare Paletten ändern ...' klicken.



**Bild 3):** Den Eintrag 'Programmierung' aktivieren und mit 'OK' bestätigen.



**Bild 4):** Die Funktionspalette 'Programmierung' mit all ihren Unterpaleetten ist nun dauerhaft sichtbar.

**Abb.4.2.1: Die Sichtbarmachung der Funktionspalette 'Programmierung'**

- 1) Klicken Sie dazu als erstes mit der rechten Maustaste auf eine freie Stelle im Blockdiagramm.  
Es erscheint ein Menü mit allen zur Zeit sichtbaren Funktionspaletten, **Bild 1**).  
Leider ist die Palette 'Programmierung' hier zur Zeit noch nicht enthalten und diese muß daher jetzt erst 'sichtbar' gemacht werden.  
Klicken Sie dazu auf das **'Heftzwecken-Symbol'** oben links im Fenster.
- 2) **Bild 2**): Das Fenster mit den Funktionspaletten wird nun auf dem Blockdiagramm fixiert.  
Klicken Sie hier nun auf den Button **'Anpassen'**.  
Es erscheint ein weiteres Auswahlmenü in dem Sie auf **'Sichtbare Paletten ändern ...'** klicken.
- 3) **Bild 3**): In der nun erscheinenden Liste der möglichen Funktionspaletten markieren Sie den Eintrag 'Programmierung' und klicken dann auf 'OK'.
- 4) **Bild 4**): Ab jetzt erscheint bei jedem neuen Aufruf der Funktionspaletten ( $\equiv$  Klick mit rechter Maustaste auf eine freie Stelle im Blockdiagramm) die Funktionspalette 'Programmierung' an erster Stelle, bereits geöffnet mit all ihren Unterpaletten.

Schließen Sie nun abschließend die Funktionspalette.

Ab jetzt werden wir den „Weg“ zu einer gewünschten Funktion für das Blockdiagramm wie folgt angeben:

**Beispiel:**

Die „Wegbeschreibung“ hin zur numerischen Funktion „Addition“:

***BD\Programmierung\Numerisch\Addieren***

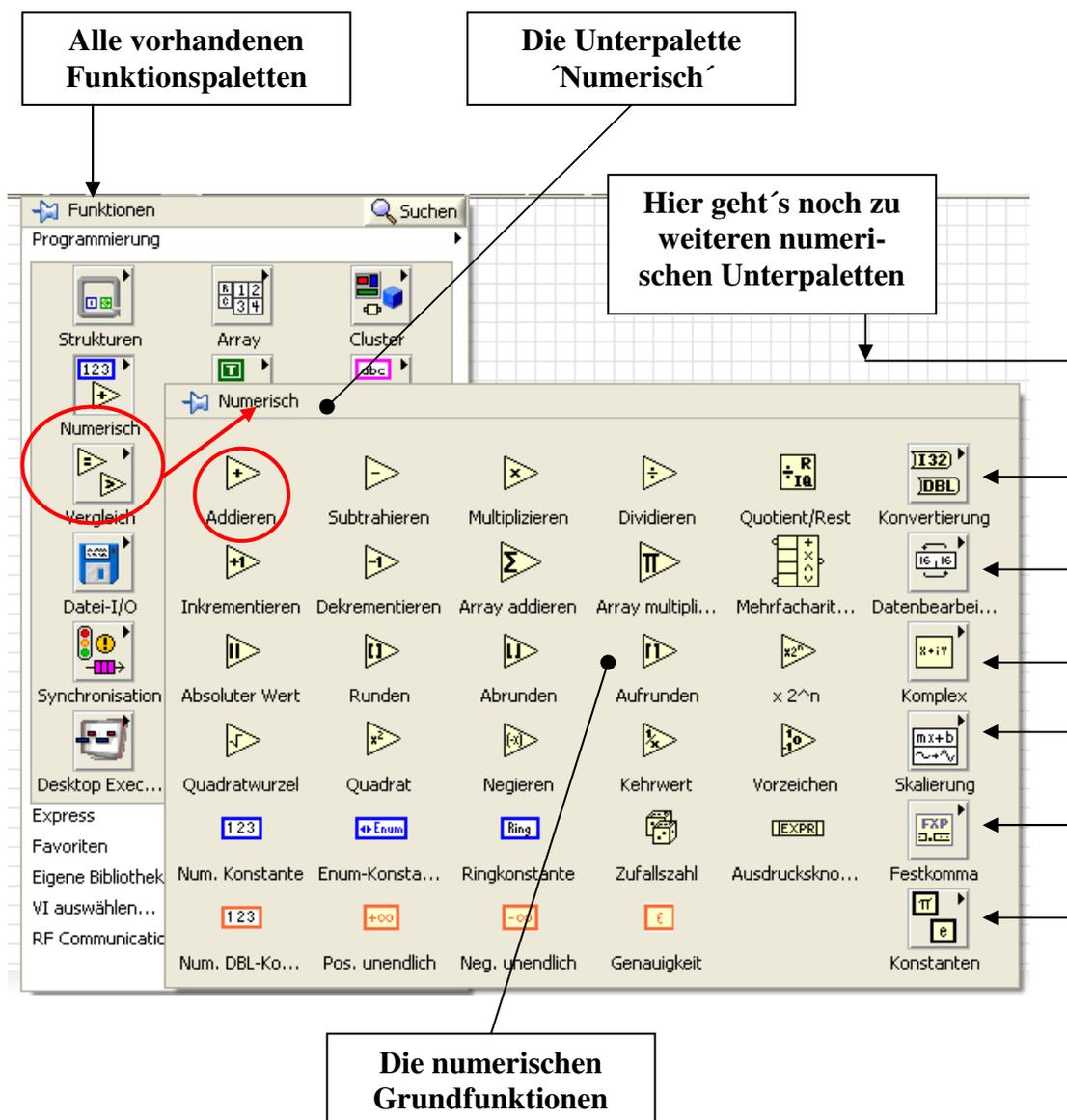
Was nun folgendes bedeutet:

- **BD**  $\equiv$  Arbeiten auf dem Blockdiagramm , d.h. Aufruf des Blockdiagramms und mit der rechten Maustaste auf eine freie Stelle des Blockdiagramms klicken.  
Es öffnet sich die Sammlung der ***Funktionspaletten***.
- **Programmierung**  $\equiv$  Auswahl der Palette 'Programmierung',  
bzw. diese Palette ist hier bereits ausgewählt und geöffnet.
- **Numerisch**  $\equiv$  Anschließend fährt man mit der Maus über 'Numerisch' und

nun erscheinen endgültig alle verfügbaren Funktionen aus dieser Unterpalette, eben die numerischen Funktionen.

- **Addieren**  $\equiv$  Nun klickt man mit der linken Maustaste auf die gewünschte Funktion. Die Funktionspaletten schließen sich und auf dem Blockdiagramm erscheint der Mauszeiger mit der angehängten Funktion, die nun beliebig auf dem Blockdiagramm platziert werden kann.

Das Ergebnis dieser Auswahl sehen Sie noch einmal zusammengefasst in der **Abb.4.2.2**:



**Abb.4.2.2: Die (Unter)Funktionspalette 'Numerisch' mit der Funktion 'Addieren'**

### Das Einschalten des Rasters

Als nächstens stellen wir das Zeichenraster für das Blockdiagramm-Arbeitsblatt ein:

Wählen Sie dazu aus der oberen Menüleiste aus:

Werkzeuge\Optionen\Blockdiagramm

Und dann: Bei „Blockdiagrammgitter anzeigen“ einen Haken setzen und mit ‚OK‘ bestätigen.

Das hat jetzt den folgenden Vorteil:

Wenn Sie jetzt ein LabVIEW-VI entwerfen, so sehen Sie in der Entwurfsphase sowohl auf dem Blockdiagramm als auch auf dem Frontpanel das Zeichenraster.

Läuft das VI dagegen ab, so ist das Raster verschwunden und Sie können so schon optisch erkennen, in welchem Zustand sich das VI befindet.

### Die Verdrahtung der Funktionen des Blockdiagramms

Die Funktionen bzw. Funktionsblöcke auf dem Blockdiagramm werden auch ganz allgemein als „**Knoten**“ (= **programmausführendes Element**) bezeichnet.

Jeder Knoten hat Ein- und/oder Ausgänge, die graphisch auf dem Blockdiagramm mit „**Verbindungsdrähten**“ untereinander verbunden werden.

Die Anschlusspunkte für die Drähte nennt man „**Terminals**“.

Fährt man mit der Maus über einen Knoten (über einen Funktionsblock), so werden die Terminals des Knotens angezeigt, bei einigen Funktionen auch die Datentransferichtung des Terminals: Daten in den Funktionsblock rein oder aus dem Funktionsblock heraus.

Je nach dem, welche Art von Daten:

- ganze Zahlen,
- float-Zahlen,
- Strings,
- Arrays,
- etc.

über die Drähte transportiert werden, haben diese Verbindungen eine andere Farbe und eine andere Form.

In den meisten Fällen „erkennt“ LabVIEW automatisch den Datentyp, der transportiert werden soll, und wählt von sich aus schon den richtigen, passenden Verbindungsdraht.

### Die Verdrahtung

Platzieren Sie nun auf dem Blockdiagramm einmal eine Additions- und eine Subtraktionsfunktion:

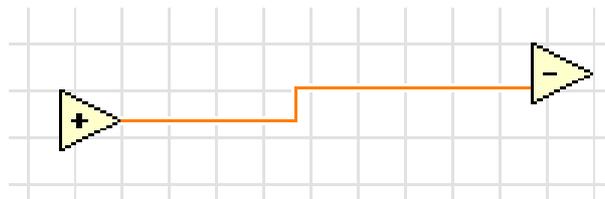
*BD\Programmierung\Numerisch\Addieren**BD\Programmierung\Numerisch\Subtrahieren*

Fahren Sie nun mit der Maus auf das Additionssymbol: die zugehörigen Anschluss terminals („kleine orange Kreise“) erscheinen.

Fahren Sie nun mit der Maus auf den Ausgangsanschluss der Additionsfunktion: Sie werden feststellen, dass sich das Kreuz-Maussymbol plötzlich in **eine kleine Drahtrolle** verwandelt: **die Maus ist nun zum Verdrahtungswerkzeug geworden.**

Halten Sie nun die linke Maustaste gedrückt und ziehen Sie den Draht nach rechts vom Additions-Symbol weg. Lassen Sie die Maustaste los, so wird ein Haltepunkt gesetzt und Sie können den Draht „um die Ecke“ legen. Der Draht wird jetzt noch als gestrichelte schwarze Linie dargestellt.

Führen Sie den Draht nun auf einem „schönen“ Weg zum unteren Eingangsanschluss der Subtraktionsfunktion. Sobald Sie diesen Punkt erreicht haben, lassen Sie die linke Maustaste los, **Abb.4.2.3:**



**Abb.4.2.3: Die erste Verbindung**

Der Draht wird jetzt zu einer durchgezogenen, **orangen, dünnen Linie**. So kennzeichnet LabVIEW eine Verbindungslinie, über die einfache numerische Daten transportiert werden (genau genommen sind es 64 Bit große reelle Zahlen, die über diesen Draht transportiert werden).

**Tipp: Einen Draht an einen anderen Draht anschließen**

Wenn Sie jetzt mit dem Mauszeiger **in die Nähe des bereits verlegten Drahtes kommen**, verwandelt sich das Maussymbol wieder in die Drahtrolle und durch Klicken mit der linken Maustaste können Sie dort einen Verbindungs(Kreuzungs)punkt setzen und von dort eine weitere neue Leitung verlegen.

**Tipp: Verdrahtungen ändern**

Sie können jederzeit die Verlegung des Drahtes im Blockdiagramm ändern oder verbessern. Dazu haben Sie zunächst folgende Möglichkeiten:

Bewegen Sie die Maus in die Nähe des zu verändernden Drahtes bis der Mauszeiger die Form eines schräg nach oben gerichteten Pfeils hat.

Dann drücken Sie die rechte Maustaste und ein **Pull-Down-Menü** erscheint. Die oberen drei Menü-Punkte beziehen sich auf die Möglichkeiten, den gerade ausgewählten Verbindungsdraht zu manipulieren:

**Verdrahtung bereinigen:** Wenn Sie im Laufe Ihrer Arbeit einmal ein Draht „total kreuz und quer“ auf dem Arbeitsblatt verlegt haben, so wählen Sie diesen Menüpunkt aus und LabVIEW optimiert für Sie automatisch die Verlegung des Drahtes. Meistens sieht diese Lösung dann optisch besser aus (Verbindungsmäßig ändert sich natürlich nichts !).

**Verzweigung erstellen:** Klicken Sie hier drauf, so wird an dieser Draht-Stelle eine neue Verzweigung erstellt, die Drahtrolle erscheint wieder und Sie können ab hier dann eine neue Verbindung verlegen.

**Verzweigung löschen:** Über diesen Menüpunkt können Sie den verlegten, vorher markierten, Draht wieder löschen.

Die **Löschung eines Drahtes** ist aber auch einfacher möglich:

Sie fahren mit der Maus auf den Draht, bis der Mauszeiger wieder die Form des Pfeils hat. Dann klicken Sie den Draht mit der linken Maustaste an, der Draht erscheint gestrichelt (der Draht wird markiert).

Und nun drücken Sie einfach auf die Lösch-Taste (‘Entf’-Taste) auf der PC-Tastatur und der Draht ist weg.

**Und nun ganz wichtig:**

Meisten bleiben nach Löschaktionen noch einige „Rest-Drahtstücke“ übrig, die jetzt „lose in der Luft herum hängen“.

Diese brauchen Sie jetzt nicht mühsam einzeln nach einander zu löschen, sondern hierzu gibt es den **Shortcut ‘Strg+B’**, mit dem LabVIEW automatisch und selbständig alle „irgendwo lose im Blockdiagramm herumhängenden überflüssigen Drähte“ entfernt.

Das ist sehr oft eine äußerst hilfreiche Funktion, um das Blockdiagramm schnell einmal aufzuräumen, d.h. es von losen (fehlerhaften) Verbindungen zu bereinigen.

## **Die verschiedenen Cursor-Formen**

Gerade am Anfang des Arbeitens mit LabVIEW schenkt man der Form bzw. der Funktion des Cursors auf dem Blockdiagramm/Frontpanel zunächst nicht allzu viel Bedeutung, aber insbesondere hierin steckt jede Menge Potential:

### **Cursor als „Kreuz“**

Befindet sich der Cursor außerhalb einer Funktion, so hat der die „normale Arbeitsform“ und wird als Kreuz dargestellt.

Führt man den Cursor nun über eine Funktion, so wechselt er, je nach aktueller Position über der Funktion, seine Form:

### **Cursor als „Hand mit ausgestrecktem Zeigefinger“**

An einigen Stellen von bestimmten Funktionsblöcken bzw. Funktionen (hauptsächlich bei Bedienelementen und Konstanten) hat der Cursor die Form einer kleinen Hand (mit ausgestrecktem Zeigefinger) und das zeigt an, das hier etwas eingestellt bzw. ein bestimmter Wert aus einem erscheinenden Pull-Down-Menü ausgewählt werden kann.

Das gilt insbesondere auch für Elemente auf dem Frontpanel.

**Allgemein** halten wir daher fest: hat der Cursor die „**Hand-Form mit ausgestrecktem Zeigefinger**“, so lassen sich bei den Anzeige- und Bedien-Elementen auf dem Blockdiagramm oder auf dem Frontpanel (Vorgabe)Werte einstellen bzw. Schalter umschalten bzw. andere, für das Element typische, Einstellungen durchführen, die dann beibehalten werden, wenn das LabVIEW-Programm gestartet wird ( $\equiv$  Einstellung von Startwerten bzw. Anfangsbedingungen).

### **Cursor als „Pfeil“**

An bestimmten Positionen über der Funktion, nimmt der Cursor die Form eines kleinen Pfeils an.

Klickt man nun auf die linke Maustaste, so erscheint ein gestrichelter Rahmen um diese Funktion, **die Funktion ist markiert**.

Außerhalb des markierten Bereichs hat der Cursor wieder die Kreuz-Form und durch Klicken mit der linken Maustaste auf eine Stelle außerhalb des Elements verschwindet die Markierung des Elements.

Mit markierten Funktionen kann man nun einiges machen:

## **Markierte Funktionen**

### **Verschieben:**

Halten Sie die linke Maustaste gedrückt (Cursor hat Pfeil-Form), so können Sie die markierte Funktion beliebig auf dem Blockdiagramm verschieben.

### **Löschen**

Durch Druck auf die 'Entf'-Taste auf der PC-Tastatur wird die markierte Funktion aus dem Blockdiagramm gelöscht.

### **Kopieren**

Auch das Kopieren von Funktionen ist sehr einfach machbar: wenn die entsprechende Funktion markiert ist, fahren Sie mit dem Mauszeiger darüber, so dass dieser die Pfeilform annimmt.

Halten Sie dann die linke Maustaste gedrückt und drücken Sie gleichzeitig die **‘Strg’-Taste** auf der Tastatur.

Es wird nun eine Kopie der Funktion angefertigt, die jetzt mit der Maus irgendwohin verschoben werden kann.

Am Zielort lassen Sie einfach die Maus- und die **‘Strg’-Taste** los und die Kopie wird platziert.

### **Tipp: Der LabVIEW-Cursor auf dem Frontpanel**

Auch auf dem Frontpanel eines LabVIEW-Programms, das wir im nächsten Kapitel besprechen werden, hat der Cursor verschiedene Formen und damit verschiedene Bedeutungen

### **Alternative Art zur Durchführung der Verschiebung einer Funktion**

Es gibt unter LabVIEW noch eine zweite Art, um eine Funktion zu markieren und sie dann zu verschieben:

Wenn Sie den Cursor **außerhalb** der Funktion platzieren, so hat er ja, wie gesehen, die Form eines kleinen Kreuzes.

Halten Sie nun die linke Maustaste gedrückt und umfahren Sie mit dem Cursor die Funktion, so erscheint der bekannte gestrichelte Markierungsrahmen.

Lassen Sie dann die Maustaste wieder los, so ist die Funktion, incl. eines ev. vorhandenen Textfeldes, markiert.

Der Rest, die Verschiebung, läuft dann so ab, wie zuvor beschrieben.

### **Merke: Die alternative Markierung von LabVIEW-Funktionen**

Es reicht hierbei sogar aus, dass Sie mit dem gestrichelten Rahmen die Funktion nur an einer Seite/Ecke **berühren**: LabVIEW markiert dann automatisch die gesamte berührte Funktion. Das ist dann sehr hilfreich, wenn mehrere Funktionen auf dem Blockdiagramm nahe beieinander angeordnet sind und man sie nur schlecht voneinander trennen kann: die einfache Berührung der gewünschten Funktion mit dem gestrichelten Rahmen reicht zu deren Markierung aus.

Das entsprechende gilt dann auch später, wenn wir einzelne Elemente im Frontpanel markieren bzw. auswählen wollen.

### **Cursor als „Drahtrolle“**

Bewegen Sie den Cursor direkt auf ein Anschlussterminal einer Funktion, so verwandelt sich das Kreuz-Maussymbol in eine kleine Drahtrolle – die Maus ist nun zum **Verbindungswerkzeug** geworden und es können Verbindungsleitungen gezogen werden: linke Maustaste gedrückt halten und Draht verlegen.

## Rückgängigmachen

Haben Sie sich bei den Änderungen (z.B. bei Verschiebungen) vertan oder möchten Sie diese wieder rückgängig machen, so ist das auch kein Problem:

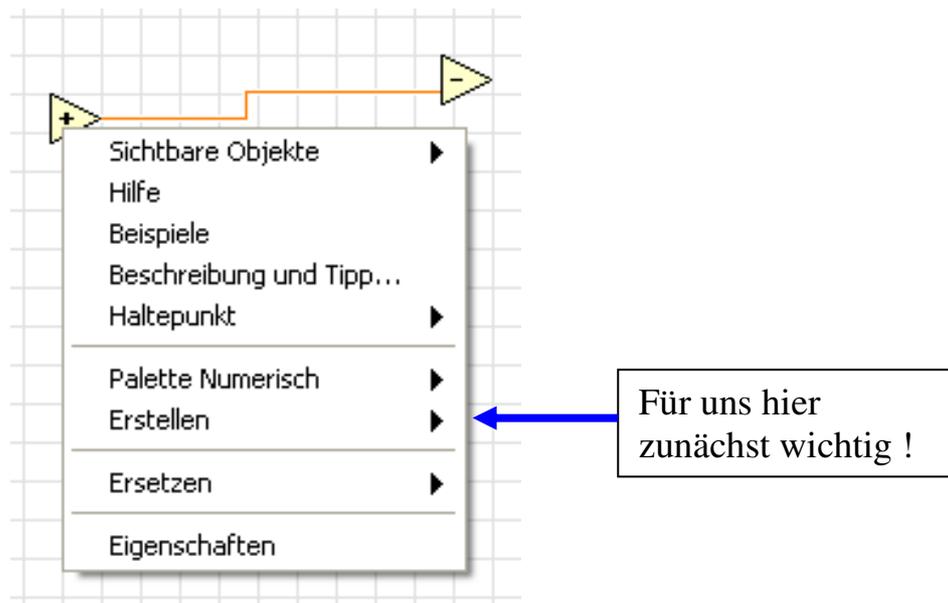
### Merke: Das Rückgängigmachen von Änderungen

Alle zuvor gemachten Änderungen (egal welcher Art) können Sie natürlich auch jeder Zeit wieder rückgängig machen und so den alten Ursprungszustand wieder herstellen.

Betätigen Sie dazu einfach den **Shortcut 'Strg+Z'**.

## Das Kontextmenü

Klicken Sie nun einmal mit der rechten Maustaste auf das Additions-Symbol: es erscheint das so genannte „**Kontext-Menü**“ zu diesem Funktionsblock, **Abb.4.2.4**:



**Abb.4.2.4: Das Kontext-Menü zu einem Funktionsblock**

### Wichtig: Das Kontext-Menü

In dieser Liste, im **Kontext-Menü**, sind alle Punkte bzw. Eigenschaften aufgeführt, die der Anwender in Bezug auf diesen Funktionsblock einstellen bzw. verändern kann.

Über dieses Kontext-Menü können also wesentlich „Dinge“ zu diesem Funktionsblock festgelegt werden.

Zu

- jedem **Funktionsblock des Blockdiagramms**,

- zu jedem **Element auf dem Frontpanel**,
- ja sogar zu jedem **Anschlussterminal eines Funktionsblocks**

gibt es unter LabVIEW solch ein individuelles Kontext-Menü, das immer dann aufspringt, wenn man **mit der rechten Maustaste** auf den Block bzw. auf das Element bzw. auf den Anschluss klickt.

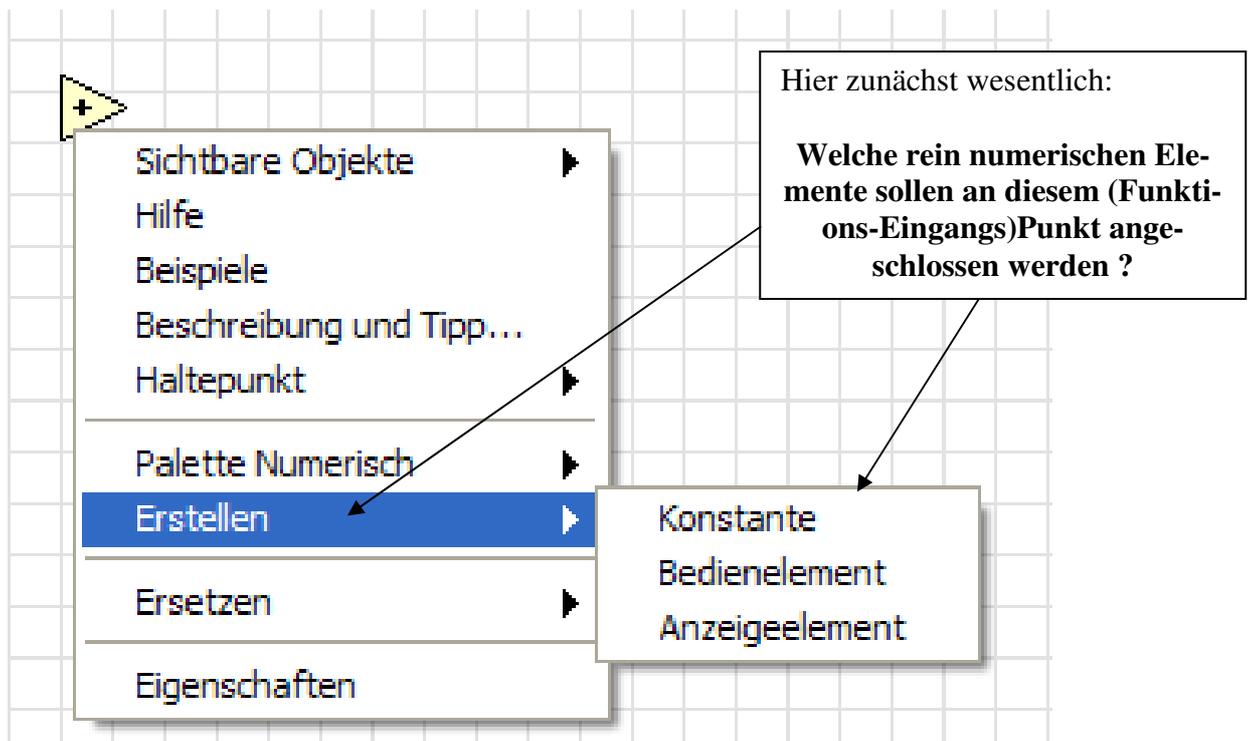
Je nach Art der Funktion bzw. Anschlusses kann dieses Kontextmenü unterschiedlich umfangreich ausfallen.

Für uns wichtig ist hier zunächst der Punkt **Erstellen**, denn damit können via Frontpanel Werte in eine Funktion eingespeist bzw. von einer Funktion ausgegeben werden.

### [Anzeigeelemente, Bedienelemente und Konstanten an Funktionen anschließen](#)

Wenn Sie nun den Punkt **Erstellen** im Kontextmenü zu einem **Funktionsanschluss (Eingangs- oder Ausgang)** anwählen, so erscheint ein weiteres kleines Auswahlfenster, in dem Sie festlegen können, was nun an diesem Anschluss angeschlossen werden soll.

Wir betrachten hierzu einmal einen der Eingangsanschlüsse der Additionsfunktion, **Abb.4.2.5**:



### Abb.4.2.5: Das Kontext-Menü zu einem (Funktions-Eingangs)Punkt und das Untermenü 'Erstellen'

Anschließend sind also:

- Konstante
- (einfaches rein numerisches) Bedienelement
- (einfaches rein numerisches) Anzeigeelement

#### **Konstante**

Diesen Punkt wählen Sie, wenn ein fester (konstanter) Wert (Summand) hinzuaddiert werden soll.

Dieser Wert wird dann in einem kleinen Kästchen auf dem Blockdiagramm dargestellt und kann dort dann ebenfalls im Wert verändert werden.

Gleichzeitig ist diese Konstante auch mit dem Eingang der Additionsfunktion verbunden. Auf dem Frontpanel erscheint hierzu allerdings dann kein besonderes Element, also keine Darstellung der Konstanten.

#### **Bedienelement**

Dieses Element wählen Sie aus, wenn Sie über das Frontpanel eine variable Eingabe für den Summanden machen wollen, d.h. hierdurch wird auf dem Frontpanel **automatisch** von LabVIEW ein rein **numerisches Bedienelement** angelegt und dieses dann im Blockdiagramm **automatisch** mit dem Eingang der Additionsfunktion verbunden.

#### **Anzeigeelement**

Durch Auswahl dieses Punktes wird versucht, ein rein numerisches **Anzeigeelement** an den **Eingang** der Additionsfunktion anzuschließen.

Das aber macht nicht viel Sinn: man kann eine Anzeige nicht als „Einspeiseelement“ für einen Eingang benutzen !

Und das merkt LabVIEW natürlich auch: es wird zwar sowohl auf dem Frontpanel als auch auf dem Blockdiagramm ein Anzeigeelement eingefügt, aber nicht mit dem Eingang der Additionsfunktion verbunden, da dieses unsinnig wäre.

Das so erzeugte Anzeigeelement steht also „völlig alleine“ (völlig unverbunden) da.

### **Merke: Der vereinfachte Anschluss von rein numerischen Anzeigeelementen und Bedienelementen**

An jeden Eingang und an jeden Ausgang von LabVIEW-Funktionen können im Blockdiagramm über das Kontext-Menü zu diesem Anschlusspunkt (über den Untermenüpunkt 'Erstellen') sehr einfach **rein numerische** Anzeige-, Bedienelemente und Konstanten angeschlossen werden.

LabVIEW prüft selbstständig nach, ob die Auswahl für diesen Anschlusspunkt sinnvoll ist oder nicht.

Bei einer gültigen Auswahl verbindet LabVIEW bereits automatisch das numerische Anzeige- bzw. Bedienelement mit dem gewünschten Aus- bzw. Eingangspunkt der Funktion. Ist die getroffene Wahl dagegen nicht sinnvoll, so wird zwar das Anzeige- bzw. Bedienelement erzeugt, aber nicht mit dem Funktionsaus- bzw. -eingang verbunden. Parallel dazu wird das Anzeige- bzw. Bedienelement bereits auf dem Frontpanel dargestellt (eingetragen). Hier müssen die Elemente dann i.a. nur noch „schön“ angeordnet werden.

### Übung:

Schließen Sie nun an die beiden Eingänge der Additionsfunktion zwei Bedienelemente an.

### Tipp: Das Umbenennen von Anzeige- und Bedienelementen

Wenn Sie den Namen von Anzeige- oder Bedienelementen ändern wollen, so klicken Sie einfach zweimal auf das Namensfeld, bis dieses schwarz hinterlegt ist. Dann können Sie den Namen des Elementes ändern.

### Übung:

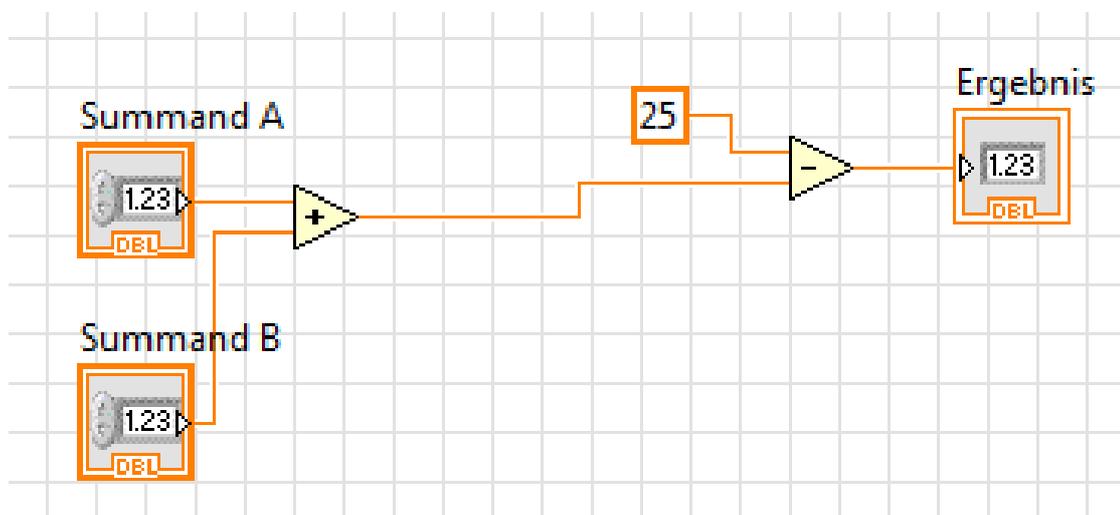
Nennen Sie nun die beiden Eingänge der Additionsfunktion um in 'Summand A' und 'Summand B'.

### Übung:

Schließen Sie am noch offenen zweiten Eingang der Subtraktionsfunktion eine Konstante mit dem Wert 25 an.

Schließen Sie am Ausgang der Subtraktionsfunktion ein Anzeigeelement an, das Sie 'Ergebnis' nennen.

Und fertig ist das erste lauffähige VI, das ungefähr so aussehen sollte, **Abb.4.2.6:**



**Abb.4.2.6: Das erste VI ('VI – 1.vi')**

Speichern Sie dieses VI unter dem Namen 'VI – 1.vi' in einem (neuen) Verzeichnis Ihrer Wahl ab.

### Tipp: Das Abspeichern

Das erstmalige Abspeichern geschieht bei Windows-Programmen wie gewohnt. Auswahl in der oberen Menü-Leiste:

**'Datei\Speichern unter ...'....**

Alle weiteren Abspeicheraktionen können dann mit dem Shortcut 'Strg+S' durchgeführt werden.

### Die Kontexthilfe

Zu jedem Element (Funktion auf dem Blockdiagramm, Element auf dem Frontpanel, ...) gibt es unter LabVIEW eine integrierte Kurzbeschreibung und einen weiteren hinterlegten ausführlichen Hilfetext, die sogenannte 'Kontexthilfe'.

### Wichtig: Die Kontexthilfe, 'Strg+H'

Aufgerufen wird diese Kontexthilfe durch den Shortcut 'Strg+H', sowohl auf dem Frontpanel als auch auf dem Blockdiagramm.

Es öffnet sich nun ein kleines weiteres Fenster, das so genannte **Kontexthilfe-Fenster**, **Abb.4.2.7**.

In diesem Fenster wird nun beschrieben, wie ein Element (eine Funktion) heißt und was es (sie) eigentlich macht.

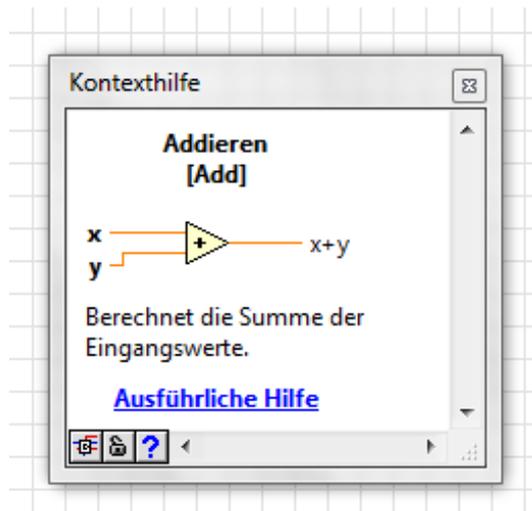
**Wenn Sie nun mit dem Cursor (mit der Maus) über ein Element auf dem Frontpanel bzw. über einen Funktionsblock auf dem Blockdiagramm fahren, so erscheint in dem Kontexthilfe-Fenster sofort eine Kurzbeschreibung zum jeweiligen Element bzw. zur jeweiligen Funktion !**

Je nach Element/Funktion fällt diese Kontexthilfe mal mehr und mal weniger ausführlich aus.

Dieses Fenster bleibt solange geöffnet, bis es durch Klicken auf das rot/weiße Kreuz in der oberen rechten Fensterecke geschlossen wird.

### Übung:

Öffnen Sie die Kontexthilfe und fahren Sie mit der Maus über die Additionsfunktion, **Abb.4.2.7:**



**Abb.4.2.7: Im Kontexthilfe-Fenster erscheinen Informationen zum Element bzw. zur Funktion, hier zum Funktion 'Addieren'**

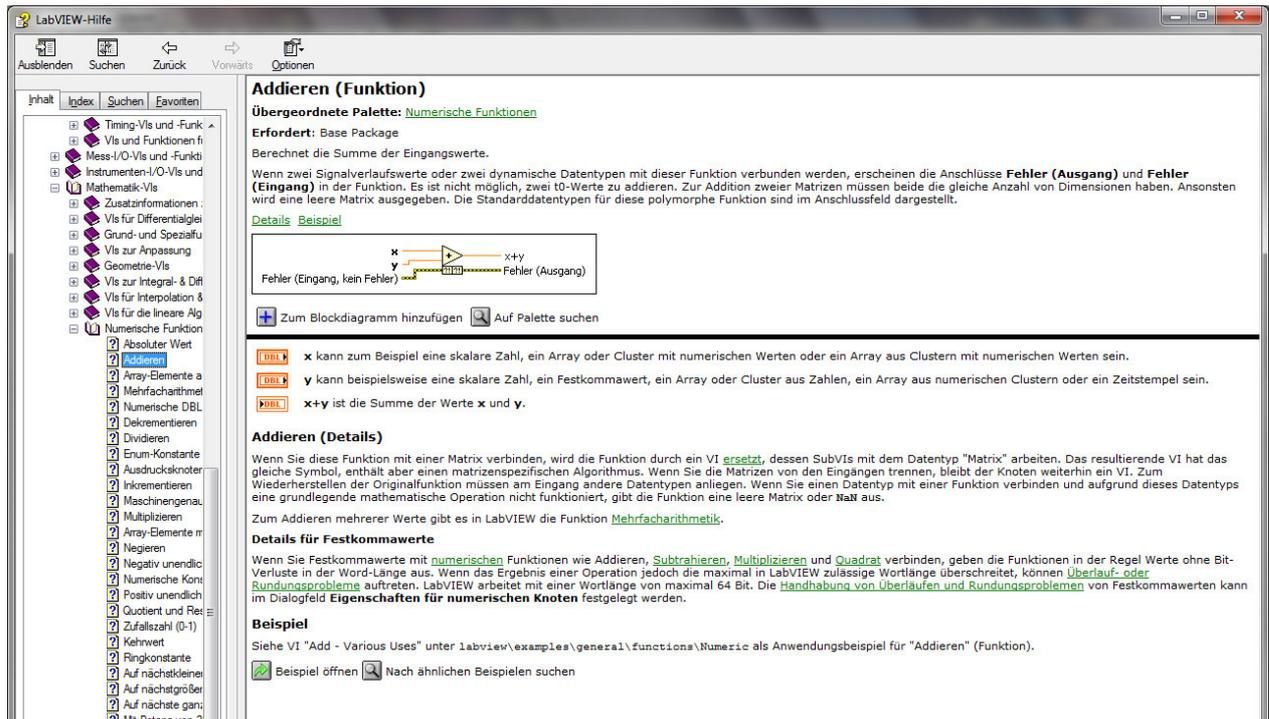
Auch bei vielen Menüs, Untermenüs bzw. Einstellmöglichkeiten auf den LabVIEW-Registerkarten funktioniert diese Kontexthilfe und zeigt Ihnen dann an, was man unter diesem Punkt alles machen kann:

- In der ersten Zeile erscheint der Name der Funktion.
- Danach folgt das 'offiziell' LabVIEW-Symbol für diese Funktion.
- Und nachfolgend eine kurze Beschreibung zur 'Funktion der Funktion'.

Und zum Schluss **ganz wichtig:**

Es folgt ein Link in die „**Ausführliche Hilfe**“, wo man noch mehr Informationen zu dieser Funktion erhält.

Klicken Sie auf diesen Link, so wird die ausführliche LabVIEW Hilfebeschreibung zu dieser Funktion aufgerufen, **Abb.4.2.8:**



**Abb.4.2.8: Es geht auch ausführlicher**

Hier erfahren Sie dann wirklich alles, was es zu dieser Funktion, zu diesem Element, ... zu sagen gibt.

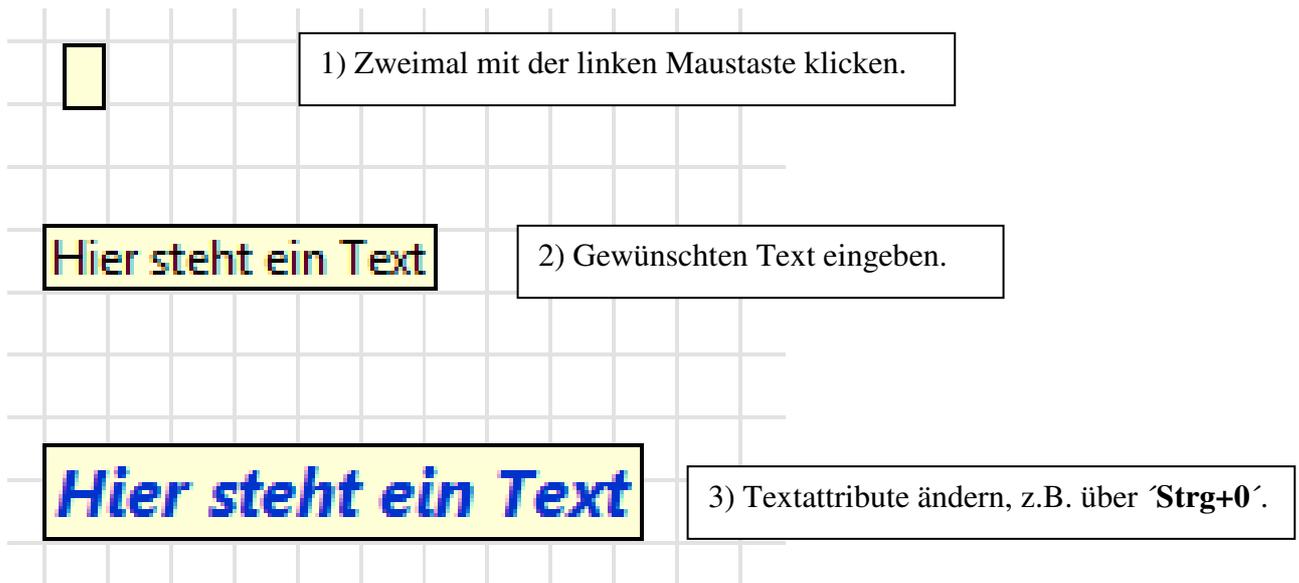
### Übung:

Schauen Sie sich die Kontexthilfen (auch die ausführliche Version) zur Funktion 'Subtrahieren' und zu den Anzeige- und Bedienelementen an.

### Das Einfügen von Festtexten

Unter LabVIEW haben Sie die Möglichkeit, im Frontpanel und im Blockdiagramm beliebige, frei wählbare zusätzliche Erläuterungstexte zu einzelnen Funktionen oder Funktionsgruppen bzw. zu einzelnen Anzeige- und Bedienelementen anzugeben.

Das Einfügen solcher Texte ist nun sehr einfach und schnell gemacht, **Abb.4.2.9:**



**Abb.4.2.9: Die Eingabe von zusätzlichen Erläuterungstexten auf dem Blockdiagramm**

- 1) Klicken Sie dazu einfach mit der linken Maustaste zweimal auf die Stelle im Blockdiagramm, wo der Text hin soll: es erscheint ein kleines hellgelbes Kästchen mit dem blinkenden „Texteingabe-Cursor“.
- 2) Geben Sie nun den gewünschten Text ein.
- 3) Die Textattribute (Texteigenschaften) können Sie danach beliebig ändern, z.B. dadurch, dass Sie den Text markieren und mit dem Shortcut **'Strg+0 (Null)'** die Registerkarte **'Auswahlschriftart'** aufrufen und dort die gewünschten Änderungen/Anpassungen durchführen.

Der so eingegebene Text erscheint allerdings nur auf dem Blockdiagramm.

Auf dem **Frontpanel** können Sie ebenfalls, genauso wie gerade beschrieben, beliebige Erläuterungstexte eingeben, allerdings mit dem kleinen Unterscheid, dass diese Texte später beim Ablauf des VI auch mit auf dem Frontpanel erscheinen, es sind also direkte zusätzliche Frontpanel-Beschriftungstexte.

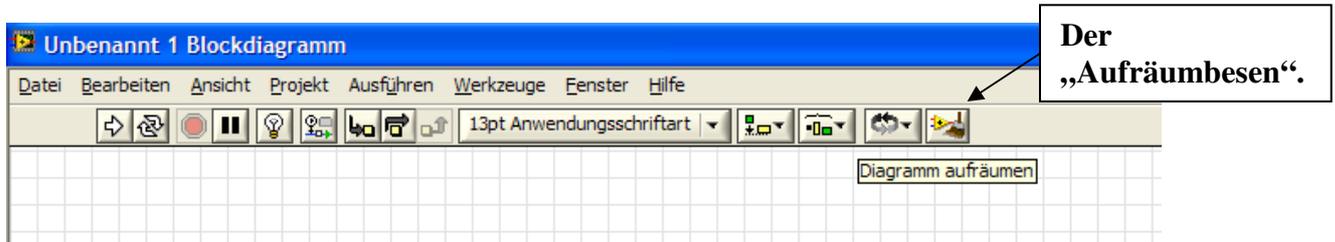
### **Jetzt wird aufgeräumt**

„Im Laufe des Gefechtes“ kann es sicherlich einmal vorkommen, dass Sie im Blockdiagramm die Verbindungsdrähte zwischen den Blöcken doch recht „wild“ und „unübersichtlich“ verlegt haben.

Hier bietet LabVIEW Ihnen nun an, das Blockdiagramm schön ordentlich aufzuräumen.

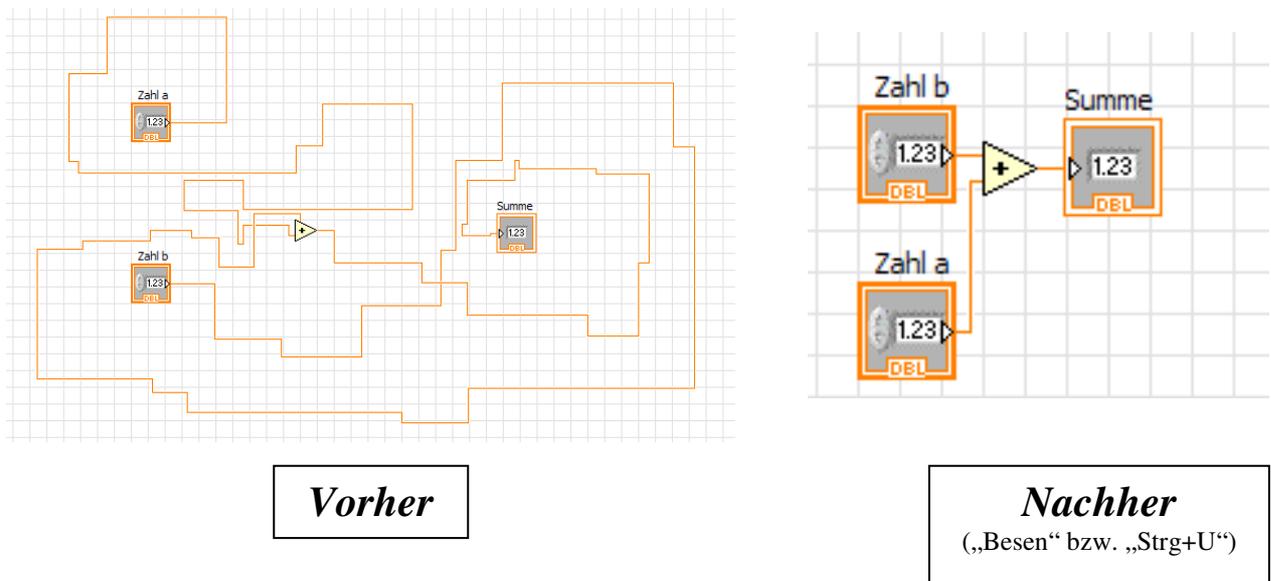
Dazu können Sie entweder

- auf das „**Besen-Ikon**“ klicken, **Abb.4.2.10**
- oder
- Sie benutzen einfach den Short-Cut **‘Strg+U’**



**Abb.4.2.10: Der Aufräumbesen**

Das Ergebnis einer Aufräumaktion können Sie in **Abb.4.2.11** sehen:



**Abb.4.2.11: Der Besen macht's**

Und denken Sie immer daran: wenn Ihnen das Aufräumergebnis nicht gefällt, so können Sie jeder Zeit mit **‘Strg+Z’** alles wieder rückgängig machen und „selbst Hand“ anlegen.

**Die Erweiterung von Funktionen**

Bei vielen LabVIEW-Funktionen steht dem Anwender nach dem Einfügen der Funktion ins Blockdiagramm nur ein bestimmter, minimaler Grundfunktionsumfang zur Verfügung.

**Beispiel:**

Die Funktion 'Strings verknüpfen': BD\String\Verknüpfen.

Hiermit werden mehrere Teilstrings zu einem Gesamtstring verknüpft.

Fügt man diese Funktion nun ins Blockdiagramm ein, so besitzt diese Funktion zunächst nur zwei Eingänge (zur Verknüpfung von zwei Teilstrings) und einen Ausgang (für den Ergebnisstring), **Abb.4.2.12 – oben**.

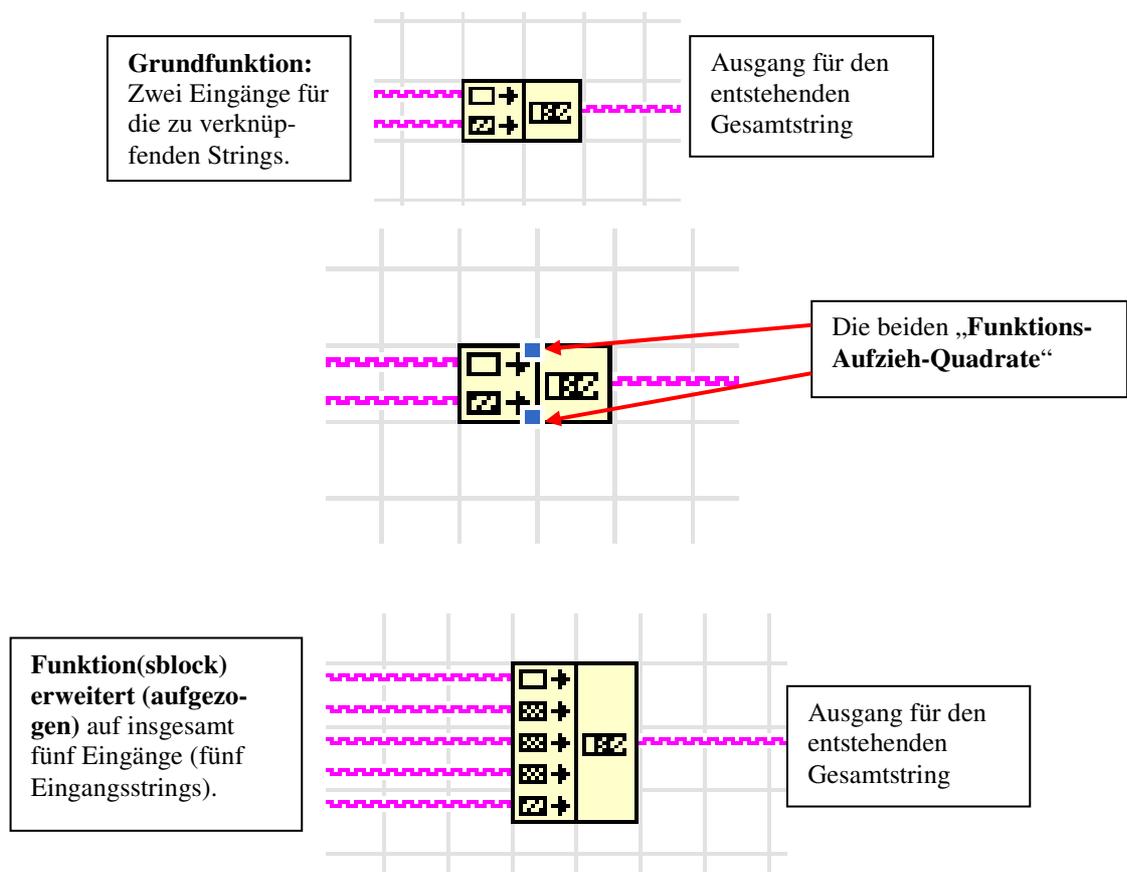
Das bedeutet: es können zunächst (**nur**) **zwei Teilstrings** zu einem neuen Gesamtstring verknüpft werden.

Und nun machen Sie Bekanntschaft mit einer weiteren, sehr wichtigen Eigenschaft von LabVIEW-Funktionen:

**Viele Funktionen können im Blockdiagramm „auf Maus-Klick hin“ einfach vergrößert bzw. in ihrem Funktionsumfang erweitert werden !**

Wenn Sie mit der Maus über das Funktionssymbol fahren und der Cursor seine Form in die des kleinen Pfeils verändert, so erscheinen zwei kleine blaue Quadrate, oben und unten am Funktionssymbol: das sind die „**Funktions-Aufzieh-Punkte**“.

Damit kann man die Funktion nun nach oben bzw. nach unten aufziehen, vergrößern, und d.h., **Abb.4.2.12 - Mitte**:



**Abb.4.2.12: Die Erweiterung der Funktion 'Strings verknüpfen'**

Die Funktion kann also durch zusätzliche Eingänge und/oder Ausgänge (je nach Art der Funktion) in ihrem Funktionsumfang erweitert werden.

In unserem konkreten Fall bedeutet das hier: man kann die Anzahl der Eingänge (fast) beliebig erhöhen und so beliebig viele Teilstrings zu einem Gesamtstring verknüpfen.

Die Anzahl der Ausgänge bleibt dabei aber bei 1, denn es gibt ja immer nur einen Ergebnisstring.

Entsprechend kann man die Funktion auch „verkleinern“ indem man das Funktionssymbol wieder „zusammen schiebt“.

Das klappt aber nur dann, wenn die entsprechenden Eingänge (noch) unbelegt sind.

Damit haben Sie nun einen ersten, ganz kleinen Überblick über die Möglichkeiten ein Blockdiagramm zu erzeugen, bekommen

Dieses Wissen reicht allerdings schon aus, um unser kleines Projekt nachfolgend zu realisieren.

Wenden wir uns daher nun dem Entwurf eines Frontpanels zu.

### 4.3 Die Gestaltung des Frontpanels

Das Frontpanel ist nun die **eigentliche Bedienoberfläche** des Messgerätes und zu deren Erstellung stehen dem Anwender unter LabVIEW mehr als 300 (!) individuell konfigurierbare Bedien- und Anzeige-Elemente zur Verfügung.

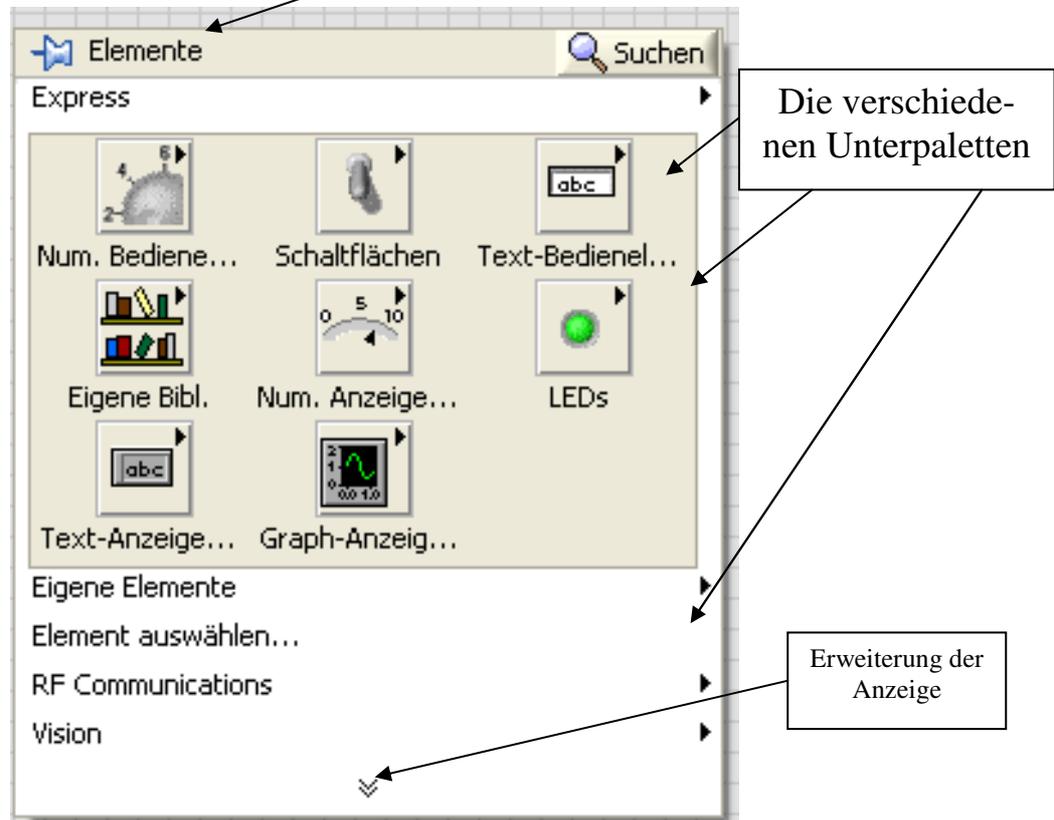
Professionell gestaltete graphische Benutzeroberflächen können so einfach entworfen werden und dazu sind im Detail vorhanden:

- **Bedien-Elemente:** Drehknöpfe, Schalter, Textfelder, Schieberegler, ...
- **Anzeige-Elemente:** Ziffernanzeigen, Rundinstrumente, Thermometer, LEDs, graphische Kurvenverläufe, ...
- **Gestaltungs- und Dekor-Elemente** zum optischen „Aufpeppen“ von Frontpanels.

Öffnen Sie daher zunächst ein noch leeres VI und schalten Sie um auf das Frontpanel.

Klickt man nun **mit der rechten Maustaste auf eine freie Fläche auf dem Frontpanel-Bereich** so erscheint die Palette (Liste) mit allen zur Verfügung stehenden Frontpanel-**Elementen, Abb.4.3.1:**

Die **Elemente** bzw. die **Elementpalette** zur Gestaltung des Frontpanels



**Abb.4.3.1: Die Elemente zur Gestaltung eines Frontpanels (Elementpalette)**

Klickt man nun auf den **nach unten gerichteten Doppelpfeil** ( $\equiv$  Erweiterung der Anzeige), so erscheinen noch weitere Unterpaletten mit ihren zugehörigen Anzeige- und Bedienelementen.

Bei der Gestaltung von Frontpanels gilt daher:

### **Merke: Der Entwurf des Frontpanels**

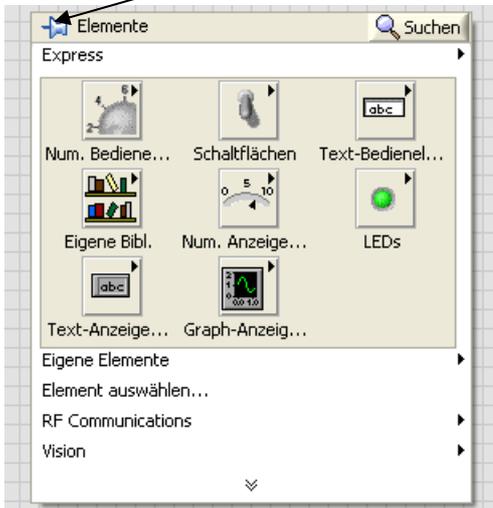
Beim Entwurf von Frontpanels arbeitet man mit den so genannten „**Elementen**“ aus der Elementpalette, die sich wiederum in verschiedene Unterpaletten unterteilt.

Zunächst passen wir die Darstellung der Elementpalette aus der Abb.4.3.1 noch etwas an, so dass diese für unseren täglichen Gebrauch besser geeignet ist.

Wir werden in diesem LabVIEW-Projekt zu fast 90% mit den Elementen der **Unterpalette 'Modern'** arbeiten, so dass diese sinnvoller Weise beim Öffnen der Gesamtelementpalette an erster Stelle steht und auch bereits dauerhaft geöffnet ist (und das auch bleibt).

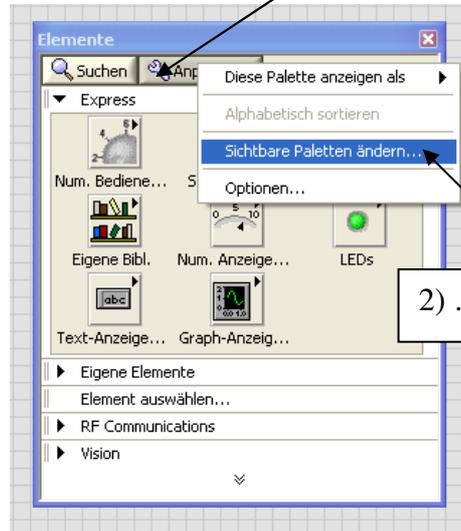
Dieses müssen wir nun noch einstellen, **Abb.4.3.2:**

Hier klicken.



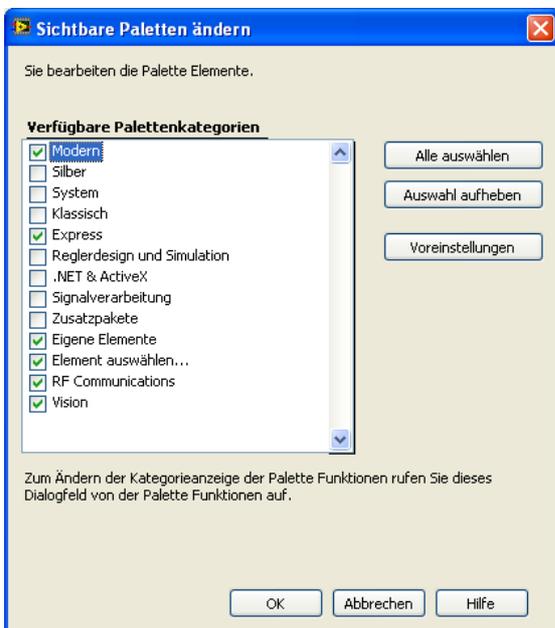
**Bild 1):** Mit der rechten Maustaste auf eine freie Stelle im Frontpanel klicken und dann auf die Heftzwecke klicken.

1) Hier klicken und dann ...

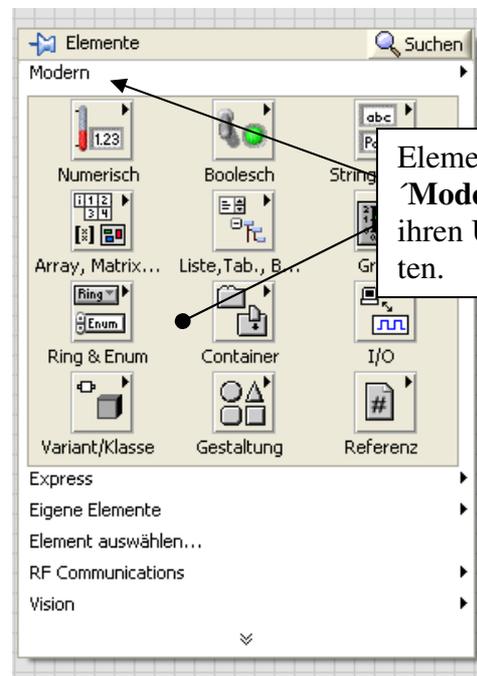


2) ... hier klicken.

**Bild 2):** Auf den Button 'Anpassen' klicken und dann auf 'Sichtbare Paletten ändern ...' klicken.



**Bild 3):** Den Eintrag 'Modern' aktivieren und mit 'OK' bestätigen.



Elementpalette 'Modern' mit all ihren Unterpaletten.

**Bild 4):** Die Elementpalette 'Modern' mit all ihren Unterpaletten ist nun dauerhaft sichtbar.

### Abb.4.3.2: Die Sichtbarmachung der Elementpalette 'Modern'

- 1) Klicken Sie dazu als erstes mit der rechten Maustaste auf eine freie Stelle im Frontpanel.  
Es erscheint ein Menü mit allen zur Zeit sichtbaren Elementpaletten, **Bild 1**).  
Leider ist die Palette 'Modern' hier zur Zeit noch nicht enthalten und diese muss daher jetzt erst 'sichtbar' gemacht werden.  
Klicken Sie dazu auf das 'Heftzwecken-Symbol' oben links im Fenster.
- 2) **Bild 2**): Das Fenster mit den Elementpaletten wird nun auf dem Frontpanel fixiert.  
Klicken Sie hier nun auf den Button 'Anpassen'.  
Es erscheint ein weiteres Auswahlm Menü in dem Sie auf 'Sichtbare Paletten ändern ...' klicken.
- 3) **Bild 3**): In der nun erscheinenden Liste der möglichen Elementpaletten markieren Sie den Eintrag 'Modern' und klicken dann auf 'OK'.
- 4) **Bild 4**): Ab jetzt erscheint bei jedem neuen Aufruf der Elementpaletten ( $\equiv$  Klick mit rechten Maustaste auf eine freie Stelle im Frontpanel) die Elementpalette 'Modern' an erster Stelle, bereits geöffnet mit all ihren Unterpaletten.

Schließen Sie nun die Elementpalette.

#### Wichtig: Die Elemente des Frontpanels

Zur Gestaltung eines Frontpanels stehen dem Anwender in der von uns verwendeten Version 'LabVIEW 2011' über 300 (!) Elemente zur Verfügung, die in bis zu 9 verschiedenen Unterpaletten angeordnet sind.

Gerade für den Einsteiger ist es daher **sehr unübersichtlich** und **verwirrend**, die richtigen Elemente zu finden bzw. überhaupt erst einmal festzustellen, welche Elemente es eigentlich wo, in welcher Palette, gibt.

Wir verwenden daher ein **einheitliches Bezeichnungsschema**, wie wir das schon bei der Erstellung des Blockdiagramms vorgestellt haben.

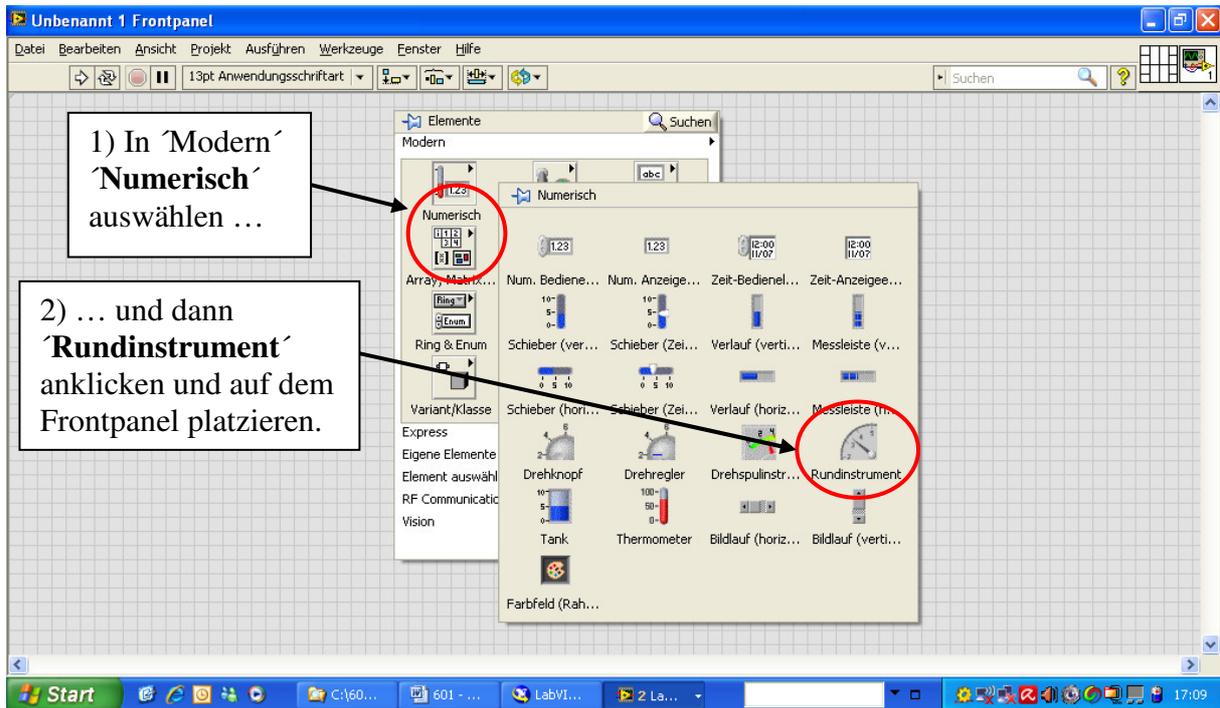
#### **Beispiel:**

Wir wollen ein 'Rundinstrument' als Anzeigeelement in unser Frontpanel einfügen.

Als '**Ziel-Wegbeschreibung**' zum Auffinden dieses Elementes geben wir einfach an:

***FP\Modern\Numerisch\Rundinstrument***

was im Detail wie folgt zu interpretieren ist, **Abb.4.3.3**:



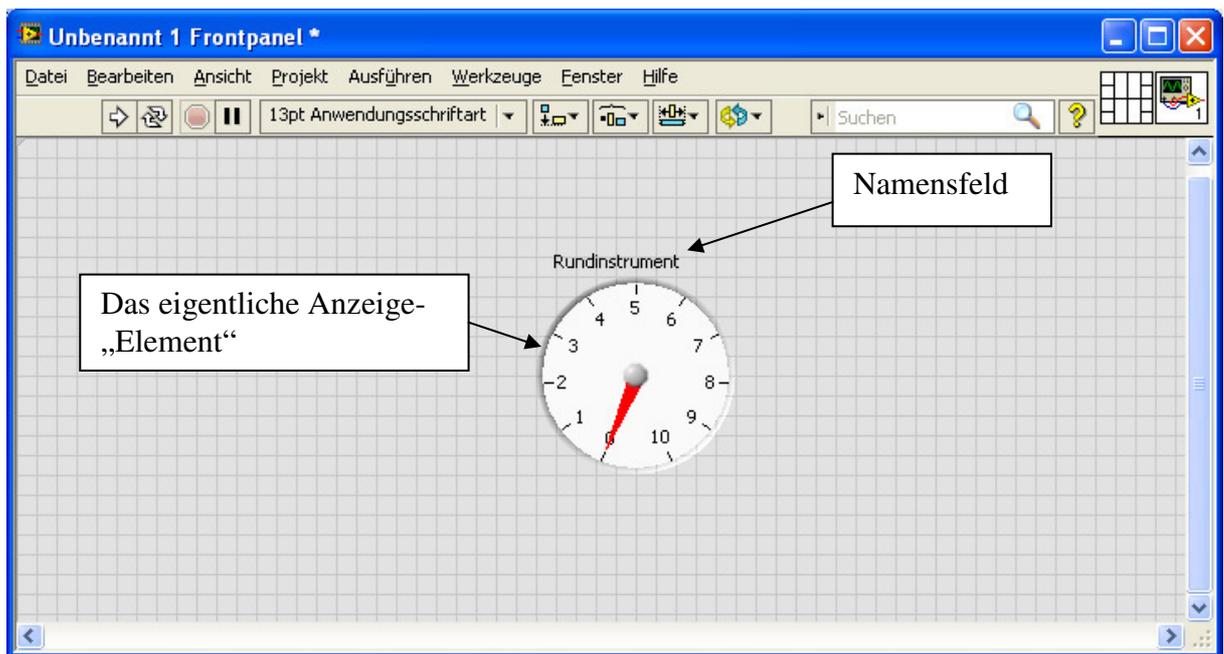
**Abb.4.3.3: Der Einbau eines Rundinstrumentes (Anzeigeelement) auf dem Frontpanel**

- **FP**  $\equiv$  Arbeiten auf dem Frontpanel , d.h. Aufruf des Frontpanels und mit der rechten Maustaste auf eine freie Stelle des Frontpanels klicken.  
Es öffnet sich die **Elementpalette** mit der nun bereits geöffneten Unterpalette 'Modern'.
- **Modern**  $\equiv$  Auswahl der Unterpalette 'Modern' (ist hier bereits schon erfolgt).  
(Bei anderen Unterpaletten muss man unter Umständen zuerst die Unterpaletten-Anzeige erweitern, bevor man diese Unterpaletten angezeigt bekommt: s. dazu Abb.4.3.1 „Erweiterung der Anzeige“, Anklicken des Doppelpfeils).
- **Numerisch**  $\equiv$  Anschließend fährt man mit der Maus über 'Numerisch' und nun erscheinen endgültig alle verfügbaren Frontpanel-Elemente aus dieser Gruppe ( $\equiv$  numerische Anzeige- und Bedienelemente).
- **Rundinstrument**  $\equiv$  Jetzt wählt man sich das gewünschte Element aus, d.h. man klickt mit der Maus auf das Element: alle Paletten schließen sich und auf dem Frontpanel erscheint der

Mauszeiger in **Handform**, dem nun das Rundinstrument angeheftet ist.

Man kann dieses Element abschließend auf seinen endgültigen Platz verschieben und dort per Mausklick ablegen.

Das Ergebnis sehen Sie in **Abb.4.3.4**:



**Abb.4.3.4: Das erste Anzeigeelement auf dem Frontpanel**

Dieses Rundinstrument besteht nun aus zwei Teilen:

- dem eigentlichen **Anzeige-Element**: also dem Zeiger mit der Rundskala

und

- dem **Namensfeld**: denn man kann diesem Anzeigeelement natürlich einen Namen geben, z.B. welche Größe eigentlich angezeigt wird und mit welcher Einheit.

Werden nun Anzeige- und Bedienelemente auf diese direkte Art und Weise ins Frontpanel eingefügt, so kreiert LabVIEW dazu auf dem Blockdiagramm automatisch das entsprechende Funktionssymbol, das dann dort beliebig weiter verdrahtet werden kann.

Prüfen Sie das einmal nach, indem Sie auf das Blockdiagramm umschalten.

Beim weiteren Entwurf bzw. bei der Gestaltung des Frontpanels stehen Ihnen nun im Wesentlichen die gleichen Hilfsmittel zur Verfügung wie bei der Erstellung des Blockdiagramms.

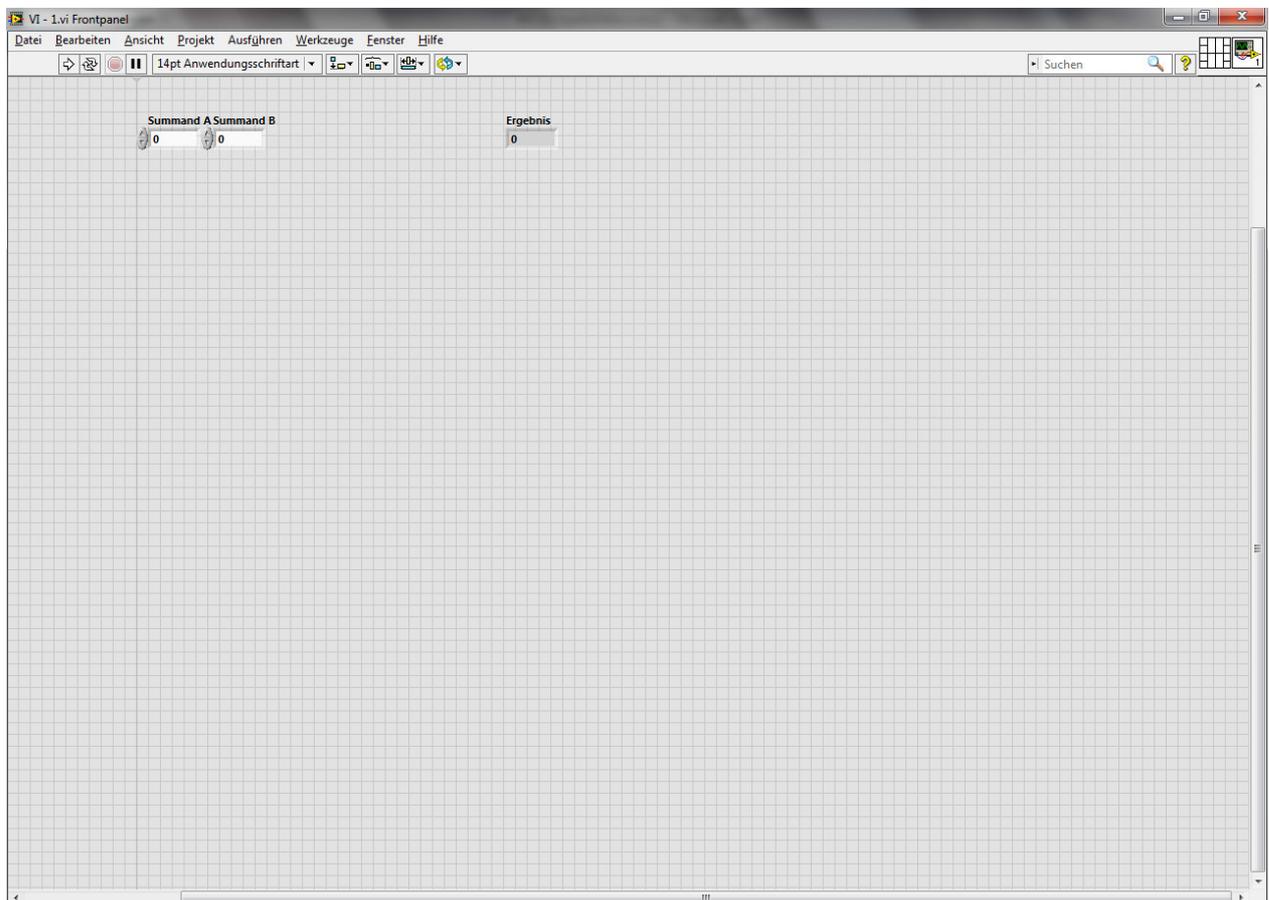
Die **Bearbeitungshilfsmittel**, die Sie jetzt dazu benötigen sind:

- das Markieren von Elementen
- das Verschieben von Elementen
- das Vergrößern/Verkleinern von Elementen
- die Änderung von Schrift-Attributen: Größe, Farbe, Ausrichtung, ...
- das Ändern von Eigenschaften von Elementen.

Das Ändern von Eigenschaften geschieht zunächst zum Großteil über das **Kontextmenü**, Unterpunkt **Eigenschaften**, des jeweiligen Anzeige- bzw. Bedienelementes.

Schließen Sie nun das VI (ohne die Änderungen zu speichern) und öffnen Sie unser bereits erstelltes VI **VI – 1.vi**.

Schalten Sie um auf das Frontpanel, **Abb.4.3.5**:



**Abb.4.3.5: Das Frontpanel zum VI 'VI – 1.vi'**

Sie sehen hier

- die beiden Bedienelemente 'Summand A' und 'Summand B' und
- das Anzeige-Element 'Ergebnis'

noch recht willkürlich angeordnet.

Markieren und verschieben Sie nun jedes Element, bis die Gestaltung des FPs ihren Vorstellungen entspricht.

Die **Attribute** eines markierten Elementes (Textgröße, Farbe, etc.) können Sie mit 'Strg+0' ändern.

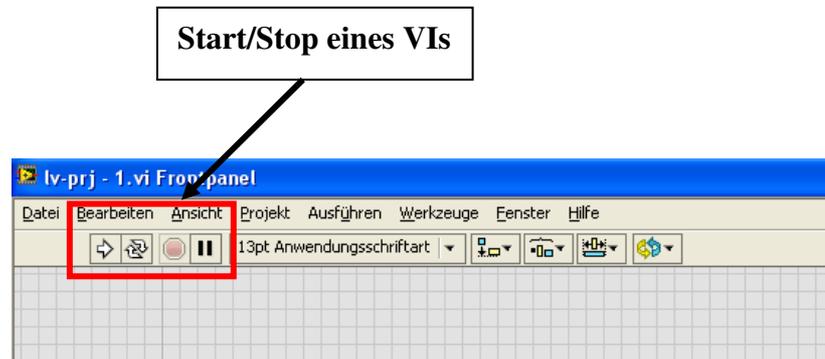
Das Ergebnis könnte dann so aussehen, **Abb.4.3.6**:



**Abb.4.3.6: Ein „hübsch“ gestaltetes Frontpanel**

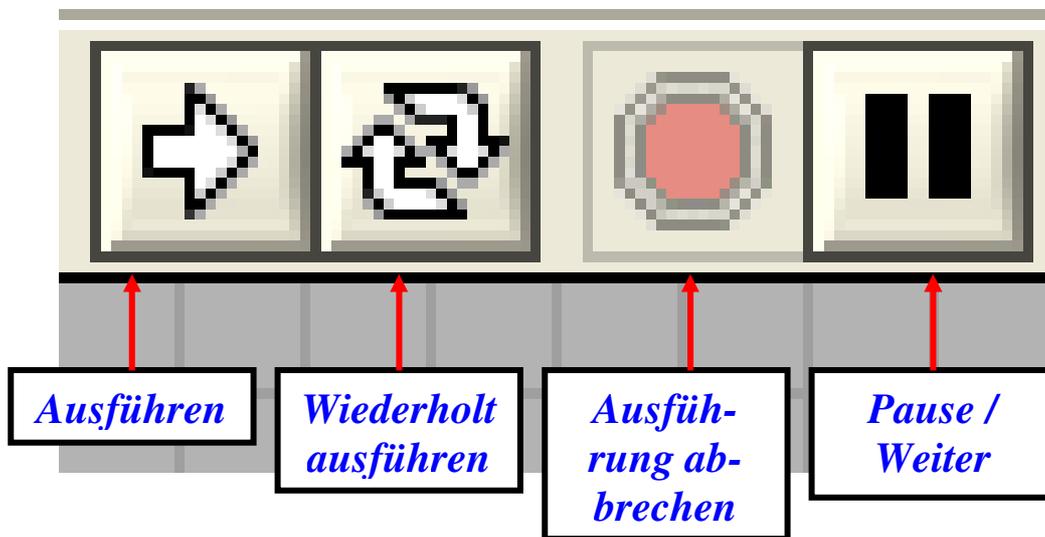
## 4.4 Der Start des VIs

Zum Starten, Stoppen und Anhalten eines VIs sind nun vier „Tastenfelder“ in der linken oberen Fensterecke maßgeblich, die Sie sowohl auf dem Frontpanel als auch auf dem Blockdiagramm vorfinden, **Abb.4.4.1**:



**Abb.4.4.1: Die wichtige „linke obere Ecke“**

Abb.4.4.2 zeigt die Details:



**Abb.4.4.2: Die wichtigen „Start/Stop-Knöpfe“**

Hier bedeuten nun:

**Ausführen:** Das VI wird genau einmal ausgeführt, d.h. abgearbeitet, danach stoppt der Ablauf.

<b>Wiederholt ausführen:</b>	Das VI wird permanent, nacheinander immer wieder ausgeführt.
<b>Ausführung abbrechen:</b>	Die Abarbeitung des VIs wird abgebrochen, d.h. endgültig beendet ( $\equiv$ „Not-Aus“-Funktion).
<b>Pause / Weiter:</b>	Durch Druck auf dieses Feld, wird die Abarbeitung des VIs angehalten und wird durch erneuten Druck auf dieses Feld wieder weiter fortgesetzt. Befindet man sich auf dem Frontpanel, so wird beim Einleiten der Pause automatisch das Blockdiagramm in den Vordergrund geholt.

Wenn Sie nun auf **‘Ausführen’** klicken, so startet unser VI und die Messwerte werden empfangen und angezeigt.  
(Zuvor muss natürlich noch das Mikrocontroller-System gestartet und dessen serielle Schnittstelle korrekt mit dem USB/Seriell-Adapter verbunden worden sein)

Als Zeichen dafür, dass das VI abläuft, werden Sie auf dem Frontpanel folgendes feststellen,

- das Zeichenraster auf dem Frontpanel verschwunden ist,
- einige Ikons aus der oberen Reihe verschwunden sind und
- die vier Start/Stop-Schaltflächen-Ikone fett dargestellt sind.

All das sind Zeichen dafür, **dass das VI abläuft !**

Stoppen können Sie das VI durch Klicken auf unseren **‘Stopp-Knopf’** auf dem Frontpanel oder, wenn das VI aufgrund eines Fehlers nicht mehr richtig reagiert oder nicht mehr richtig arbeitet, durch Klicken auf das **‘Ausführung abbrechen’**-Ikon (als **‘Not-Aus-Funktion’**).

Starten Sie unser VI **‘VI – 1.vi’** nun vom Frontpanel aus durch Klicken auf **‘Wiederholt ausführen’** (da das VI ja jetzt permanent ablaufen soll und wir noch keine Endlosschleife programmiert haben).

Sie können nun bei den beiden Summanden beliebige Werte eingeben und erhalten sofort das Ergebnis unserer kleinen Berechnung, **Abb.4.4.3:**



**Abb.4.4.3: Das VI läuft ...**

Anhalten können Sie das VI nun durch Klicken auf 'Ausführung abbrechen'.

## **5. Das Projekt**

Nun endlich kommen wir zur Realisierung des in Kapitel 2 beschriebenen Projektes.

## 5.1 Das Blockdiagramm

Eine Kernfunktion unseres zu entwickelnden LabVIEW-VIs ist der Sende-/Empfangs-Betrieb über eine seriellen COM-Schnittstelle bzw. über eine USB-Schnittstelle mit USB/Seriell-Adapter.

Grundsätzlich sind dabei in LabVIEW (immer) die folgenden Aktionen durchzuführen:

- 1) Serielle Schnittstelle initialisieren, d.h. Auswahl der gewünschten seriellen COM-Schnittstelle und Festlegung der passenden Datenübertragungsparameter (Baudrate, Datenwortbreite, ...).
- 2) Vor Beginn des eigentlichen Empfangs von Daten bzw. direkt nach dem Start des VIs: Empfangsbuffer der seriellen Schnittstelle leeren, d.h. ev. Reste von der vorhergehenden Datenübertragung löschen.  
Ferner bei Bedarf: vor erstmaligem Beginn der Kommunikation auch den zugehörigen Sendedatenpuffer löschen.
- 3) Zeichen über die serielle Schnittstelle empfangen bzw. Zeichen über die serielle Schnittstelle aussenden.
- 4) Ganz zum Schluss, wenn die Datenübertragung endgültig beendet ist bzw. wenn das gesamte VI beendet wird: ordnungsgemäßes Schließen bzw. Freigabe der seriellen Schnittstelle, damit auch andere Programme ab jetzt wieder auf diese Schnittstelle zugreifen können.

Zwischen den Punkten 2), 3) und 4) liegt dann, bildlich gesprochen, die eigentliche Applikation:

- Auswertung von empfangenen Daten.
- Zusammenstellung von Sendedaten.
- „Optisch schöne“ Darstellung der Daten auf dem Monitor.
- Weitere sinnvolle „Zusatzfunktionen“.

### Die serielle Datenübertragung unter LabVIEW mittels VISA-Funktionen

Zum kompletten Betrieb von seriellen Schnittstellen bietet LabVIEW dem Anwender eine ganze Sammlung von bereits fertig programmierten Funktionen, die nur noch entsprechend aufgerufen und für die jeweiligen Zwecke parametrisiert werden müssen: die sogenannten **VISA-Funktionen**.

**VISA = Virtual Instruments System Architecture**

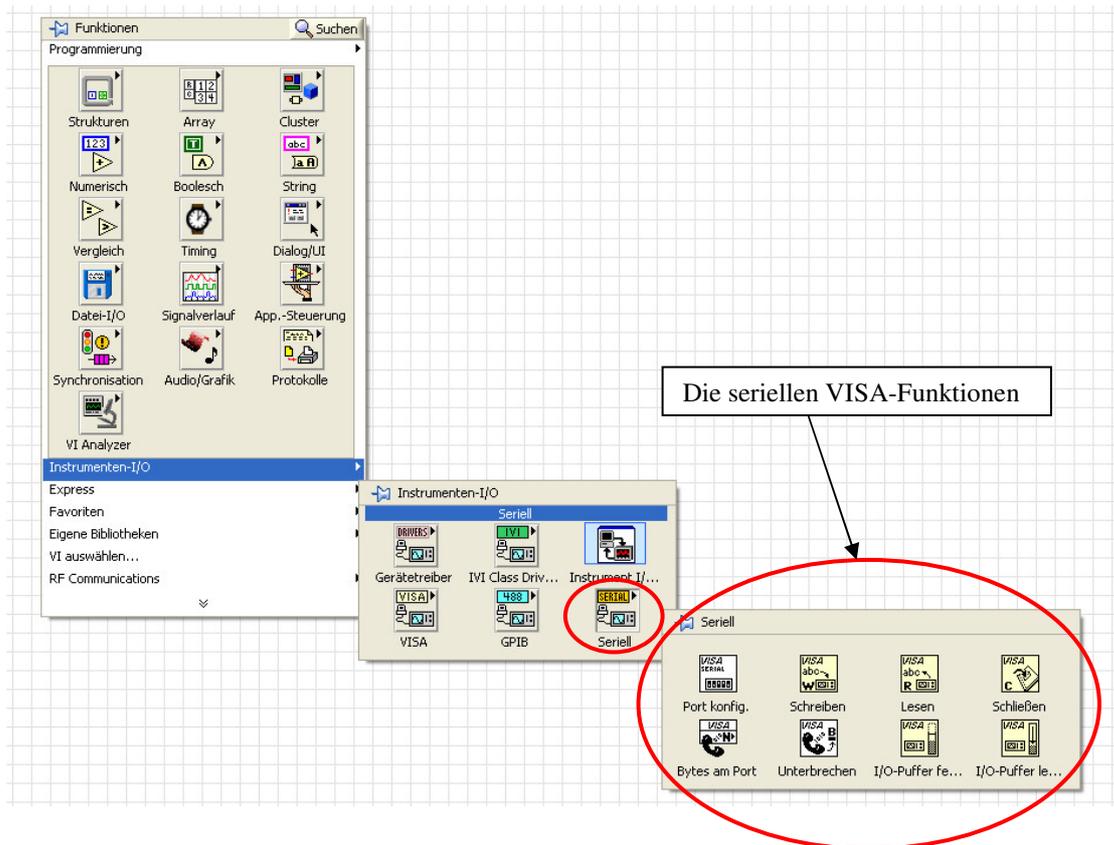
≡ Sammlung von Funktionen zur Steuerung von Geräten über:

- Serielle Schnittstellen (RS232, RS422, RS485)

- USB
- Ethernet
- GPIB
- u.ä

Diese Funktionen sind zu finden unter, **Abb.5.1.1:**

### BD\Instrumenten-I/O\Seriiell



**Abb.5.1.1: Der Weg zu den seriellen VISA-Funktionen**

Für unser erstes LabVIEW-VI in diesem Projekt benötigen wir aus dieser Funktionssammlung im Wesentlichen **nur fünf Funktionen:**

- VISA: Seriellen Port konfigurieren.
- VISA: I/O-Puffer entleeren.
- VISA: Lesen.
- VISA: Schreiben.
- VISA: Schließen.

**Hinweis:**

Die Funktion **´VISA: Schreiben´**, also das Aussenden von Daten über die serielle Schnittstelle, benötigen wir in diesem Projekt nicht.

Nach diesen Vorbemerkungen können wir mit der Entwicklung des Blockdiagramms für unser VI beginnen.

Starten Sie LabVIEW und öffnen Sie ein leeres VI. Der Cursor befindet sich zunächst noch auf dem Frontpanel.

Speichern Sie dieses VI in einem Verzeichnis Ihrer Wahl unter dem Namen **´lv-prj – 1.vi´** ab. (**Datei\Speichern unter ...** und dann weiter, wie unter Windows gewohnt).

Schalten Sie danach um auf das Blockdiagramm (**´Strg+E´**) und fügen Sie als erstes die VISA-Funktion **´VISA: Seriellen Port konfigurieren´** ein. (**BD\Instrumenten-I/O\Seriell\Port konfig.´**)

Öffnen Sie die **Kontexthilfe** (**´Strg+H´**) und fahren Sie mit der Maus über die Funktion **´VISA: Seriellen Port konfigurieren´** so dass die Kurzbeschreibung dazu in der Kontexthilfe angezeigt wird.

Rufen Sie dann die **´Ausführliche Hilfe´** zu dieser Funktion auf, **Abb.5.1.2:**

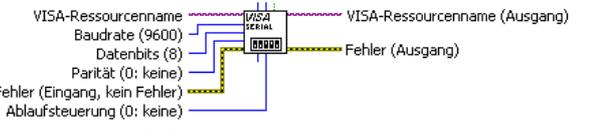
Die Funktion



Die (kurze) Kontexthilfe

**Kontexthilfe**

**VISA: Seriellen Port konfigurieren**  
[VISA Configure Serial Port]



VISA-Ressourcenname (Ausgang)  
Baudrate (9600)  
Datenbits (8)  
Parität (0: keine)  
Fehler (Eingang, kein Fehler)  
Ablaufsteuerung (0: keine)

Initialisiert den über **VISA-Ressourcenname** ausgewählten seriellen Port mit den angegebenen Einstellungen. Zur Auswahl der polymorphen Instanz verbinden Sie Daten mit dem Eingang **VISA-Ressourcenname** oder wählen Sie die Instanz manuell aus.

[Ausführliche Hilfe](#)

**LabVIEW-Hilfe**

Ausblenden Suchen Zurück Vorwärts Optionen

Inhalt Index Suchen Favoriten

- VI- und Funktionenübersicht
  - VIs und Funktionen zur Pr
  - Mess-/I/O-VIs und -Funktio
  - Instrumenten-/I/O-VIs und
    - Konfigurieren serieller Element "VISA-Ressou
    - Voreinstellungen für G
    - VISA-Ereignisse
    - Gerätetreiber-VIs
  - GPiB-Funktionen
  - Serielle VIs und Funktio
    - VISA: Bytes am st
    - VISA: Schließen
    - VISA: Seriellen Pc
    - VISA: I/O-Puffer e
    - VISA: Lesen
    - VISA: Serielle Übe
    - VISA: I/O-Pufferg
    - VISA: Schreiben
  - VISA-VIs und -Funktio
    - VISA: Trigger freig
    - VISA: Zurücksetz
    - VISA: Lesen
    - VISA: STB lesen
    - VISA: Schreiben
  - VISA: Fortgeschrit
    - Eigenschaftsk
    - VISA: Schließ
    - VISA: Ressou
    - VISA: Asynch
    - VISA: Öffnen
    - VISA: In Date
    - VISA: Timeou

**VISA: Seriellen Port konfigurieren (VI)**

Übergeordnete Palette: [Serielle VIs und Funktionen](#)

Erfordert: Base Package

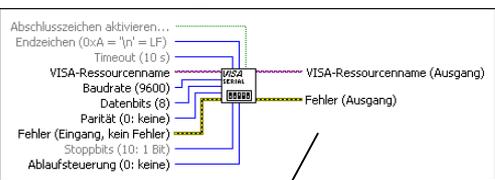
Initialisiert den über **VISA-Ressourcenname** ausgewählten seriellen Port mit den angegebenen Einstellungen. Zur [Beispiel](#)

In diesem Pull-down-Menü wird die gewünschte VI-Instanz ausgewählt.

Instanz auswählen

Zum Blockdiagramm hinzufügen Auf Palette suchen

**VISA: Seriellen Port konfigurieren**



Abschlusszeichen aktivieren...  
Endzeichen (0xA = '\n' = LF)  
Timeout (10 s)  
VISA-Ressourcenname (Ausgang)  
Baudrate (9600)  
Datenbits (8)  
Parität (0: keine)  
Fehler (Eingang, kein Fehler)  
Stoppbits (10: 1 Bit)  
Ablaufsteuerung (0: keine)

**Abschlusszeichen aktivieren** bereitet das serielle Gerät auf das **Abschlusszeichen** vor. Bei TRUE (Vo gesetzt, das heißt, das Gerät erkennt das **Abschlusszeichen** nicht.

**Abschlusszeichen** führt zur Beendigung des Lesevorgangs. Der Lesevorgang wird beendet, wenn das Ab 0xD bei Nachrichten, die mit einem Wagenrücklauf (\r) enden.

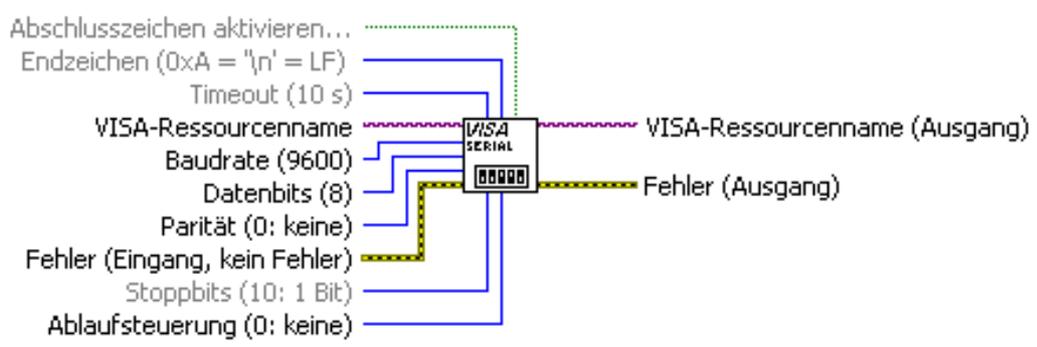
**Timeout** gibt die maximale Dauer von Schreib- und Leseoperationen in Millisekunden an. Die Voreinstellung

**VISA-Ressourcenname** gibt die zu öffnende Ressource an. Mit dem Element [VISA-Ressourcenname](#) wer

**Baudrate** ist die Übertragungsgeschwindigkeit. Die Voreinstellung lautet 9600.

**Datenbits** ist die Anzahl der Bits in den eingehenden Daten. Der Wert von **Datenbits** liegt zwischen 5 und

**VISA: Seriellen Port konfigurieren**



Abschlusszeichen aktivieren...  
Endzeichen (0xA = '\n' = LF)  
Timeout (10 s)  
VISA-Ressourcenname (Ausgang)  
Baudrate (9600)  
Datenbits (8)  
Parität (0: keine)  
Fehler (Eingang, kein Fehler)  
Stoppbits (10: 1 Bit)  
Ablaufsteuerung (0: keine)

### **Abb.5.1.2: Die Funktion 'VISA: Seriellen Port konfigurieren' und die Hilfen zu dieser Funktion**

Mit Hilfe dieser Funktion kann nun die gewünschte serielle Schnittstelle ausgewählt und die zugehörigen Datenübertragungsparameter entsprechend eingestellt werden.

Der untere Teil der Abb.5.1.2 zeigt im Detail die vorhandenen Ein- und Ausgänge der Funktion, die nun „passend verdrahtet“ werden müssen: Eingänge sind dabei i.a. auf der linken Seite und Ausgänge auf der rechten Seite des Funktionssymbols angeordnet.

#### **Beachten:**

Wenn an einem Eingangsanschluss hinter dem Eingangsnamen ein Wert in Klammern steht, so übernimmt LabVIEW diesen Wert ( $\equiv$  **Standard-, Grund- oder Default-Wert**), wenn der Anwender nicht explizit einen anderen Wert dort festlegt bzw. anschließt.

#### **Beispiel:**

Anschluss **Baudrate (9600)**: wird hier kein anderer Wert angeschlossen, so parametrisiert LabVIEW die Schnittstelle automatisch mit der Baudrate 9600 Bd.

Schauen wir uns nun die Eingänge dieser Funktion etwas näher an:

### **Eingänge (an der oberen, linken und unteren Seite des Funktionssymbols):**

#### **Abschlusszeichen aktivieren**

Hier kann festgelegt werden, ob die Datenübertragung mit einem besonderen Abschlusszeichen arbeitet.

Das ist bei uns nicht der Fall, also schließen wir hier eine (logische) **Konstante** an.

**Kontextmenü zu diesem Eingang aufrufen, 'Erstellen' anwählen und dann 'Konstante' anklicken.**

Danach ändern Sie den Wert dieser Konstanten mit dem „**Hand-Cursor**“ auf 'F' ( $\equiv$  logisch FALSE).

#### **Endzeichen(0xA=''\n'= LF)**

Da wir, wie zuvor erwähnt, kein besonderes Endzeichen bei der Datenübertragung verwenden, wird hier nichts angeschlossen.

#### **Timeout(10s)**

Die hier angeschlossene Timeout-Zeit legt fest, wie lange der Empfänger auf ein ankommendes Zeichen wartet, ehe die Empfangs-Funktion mit einem Fehler abbricht. Mit anderen Worten: wenn Zeichen empfangen werden sollen und hierbei nach 10 Sekunden kein erwartetes Zeichen mehr kommt ( $\equiv$  Timeout), so wird ein Fehler gemeldet.

Da unser Mikrocontroller-System die beiden Messwert-Bytes aber sehr schnell hintereinander sendet, können wir die Timeout-Zeit hier auf 4 Sekunden herunter setzen. Wir schließen an diesem Anschluss also eine Konstante an:

**Kontextmenü zu diesem Eingang aufrufen, 'Erstellen' anwählen und dann 'Konstante' anklicken.**

Danach ändern Sie den Wert dieser Konstanten mit dem „Hand-Cursor“ auf '4000' (die Timeout-Zeit wird immer in ms angegeben).

### VISA-Ressourcenname

Hier wird nun die serielle Schnittstelle des PCs/LapTops festgelegt, über die die Kommunikation ablaufen soll.

Da diese Eingabe, je nach verwendetem Rechner, variabel gestaltet werden soll, schließen wir hier ein **Bedienelement** an:

**Kontextmenü zu diesem Eingang aufrufen, 'Erstellen' anwählen und dann 'Bedienelement' anklicken.**

Die konkrete Auswahl der Schnittstelle erfolgt dann später auf dem Frontpanel. Ändern Sie den Namen des Bedienelementes in '**Schnittstelle**'.

### Baudrate (9600)

Auch die Baudrate wollen wir flexibel einstellen können und schließen daher dort ebenfalls ein Bedienelement an, das dann später über das Frontpanel eingestellt wird.

Ändern Sie den Namen des Bedienelementes in '**Baudrate**'.

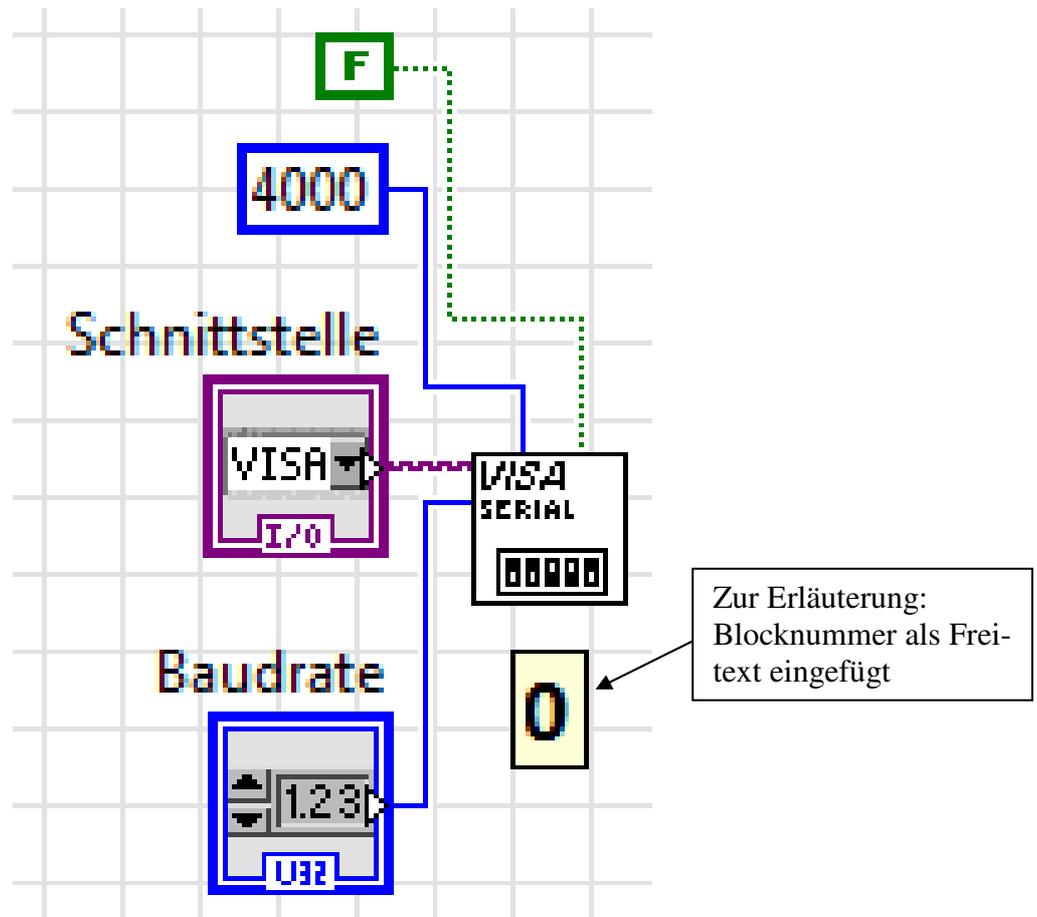
Die anderen Eingänge lassen wir hier zunächst unbeschaltet, da die festgelegten Standardwerte für unsere Kommunikation schon passend gewählt sind:

- **Datenbits (8):** wir empfangen 8 Datenbits pro UART-Zeichen.
- **Parität (0: keine):** das Mikrocontroller-System sendet die Daten ohne Paritätssicherung aus.
- **Fehler (Eingang, kein Fehler):** hier würde eine Fehlermeldung von einer Vorstufe eingespeist und weiter zum Fehlerausgang durchgereicht werden. Da wir hier aber keine Vorstufe haben, wird hier nichts angeschlossen.
- **Stoppbits (10: 1 Bit):** das Mikrocontroller-System arbeitet mit einem Stopp-Bit.

- **Ablaufsteuerung (0: keine):** bei der Datenübertragung wird kein Handshake, XON/XOFF-Verfahren o.ä. verwendet.

(Die beiden Ausgänge auf der rechten Seite des Funktionssymbols bleiben zur Zeit noch unbeschaltet)

Um das Blockdiagramm „optisch nun schön aufzuräumen“, klicken Sie einfach auf das **‘Besen-Ikon’** in der oberen Ikon-Leiste und das Ergebnis sollte so aussehen, **Abb.5.1.3:**



**Abb.5.1.3: Die erste VISA-Funktion auf dem Blockdiagramm, schön aufgeräumt (Block Nr. 0)**

(Wenn Ihnen diese Anordnung nicht gefällt, so können Sie diese ganz einfach mit **‘Strg+Z’** rückgängig machen)

Damit haben Sie nun den Block Nr. 0 aus unserem Gesamtdiagramm (s. am Anfang des Projektes) erstellt.

Speicher Sie nun das VI mit **‘Strg+S’** ab.

Als nächstes muss sinnvoller Weise, bevor überhaupt irgendwelche Zeichen empfangen werden, der Schnittstellenempfangs-Puffer von ev. noch vorhandenen unerwünschten „Rest-Zeichen“ der vorherigen Übertragung gelöscht werden.

Dazu dient die VISA-Funktion:

## VISA: I/O-Puffer entleeren

(BD\Instrumenten-I/O\Seriell\VISA: I/O-Puffer leeren)

Die Kontexthilfe zeigt, was diese Funktion bewirkt, **Abb.5.1.4:**

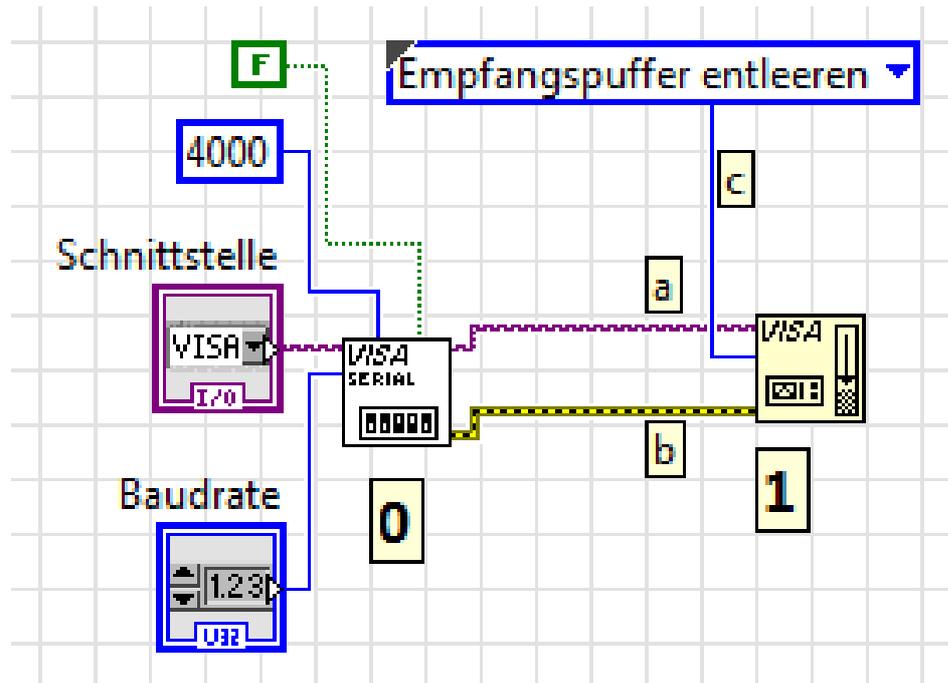


**Abb.5.1.4: Die Beschreibung zur Funktion: 'VISA: I/O-Puffer entleeren'**

Mit dieser Funktion können unterschiedliche Puffer der jeweiligen VISA-Ressource auf verschiedene Arten und Weisen entleert werden.

Diese hier gemeinte VISA-Ressource ist natürlich unsere serielle Schnittstelle und das müssen wir der Funktion an ihrem Eingang **'VISA-Ressourcenname'** auch mitteilen.

Damit sind wir dann auch schon bei der (recht einfachen) Verdrahtung dieser Funktion, **Abb.5.1.5:**



**Abb.5.1.5: Die Verdrahtung der VISA-Funktionen (Block ①)**

Nun kommt eine weitere wesentliche Eigenschaft der VISA-Funktionen zum Tragen: VISA-Ressourcenamen und Fehlermeldungen von Funktionen werden bei jeder Funktion vom Eingang immer zum entsprechenden Ausgang durchgereicht, so dass VISA-Funktionen einfach „hintereinander“ geschaltet werden können, wenn sie die selbe Ressource (hier also: die selbe serielle Schnittstelle) benutzen.

Wir verbinden daher den Ausgang **‘VISA-Ressourcenname (Ausgang)’** der Funktion **‘VISA: Seriellen Port konfigurieren’** mit dem Eingang **‘VISA-Ressourcenname’** der Funktion **‘VISA: I/O-Puffer entleeren’**  $\equiv$  **Verbindung (a)** in Abb.5.1.5.

Entsprechend wird der **Fehlerausgang** von **‘VISA: Seriellen Port konfigurieren’** mit dem **Fehlereingang** von **‘VISA: I/O-Puffer entleeren’** verbunden  $\equiv$  **Verbindung (b)**.

Am Eingang **‘Maskieren (16)’** der Funktion schließen wir eine Konstante an ( $\equiv$  **Verbindung (c)**) und wählen als Wert für die Konstante mit dem **‘Hand-Cursor’**: **‘Empfangspuffer entleeren’**, sofern dieses nicht schon automatisch von Anfang an eingetragen ist.

Mit dieser Festlegung teilen wir der Funktion mit, dass der Empfangspuffer (der seriellen Schnittstelle) vor Beginn des Datentransfers gelöscht werden soll.

(Die Löschung des Sendepuffers wird hier noch nicht durchgeführt, da unser VI ja nichts aus-senden soll)

Damit ist der Block ① in unserem Blockdiagramm fertig verdrahtet.

Speichern Sie das VI mit **‘Strg+S’** ab.

Nun kann der **Empfang von Daten** programmiert werden wobei folgendes gelten soll:

Die Temperaturmessdaten vom Mikrocontroller-System sollen permanent fortlaufend empfangen werden und zwar so lange, bis der Anwender auf einen 'Stopp-Knopf' drückt und den Ablauf des VIs damit beendet.

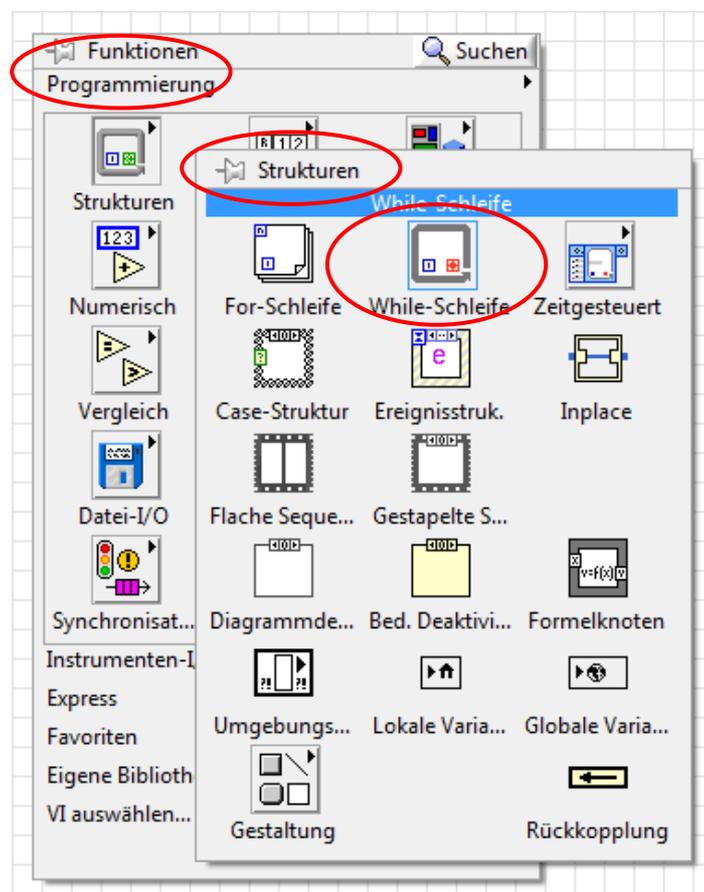
Solch einen Programmkonstrukt:

*„Mache permanent solange etwas, bis eine Stopp-Bedingung eintritt“*

gibt es in vielen Programmiersprachen und diese Konstruktion wird i.a. als **while-Schleife** bezeichnet, z.B. in 'C'.

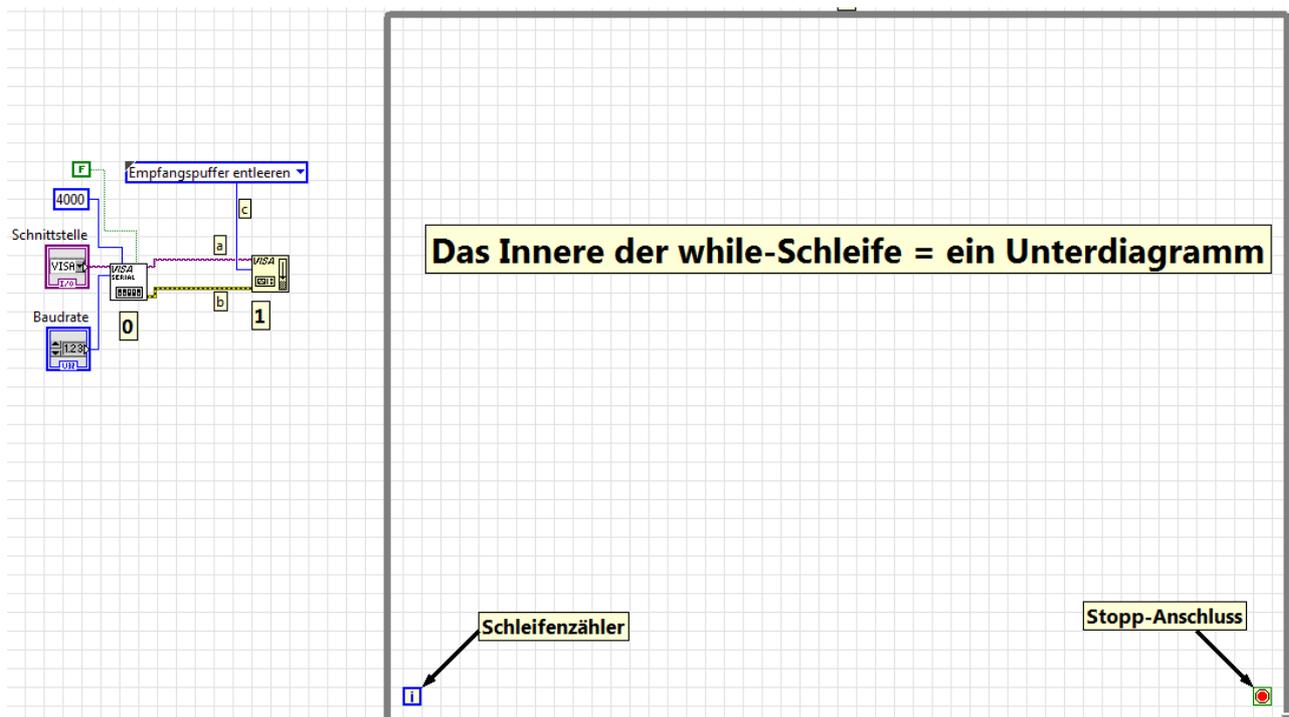
Natürlich existiert so etwas auch in LabVIEW und „es“ ist zu finden unter den so genannten 'Programmstrukturen', **Abb.5.1.6:**

### BD\Programmierung\Strukturen\While-Schleife



**Abb.5.1.6: Der Weg zu den Programmstrukturen und zur while-Schleife**

Wählen Sie daher als nächstes eine solche while-Schleife aus und ziehen Sie den Kasten mit dem Cursor auf dem Blockdiagramm recht großzügig auf, **Abb.5.1.7:**



**Abb.5.1.7: Die while-Schleife und ihre Elemente**

Alles, was nun im Inneren dieser while-Schleife programmiert wird (das so genannte **Unterdiagramm**), wird solange permanent ausgeführt, bis eine **Stopp-Taste** gedrückt wird, die am **Stopp-Anschluss** angeschlossen ist.

Der **Schleifenzähler i** gibt dabei an, wie oft die while-Schleife bereits durchlaufen wurde, d.h. der Wert von i wird nach jedem Schleifendurchlauf um '1' erhöht.

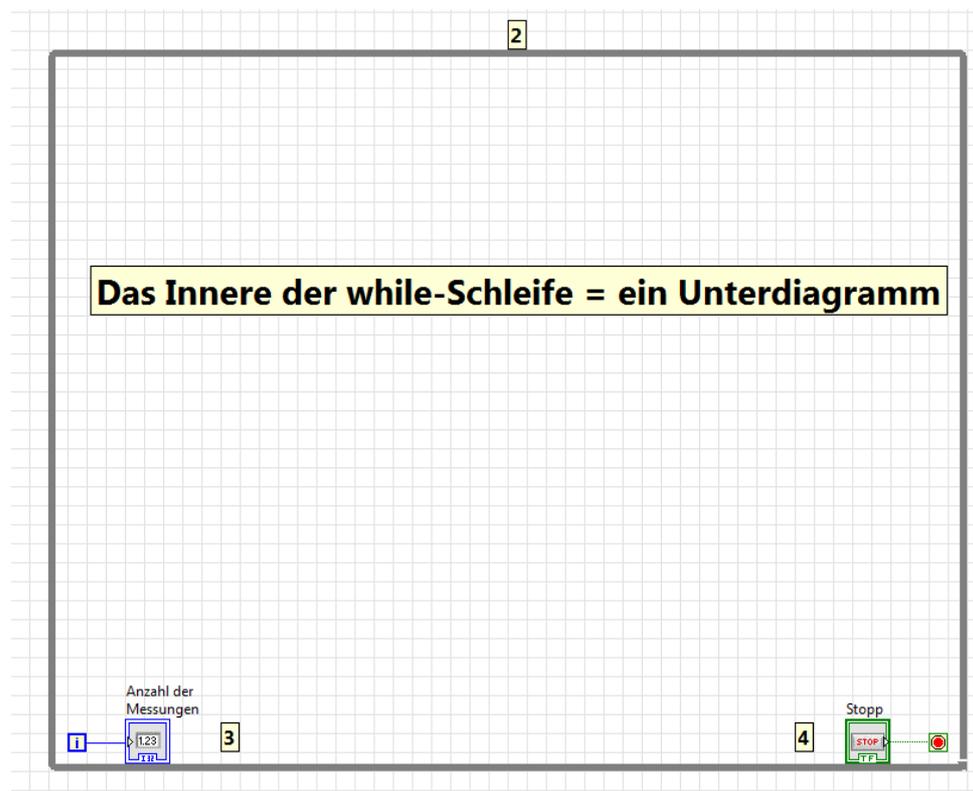
Diesen Wert kann man z.B. dazu benutzen, um anzuzeigen, wie oft ein Messwert vom Mikrocontroller-System empfangen wurde (da ja später bei jeder Abarbeitung des Unterdiagramms der while-Schleife genau ein Messwert empfangen und verarbeitet wurde).

Als erstes müssen wir nun einen **Stopp-Taster** und eine **Anzeige für den Schleifenzähler** in die while-Schleife einbauen:

Wählen Sie aus dem Kontextmenü des Schleifenzählers i den Punkt **Anzeigeelement erstellen** aus und nennen Sie diese Anzeige um in **Anzahl der Messungen**.

Wählen Sie aus dem Kontextmenü des Stopp-Anschlusses den Punkt **Bedienelement erstellen**: es wird automatisch ein Stopp-Taster eingefügt.

Damit haben wir die Blöcke ②, ③ und ④ in unserem Gesamt-Blockdiagramm fertig erstellt, **Abb.5.1.8**:



**Abb.5.1.8: Die Kernelemente der while-Schleife sind verdrahtet (Blöcke ②, ③ und ④)**

Und nun geht's ans Ausfüllen der while-Schleife, an das konkrete Erstellen des Unterdiagramms, also an den Empfang und die Auswertung der Messdaten  
Die dazu benötigte VISA-Empfangs-Funktion heißt

### VISA: Lesen

(BD\Instrumenten-I/O\Seruell\VISA: Lesen)

und wird innerhalb der while-Schleife platziert.

In der **Abb.5.1.9** (Kontexthilfe) ist die Kernbelegung zu sehen:

**VISA: Lesen (Funktion)**

**Übergeordnete Palette:** [VISA-VIs und -Funktionen](#)

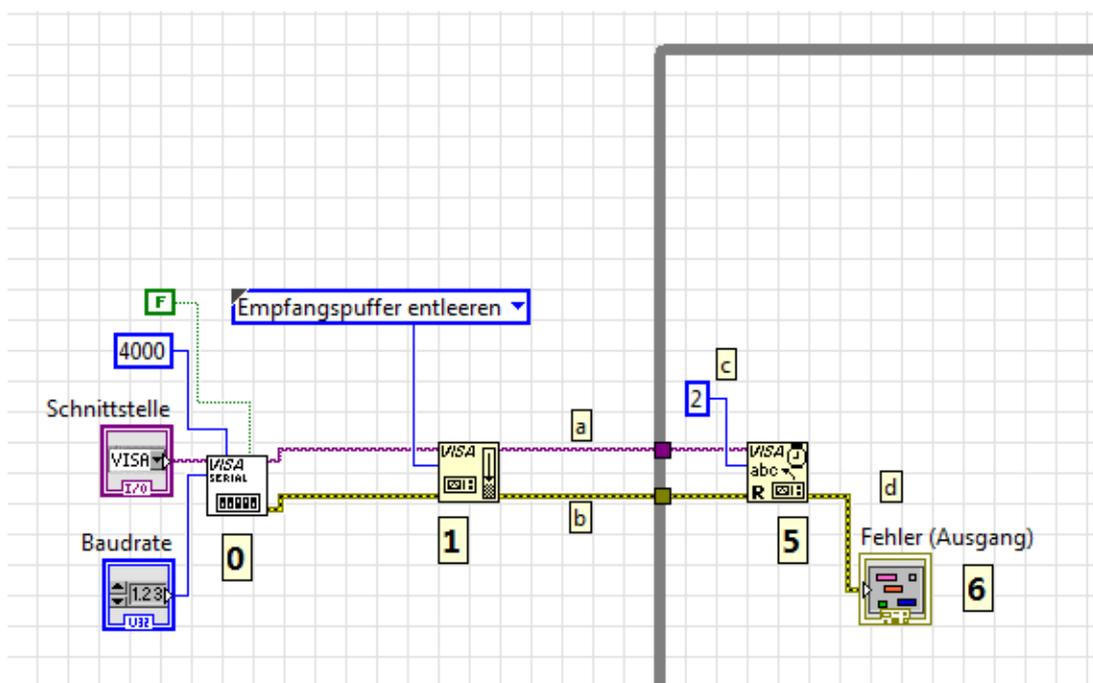
**Erfordert:** Base Package

Gibt die angegebene Anzahl von Bytes von der Ressource, die durch **VISA-Res** Gerät oder eine Schnittstelle handeln.

[Details](#) [Beispiele](#)

**Abb.5.1.9: Die Empfangsfunktion: 'VISA: Lesen'**

Die notwendige Verdrahtung ist auch hier wieder denkbar einfach, **Abb.5.1.10:**



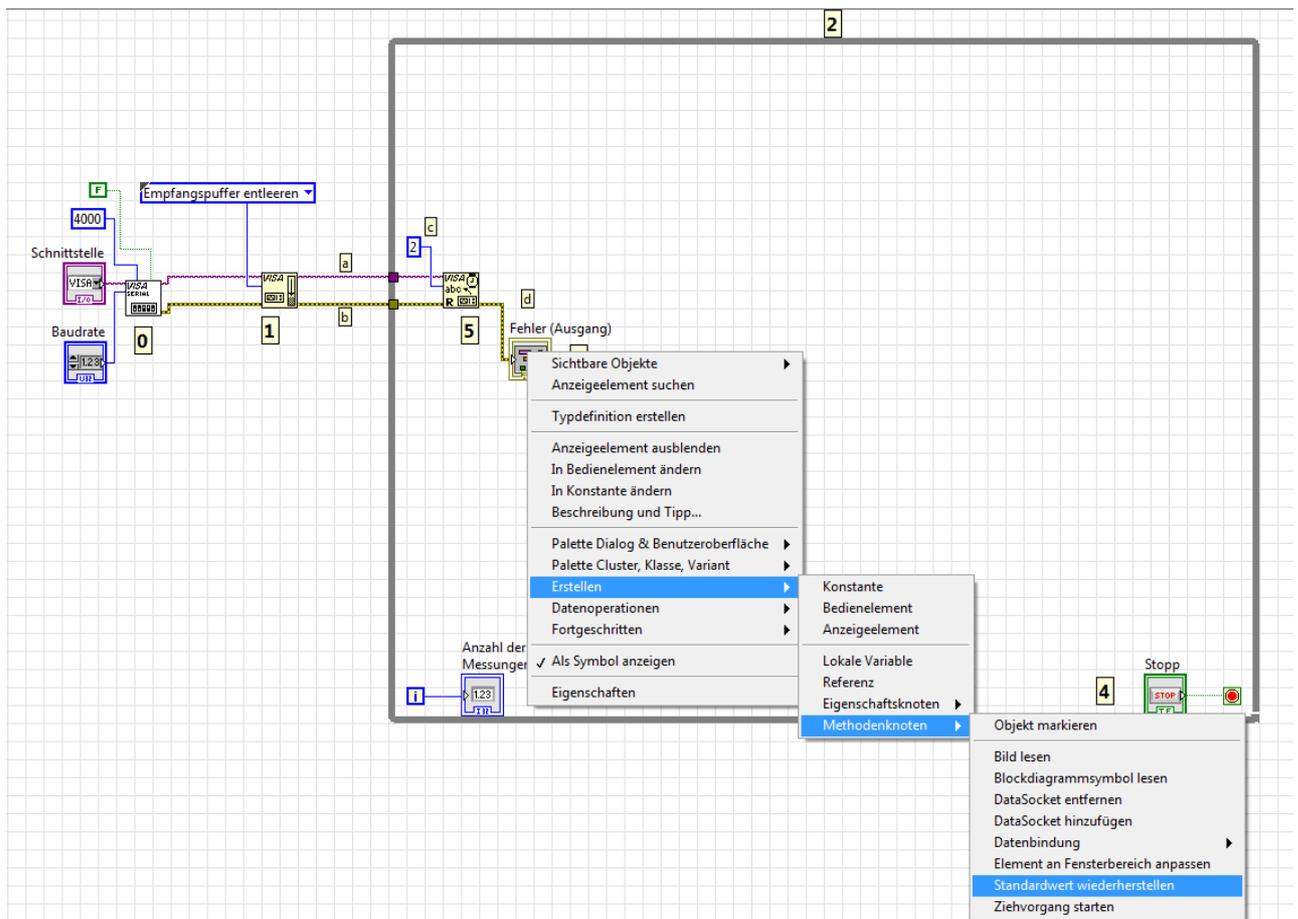
**Abb.5.1.10: Die Verdrahtung der Empfangsschleife (Blöcke ⑤ und ⑥)**

- Der Ressourcenname-Eingang von 'VISA: Lesen' wird mit dem Ressourcennamen-Ausgang der Puffer-Leeren-Funktion verbunden (Verbindung (a)).

- Der Fehlereingang wird mit dem Fehlerausgang der Puffer-Leeren-Funktion verbunden (Verbindung (b)).
- An den Anschluss **Byte-Anzahl** wird eine Konstante mit dem Wert **2** angeschlossen: hierüber wird festgelegt, wie viele Bytes empfangen werden sollen, in unserem Fall eben 2 Stück (Verbindung (c)).
- Ausgangsseitig wird zunächst **am Fehlerausgang ein Anzeigeelement** angeschlossen (Verbindung (d), Block 6).  
Hierüber wird dann später auf dem Frontpanel angezeigt, wenn beim Empfang der Bytes ein Empfangsfehler aufgetreten ist, z.B. ein Time-Out-Fehler, wenn vom Mikrocontroller-System keine Daten mehr gesendet werden.  
Den Namen des Anzeigeelementes **Fehler (Ausgang)** lassen wir unverändert.

Diese Fehlermeldung sollte beim Start des VIs allerdings als Erstes **gelöscht werden**, damit „alte“ Fehlermeldungen nicht mehr angezeigt werden.

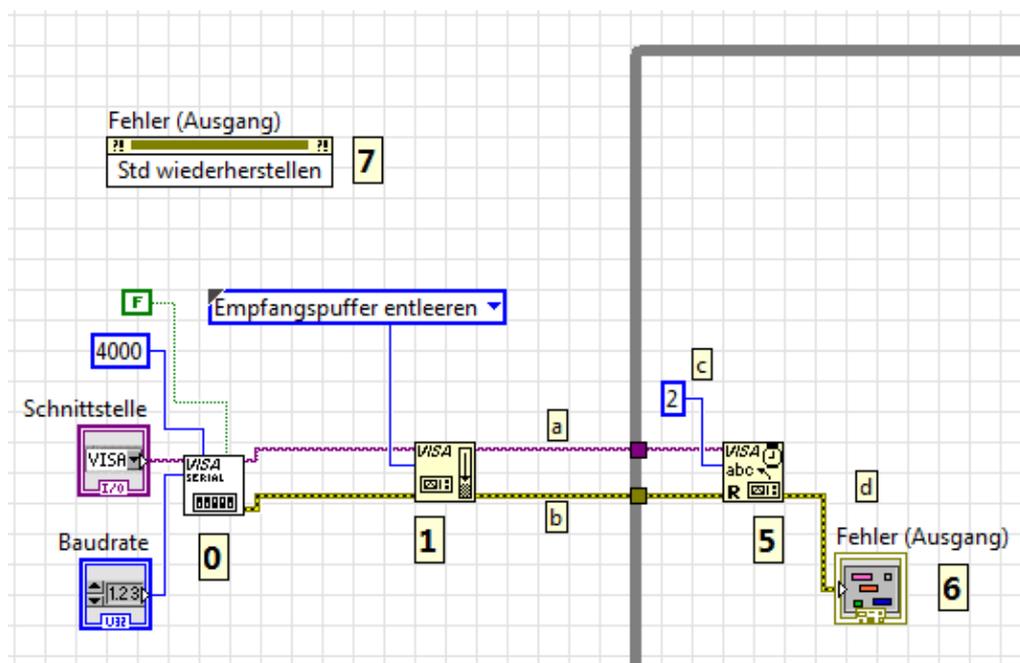
Um diese Löschung durchzuführen, geht man wie folgt vor, **Abb.5.1.11**:



**Abb.5.1.11: Der Weg zum Löschen der Fehlermeldung**

Sie öffnen das Kontextmenü zu diesem Anzeigeelement und wählen nacheinander die Untermenü-Punkte: **Erstellen\Methodenknoten\Standardwert wiederherstellen**. (Über Methoden- und Eigenschaftsknoten können bestimmte Eigenschaften von LabVIEW-Elementen per Programm eingestellt werden. In unserem Fall bedeutet: 'Standardwert wiederherstellen' die Löschung der Fehlermeldung im Anzeigeelement)

Den so erzeugten Methodenknoten platzieren wir außerhalb der while-Schleife (Block 7) und so wird garantiert, dass die Fehlermeldung bei jedem Neu-Start des VIs auch wirklich gelöscht wird, **Abb.5.1.12**:



**Abb.5.1.12: Das Löschen der Fehlermeldung außerhalb der while-Schleife (Block 7)**

Damit sind die Blöcke 5 - 7 des Blockdiagramms fertig erstellt.

Speichern Sie das VI ab.

Am Ausgang 'Lesepuffer' des VISA-Empfangsfunktion (s. Abb.5.1.9) liegen nun die beiden empfangenen Bytes zur Weiterverarbeitung an.

Hier ist allerdings noch ein **kleines Problem** vorhanden: LabVIEW arbeitet an dieser Stelle mit Strings (Zeichenketten), d.h. die beiden empfangenen Bytes werden zunächst als ein String mit zwei Elementen ( $\equiv$  Array) interpretiert und an diesem Ausgang auch so ausgegeben. Wir müssen also zunächst zwei Aktionen ausführen, um die beiden Bytes wieder als „echte“ Zahlen (Messwerte) zu erhalten:

- Umwandlung des String(-Arrays) in ein Byte-Array mit zwei Zahlenwerten ( $\equiv$  Zahlen-Array).

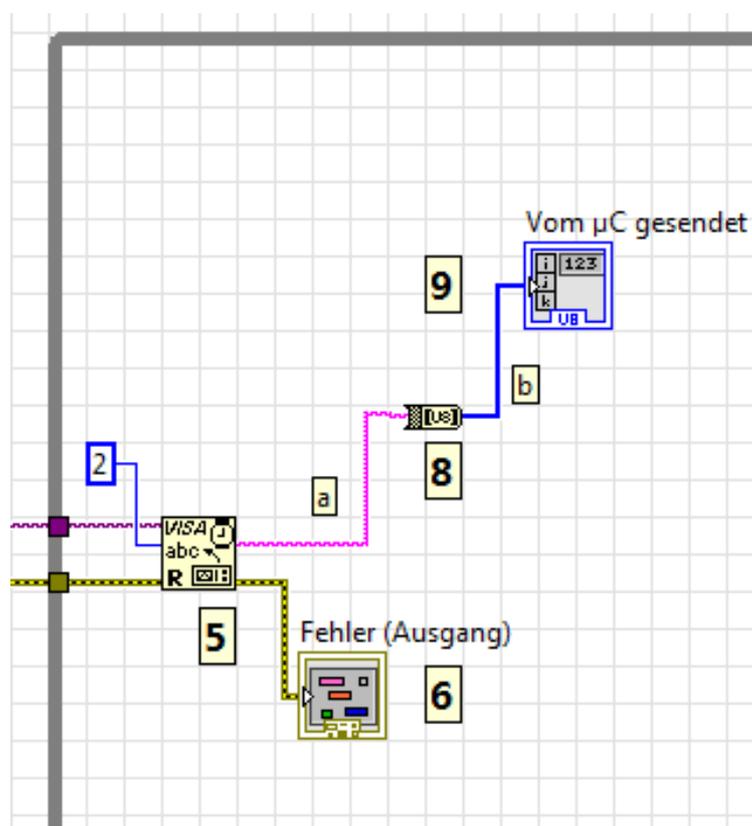
- Einzelner Zugriff auf die beiden Zahlen-Array-Elemente, damit diese geeignet weiter verarbeitet werden können.

Wir erweitern unser Blockdiagramm also zunächst um die Funktion:

### Nach Byte-Array

(BD\Programmierung\String\String-/Array-/Pfadkonvertierung\String nach Byte-Array)

Die Verdrahtung ist in **Abb.5.1.3** dargestellt:



**Abb.5.1.13: Die Umwandlung vom String zum Zahlenarray (Blöcke ⑧ und ⑨)**

- Ausgang 'Lesebuffer' von 'VISA: Lesen' an den Eingang der Funktion 'String nach Byte-Array' (Verbindung (a)).
- An den Ausgang der Funktion 'String nach Byte-Array' schließen wir ein Anzeigeelement an (Block ⑨), das wir 'Vom  $\mu\text{C}$  gesendet' nennen (Verbindung (b)). Hier werden dann später auf dem Frontpanel die beiden empfangenen Bytes als „echte“ Zahlen in Form eines Arrays mit zwei Elementen angezeigt.

Die Blöcke ⑧ und ⑨ des Blockdiagramms sind damit erledigt.

Nun benötigen wir noch eine Funktion, mit der man gezielt auf die beiden Elemente des Zahlenarrays zugreifen, die Zahlen also einzeln aus dem Array auslesen kann.

Das wird bewerkstelligt durch die Funktion:

## Array indizieren

(BD\Programmierung\Array\Array indizieren)

Was diese Funktion macht, zeigt die **Abb.5.1.14**:

---

### Array indizieren (Funktion)

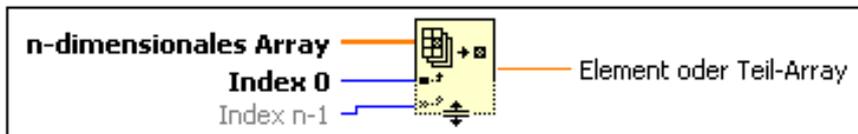
Übergeordnete Palette: [Array-Funktionen](#)

Erfordert: Base Package

Gibt das **Element oder Teil-Array** des Eingangs **n-dimensionales Array** am ar

Wenn ein Array an diese Funktion angeschlossen wird, zeigt diese automatisch **Ind** das Sie mit **n-dimensionales Array** verbunden haben. Zum Hinzufügen zusätzlich [Sie die Funktion mit der Maus auf](#). Die Standarddatentypen für diese polymorphe Fu

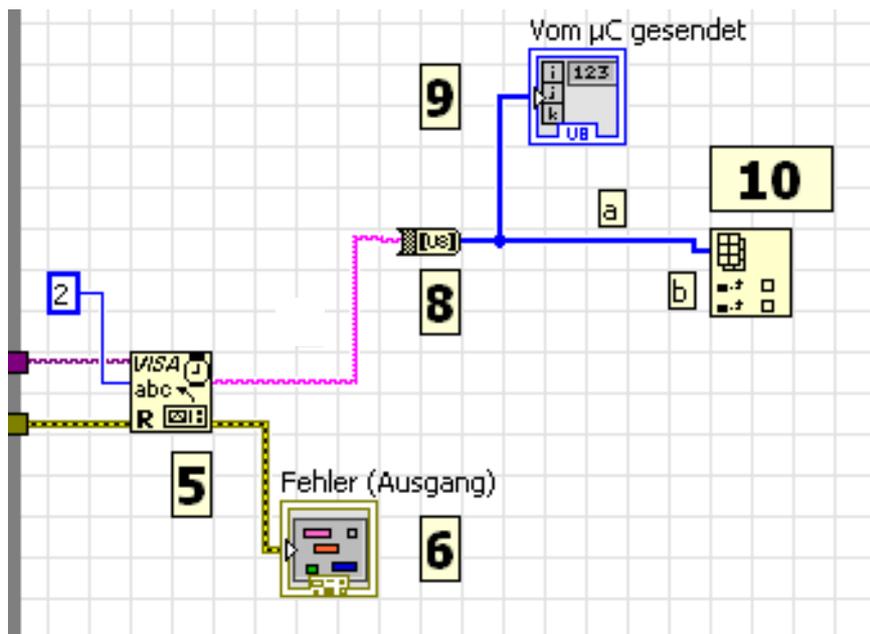
[Details](#)



#### Abb.5.1.14: Die Funktion 'Array indizieren'

Am Eingang 'n-dimensionales Array' dieser Funktion wird ein n-dimensionales Array angeschlossen und am Ausgang 'Element oder Teil-Array' stellt diese Funktion die Array-Elemente einzeln zur Weiterverarbeitung zur Verfügung.

Verbinden Sie also, **Abb.5.1.15**:



**Abb.5.1.15: Der Zugriff auf die einzelnen Elemente des Arrays (Block ⑩)**

- Den Ausgang der Funktion 'Nach Byte-Array' mit dem Eingang 'n-dimensionales Array' der Funktion 'Array indizieren' (Verbindung (a)).
- Ziehen Sie die Funktion 'Array indizieren' auf zwei Ein-/Ausgänge auf (Punkt (b)).
- Die restlichen Eingänge bleiben unbeschaltet.
- An den beiden Ausgängen der Funktion 'Array indizieren' können nun die beiden (Messwert)Bytes abgegriffen und weiter verarbeitet werden.

Mit diesem Block Nr. ⑩ ist der eigentliche Empfang (und die Konvertierung) der Messwerte beendet und wir können uns nun dem letzten Punkt des LabVIEW-VIs zuwenden, der ...

### Auswertung und Darstellung der Messwerte

Wie bereits in Kapitel 2.1 erwähnt und erläutert, überträgt das Mikrocontroller-System die Messdaten im **Zweier-Komplement** und d.h.:

- Wenn das zweite Messdaten-Byte ( $\equiv$  Vorzeichen-Byte) '0' ist, so ist der eigentliche Messwert im ersten Byte positiv und dieser Wert braucht nicht weiter verändert zu werden.
- Hat das zweite Byte ( $\equiv$  Vorzeichen-Byte) dagegen den Wert '255', so ist der Messwert (die Temperatur) negativ und im ersten Byte ist dieser Messwert in Form eines Zwei-

er-Komplements enthalten.

Dieser Wert muß daher umgewandelt werden: Invertierung aller 8 Bits des Byte, Addition von '1' und abschließende Multiplikation mit '-1', um den korrekten negativen Zahlenwert zu erhalten.

Diese Unterscheidung 'positiver/negativer Temperaturmesswert' anhand des Wertes des zweiten Bytes kann in klassischen Programmiersprachen (wie z.B. in 'C') durch eine **if-Abfrage** oder durch eine **case-Struktur** realisiert werden.

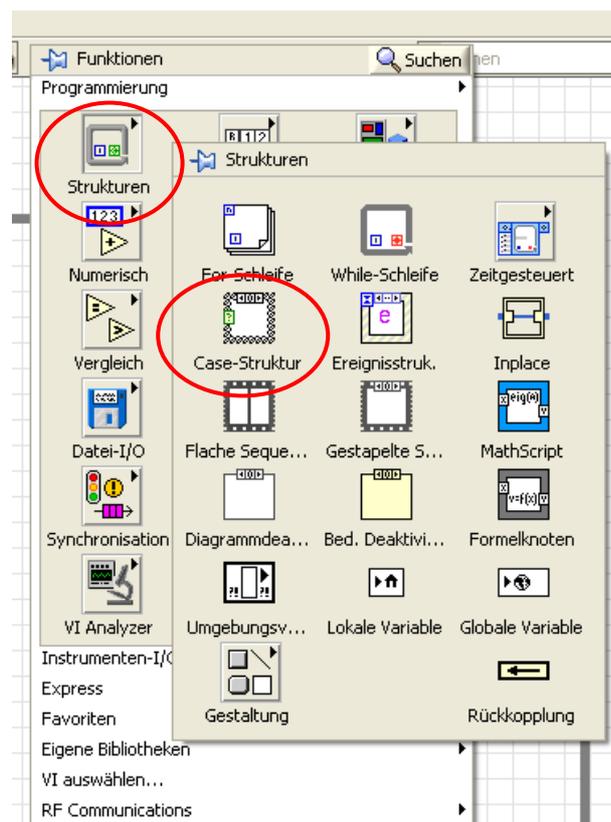
In LabVIEW gibt es dazu (nur) eine **case-Programmstruktur**, die jedoch eine „traditionelle“ if-Auswertung als Spezial-Unterfall enthält.

Sie finden die

## Case-Struktur

unter:

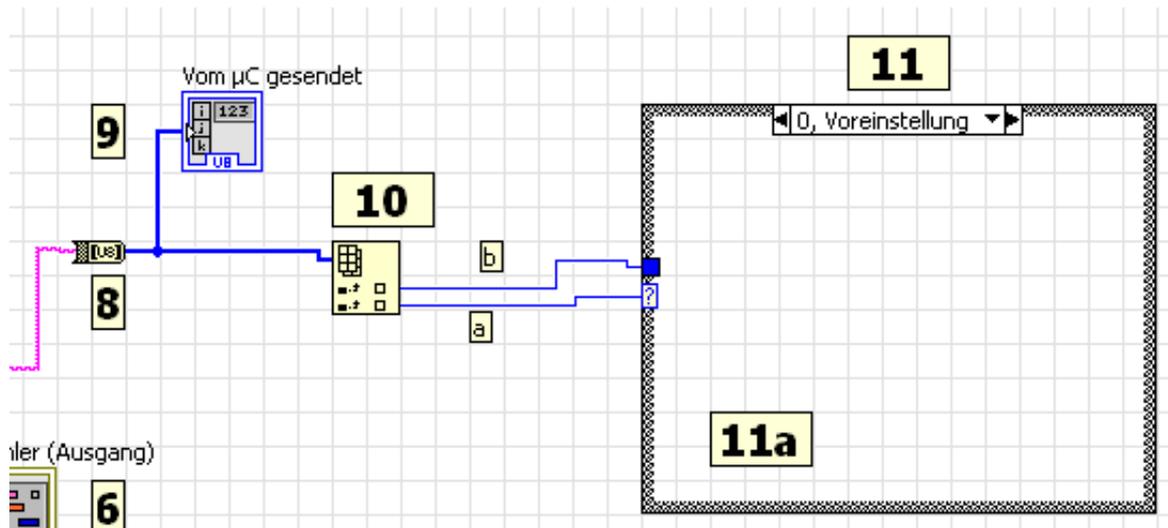
**BD\Programmierung\Strukturen\Case-Struktur, Abb.5.1.16:**



**Abb.5.1.16: Der Weg zur Case-Struktur**

Platzieren Sie solch eine case-Struktur nun rechts neben dem Block ⑩.

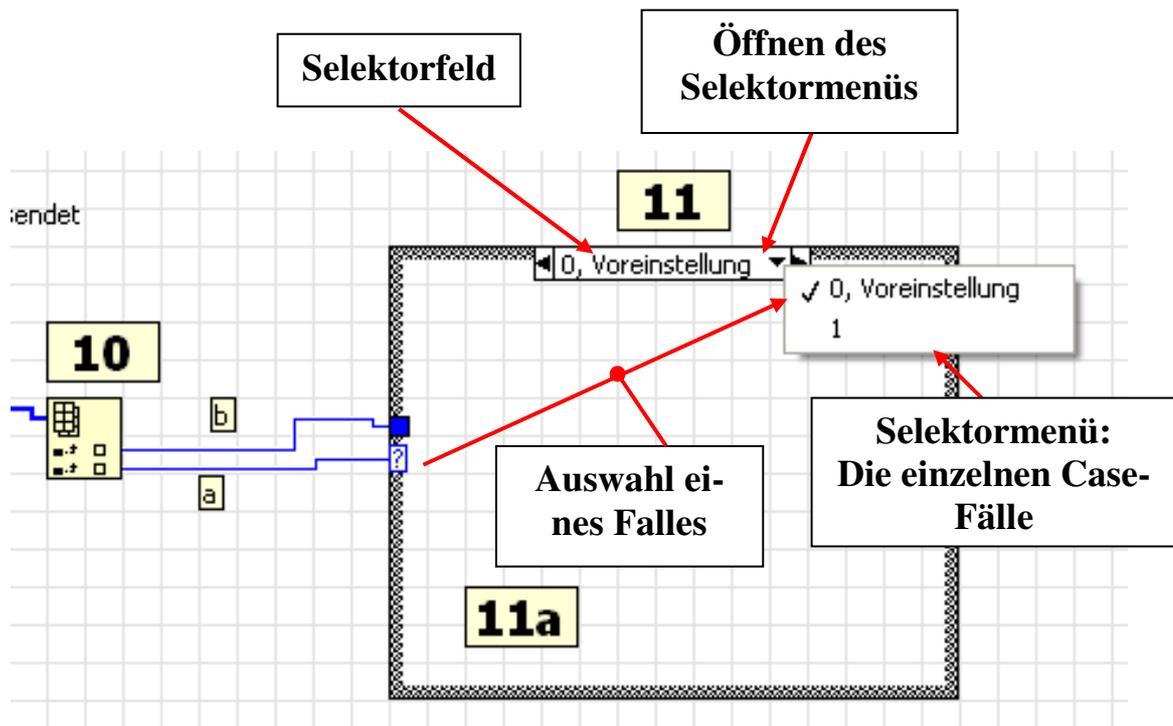
Die jetzt notwendigen Verbindungen erläutern wir mit Hilfe von **Abb.5.1.17:**



**Abb.5.1.17: Die äußere Verdrahtung der case-Struktur (Block 11)**

- Schließen Sie das zweite Messwert-Byte (das Vorzeichen-Byte  $\equiv$  unterster Ausgang der Funktion 'Array indizieren') an den '?'-Anschluss des Case-Rahmens an, Verbindung (a).  
Hierüber wird nachfolgend entschieden, welcher Case-Fall ausgewählt wird.
- Legen Sie den eigentlichen Messwert ( $\equiv$  oberster Ausgang der Funktion 'Array indizieren') einfach mit einem Draht 'irgendwo' genau bis an den äußeren Rand des Case-Rahmens, Verbindung (b)).  
Am Rand erscheint dort ein kleines blaues Quadrat.  
Von diesem Punkt aus kann der Messwert nun abgegriffen und innerhalb des Case-Rahmens beliebig weiter verarbeitet werden.

Mit dem Zahlenwert am '?'-Anschluss kann nun ausgewählt werden, welcher Bearbeitungsfall in der Case-Struktur abgearbeitet werden soll, **Abb.5.1.18:**



**Abb.5.1.18: Die Auswahl der einzelnen Case-Fälle**

Zu jedem auswählbaren Case-Fall gehört nun ein eigener Case-Rahmen, d.h. die gesamte Case-Struktur besteht aus mehreren übereinander gestapelten Rahmen, von denen allerdings nur der jeweils gerade oben liegende sichtbar ist.

In jedem dieser Rahmen kann nun ein entsprechendes Unterdiagramm programmiert werden, das die gewünschte Funktion für diesen Case-Fall realisiert.

Wenn nun die einzelnen Unterdiagramme entsprechend **im Blockdiagramm programmiert werden sollen**, so erfolgt die Umschaltung zwischen den Case-Rahmen über das so genannte **‘Selektorfeld’** im oberen Teil des Rahmens: hier wird dann auch angezeigt, welcher Case-Rahmen gerade aktuell dargestellt wird.

Klicken Sie daher einmal mit der linken Maus-Taste auf den kleinen nach unten gerichteten Pfeil im Selektorfeld: es öffnet sich ein kleines Fenster, in dem die zur Zeit vorhandenen Case-Fälle dieser Case-Struktur aufgeführt sind (Selektormenü).

In unserem Beispiel sind das die Fälle:

- 0, Voreinstellung
- 1

Der kleine Haken vor ‘0, Voreinstellung’ bedeutet, dass zur Zeit der Rahmen zu diesem Fall sichtbar ist und das zugehörige Unterdiagramm im Case-Rahmen eingegeben werden kann. Klicken Sie nun im Selektormenü auf die ‘1’, so wird der Case-Rahmen zum Case-Fall 1 angezeigt.

Zur Zeit sehen Sie hier keinen großen Unterschied, da ja beide Case-Rahmen noch leer sind.

Lediglich im Selektorfeld selber erkennen Sie, dass jetzt die Bezeichnung gewechselt hat, dass also jetzt der Rahmen zum Case-Fall 1 angezeigt wird.

### Wie wird nun beim **ablaufenden** LabVIEW-VI zwischen den einzelnen Case-Fällen unterschieden ?

Das ist nun ganz einfach zu verstehen, denn hier spielt der Wert am '?'-Anschluss ( $\equiv$  Auswahlanschluss) eine große Rolle.

Liegt dort der Wert '0' an, so wird der Case-Rahmen zum Fall '0, Voreinstellung' angearbeitet.

Liegt dagegen am '?'-Anschluss der Wert '1' an, so wird der Case-Rahmen zum Fall '1' abgearbeitet.

Für alle anderen Werte am '?'-Anschluss, also z.B. für die Werte 6, 123, -3456, 4, ... gilt: es wird in solchen Fällen immer der Case-Rahmen abgearbeitet, in dem das Schlüsselwort 'Voreinstellung' steht, hier also der Case-Rahmen '0, Voreinstellung'.

So ist sichergestellt, dass die Case-Struktur immer definiert bearbeitet wird, dass also kein Fall auftritt, in dem die Case-Struktur „ins Leere läuft“.

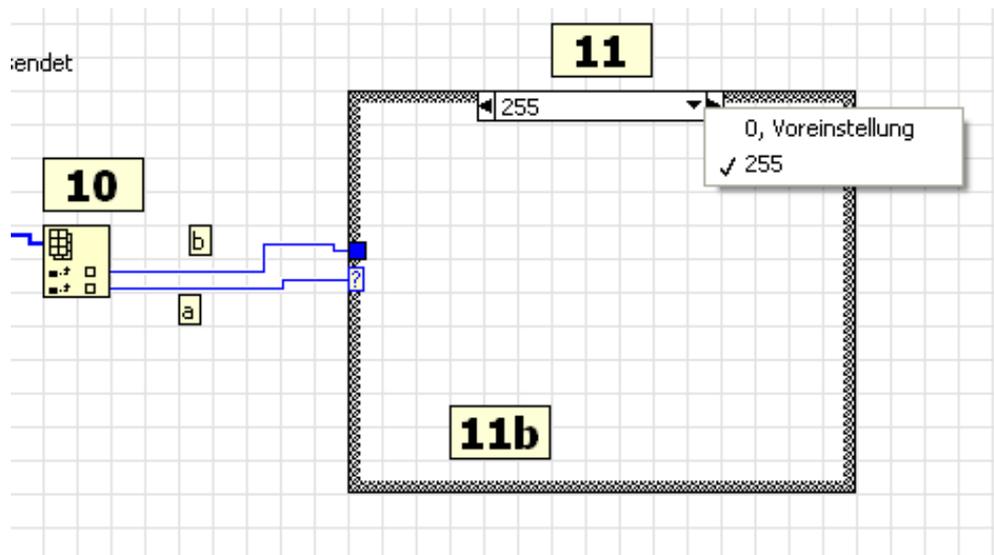
Kommen wir zurück zu unserer Anwendung:

Am '?'-Auswahl-Anschluss haben wir das Vorzeichen-Byte des Messwertes angeschlossen, das ja bekannter Maßen (nur) die beiden Werte '0' oder '255' annehmen kann.

Wir benötigen also eine Case-Struktur mit den Case-Fällen '0' und '255', wobei wir als 'Voreinstellungsrahmen' den '0'-Rahmen benutzen bzw. beibehalten können.

Wir müssen in unserer vorhandenen Case-Struktur also lediglich den Wert '1' im Selektorfeld umändern in den Wert '255'.

Wählen Sie dazu (über den kleinen nach unten gerichteten Pfeil im Selektorfeld) das Selektormenü aus, dort den Case-Rahmen zum Case-Fall '1' und ändern Sie im Selektorfeld die Zahl '1' in die Zahl '255', **Abb.5.1.19**:



**Abb.5.1.19: Die Änderung des Case-Falles '1' in den Case-Fall '255'**

Damit haben wir die Grundstruktur der Case-Struktur für unseren Anwendungsfall erstellt und wir müssen uns jetzt lediglich überlegen, wie denn die einzelnen Unterdiagramme in den beiden Case-Rahmen aussehen müssen.

Die gesamte Case-Struktur ist daher unser Block **11** im Blockdiagramm.

Speichern Sie das VI ab.

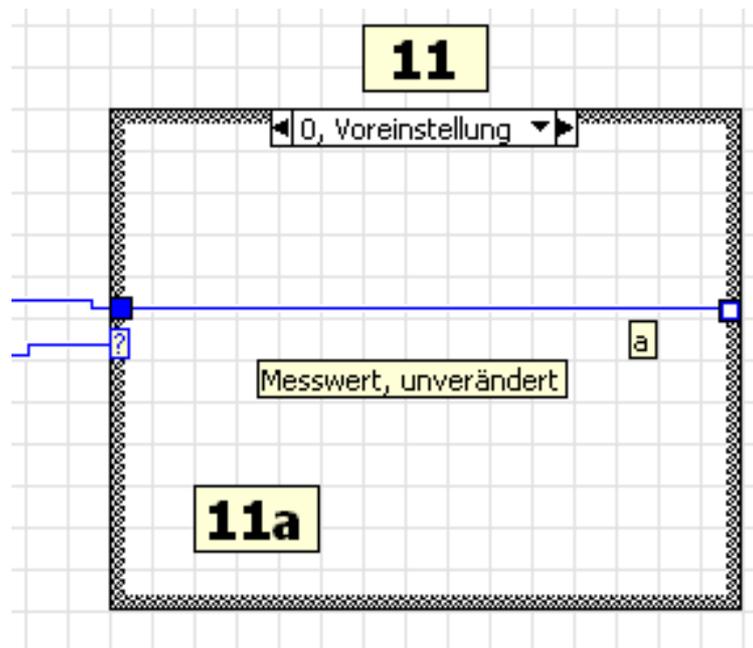
### Case-Fall '0, Voreinstellung'

In diesem Fall hat das Vorzeichen-Byte (am '?'-Anschluss) ja den Wert '0' und das bedeutet: der eigentlich Messwert ist positiv und braucht nicht weiter verändert zu werden.

Mit anderen Worten:

Sie wählen den **Case-Rahmen '0, Voreinstellung'** aus und verbinden den Messwert einfach von linken Rand des Case-Rahmens herüber bis zum rechten Rahmen (Verbindung (a)),

**Abb.5.1.20:**



**Abb.5.1.20: Der Case-Rahmen zum Fall '0, Voreinstellung' - der Messwert wird unverändert nach rechts hin wieder ausgegeben (Block **11**a)**

An der rechten Seite des Case-Rahmens erscheint nun ein kleines blaues **hohles** Quadrat ( $\equiv$  blaues Quadrat, ausgefüllt mit einem noch kleineren weißen Quadrat) und das bedeutet zweierlei:

- Der Messwert kann jetzt hier unverändert abgegriffen werden und „nach rechts hin“, außerhalb des Case-Rahmens, weiter verarbeitet werden.
- Das „hohle“ Quadrat zeigt allerdings an, dass die gesamte Case-Struktur noch nicht vollständig und das LabVIEW-VI so noch nicht ablauffähig ist: es fehlt nämlich noch die Verdrahtung dieses Ausganges im anderen Case-Rahmen, im '255er'- Case-

Rahmen.

Erst wenn dort etwas ordnungsgemäß an das blaue Ausgangsquadrat angeschlossen ist, kann man das VI problemlos starten.

Dieser '0er'-Case-Rahmen ist nun der Block **1 1 a** in unserem Blockdiagramm.

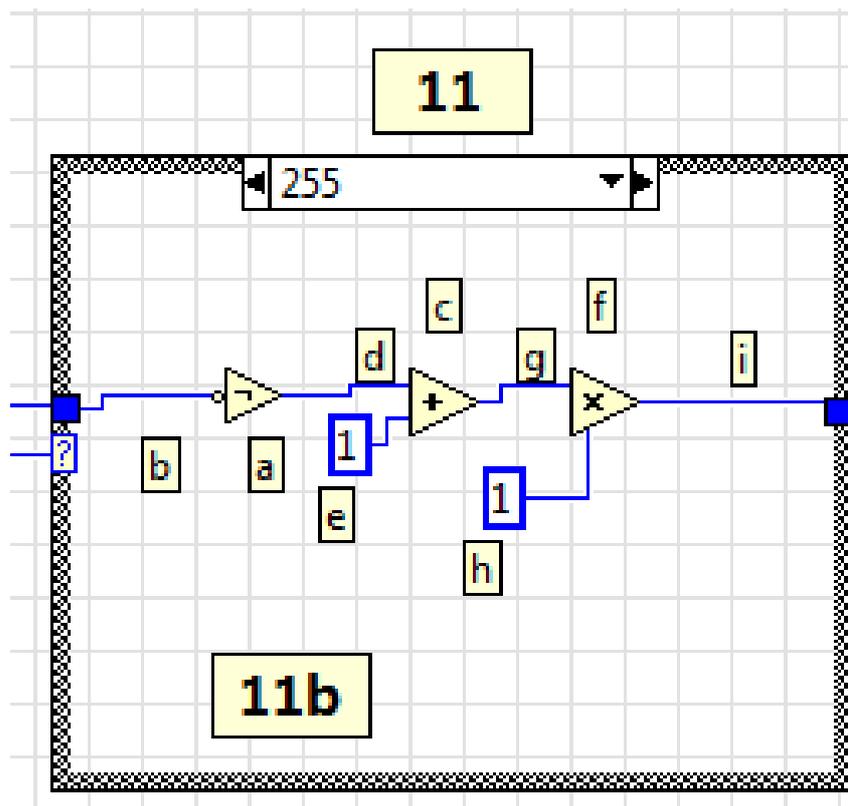
Wählen Sie nun mit Hilfe des kleinen nach unten gerichteten Pfeils im Selektorfeld den zweiten Case-Rahmen, den '255er'-Rahmen aus.

Dieses Unterdiagramm wird abgearbeitet, wenn am '?'-Anschluss der Wert '255' anliegt, wenn also der eigentliche Messwert negativ ist.

In diesem Fall muss aus dem Messwert das Zweier-Komplement gebildet (Invertierung und Addition von '1') und der so entstandene Zahlenwert noch mit '-1' multipliziert werden. Dazu benötigen wir also als erstes die **Invertierungsfunktion**, die man unter

### BD\Programmierung\Boolesch\NICHT

findet: **Block (a)** in **Abb.5.1.21**:



**Abb.5.1.21: Die Bildung des Zweier-Komplements im '255er'-Case-Rahmen**

Der weitere Zusammenbau des Unterdiagramms ergibt nun wie folgt:

- Verbindung des Messwert-Bytes ( $\equiv$  blaues Quadrat am linken Rand des Case-Rahmens) mit dem Eingang der Invertierungsfunktion, **Verbindung (b)**.
- Als nächstes benötigen wir die **Additionsfunktion**:

### BD\Programmierung\Numerisch\Addieren

#### Block (c)

- Den Ausgang der Invertierungsfunktion verbinden wir mit einem Eingang der Additionsfunktion, **Verbindung (d)**.
- An den anderen Eingang der Additionsfunktion schließen wir eine Konstante mit dem Wert '1' an, **Block (e)**.
- Die nächste Funktion, die eingefügt wird, ist die Multiplikation:

### BD\Programmierung\Numerisch\Multiplizieren

#### Block (f).

- Den Ausgang der Additionsfunktion verbinden wir mit einem Eingang der Multiplikationsfunktion, **Verbindung (g)**.
- Am anderen Eingang der Multiplikationsfunktion schließen wir eine Konstante mit dem Wert '-1' an und haben auf einmal ein „großes“ **Problem**:  
Die Konstante lässt sich nicht auf den Wert '-1' einstellen ! (nur auf den Wert '1').

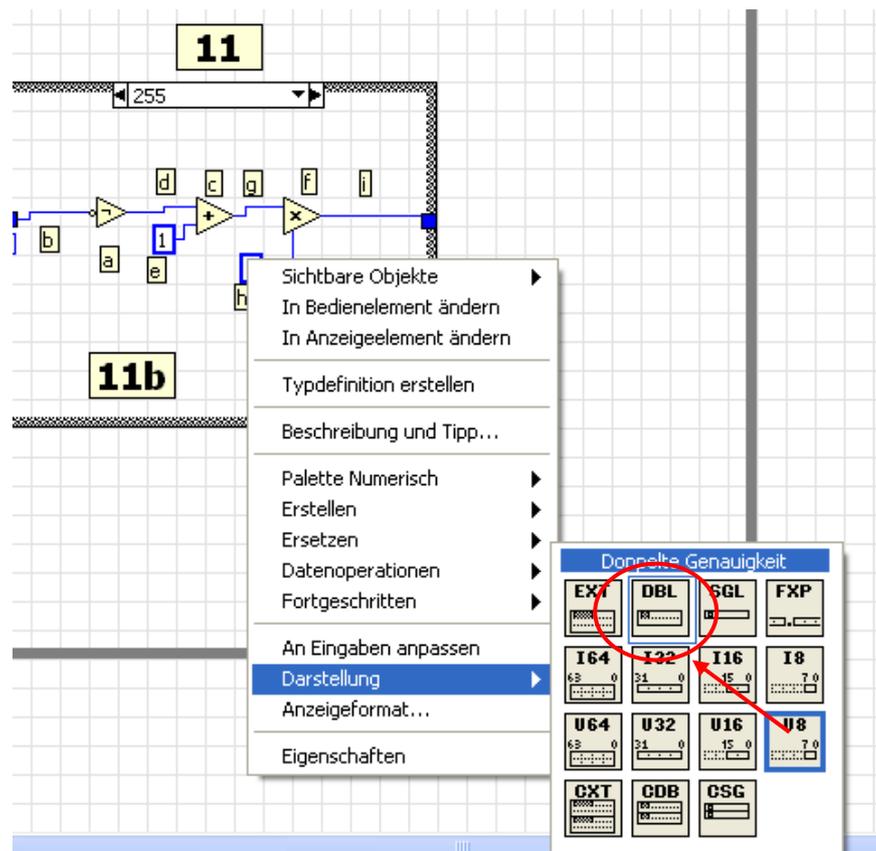
Der Grund und auch die Lösung dafür ist recht einfach:

Betrachten Sie sich die ganzen Verbindungsleitung und die eingefügten Konstanten vom blauen Quadrat an der linken Seite des Case-Rahmen bis hin zur Multiplikationsfunktion, so erkennen Sie, dass „alles schön in blau“ gehalten ist.

Und dieses Blau ist die Standardfarbe von LabVIEW für (positive) ganze Zahlen.

Auch die Konstante, die wir zuletzt an die Multiplikationsfunktion angeschlossen haben, wird von LabVIEW zunächst als positive, ganze Zahl angesehen und solch eine Zahl kann nun einmal nicht den Wert '-1' annehmen.

Wir müssen daher den **Datentyp** der Konstanten auf '**Fließkomma-Zahl, doppelte Genauigkeit**' ändern (das ist der normale Standard-Datentyp von LabVIEW), **Abb.5.1.22**:

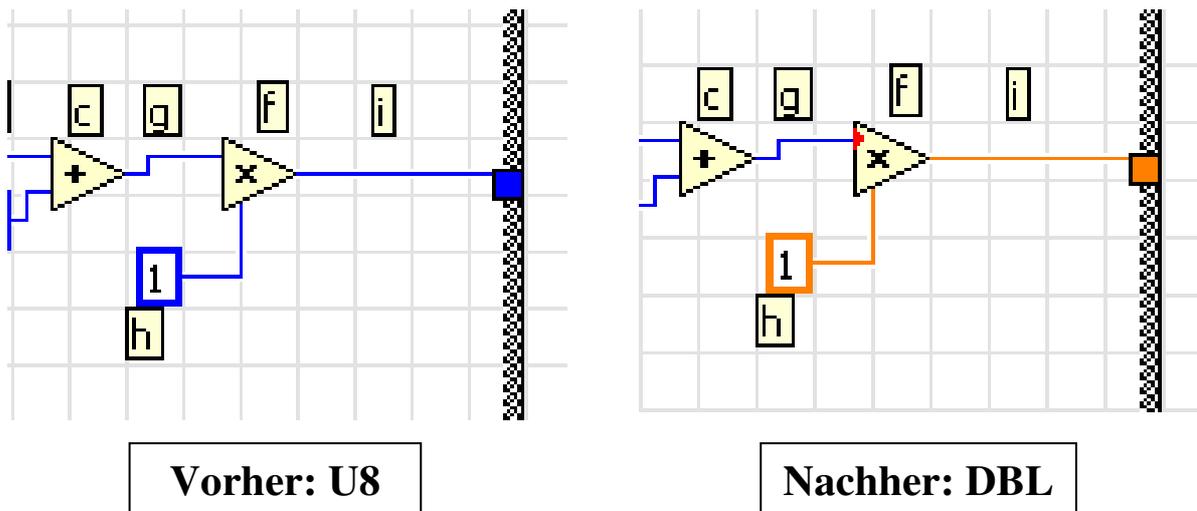


**Abb.5.1.22: Die Änderung des Datentyps von 'U8' nach 'DBL'**

Öffnen Sie dazu das zugehörige Kontextmenü zur Konstanten und wählen Sie darin den Punkt **'Darstellung'**.

Es öffnet sich ein weiteres Fenster, in dem alle Datentypen aufgeführt sind, die die Konstante nun annehmen kann.

Zur Zeit hat die Konstante den Datentyp **'U8'** (blau umrandet  $\equiv$  unsigned integer, 8 Bit breit). Klicken Sie jetzt einfach auf das Feld **'DBL'**  $\equiv$  Fließkomma-Zahl mit doppelter Genauigkeit. Sofort ändert sich im Blockdiagramm die Farbe der Umrandung der Konstanten und die Farbe der Verbindungsleitung zwischen der Konstanten und der Multiplikationsfunktion in ein „schönes orange“ (Orange ist die LabVIEW-Farbe für Fließkomma-Zahlen doppelter Genauigkeit), **Abb.5.1.23:**



**Abb.5.1.23: Der geänderte Datentyp der Konstanten**

Ändern Sie nun den Wert der Konstanten auf  $-1$  und verbinden Sie den Ausgang der Multiplikationsfunktion mit dem blauen hohlen Quadrat am rechten Rand des Case-Rahmens. Sie erkennen sofort: ab jetzt rechnet LabVIEW automatisch mit Fließkomma-Zahlen weiter. Die Verbindung zum rechten Rand ist in orange gehalten und auch das Quadrat hat eine Farbe gewechselt.

Weiterhin ist das Quadrat jetzt komplett ausgefüllt und das ist ein Zeichen dafür, dass unsere gesamte Case-Struktur nun komplett verdrahtet ist und das VI ablauffähig wäre.

Damit ist unser Block **1 1 b** das Blockdiagramms fertig.

Speichern Sie das VI ab.

Jetzt fehlt eigentlich nur noch eine kleine Abschlussberechnung:

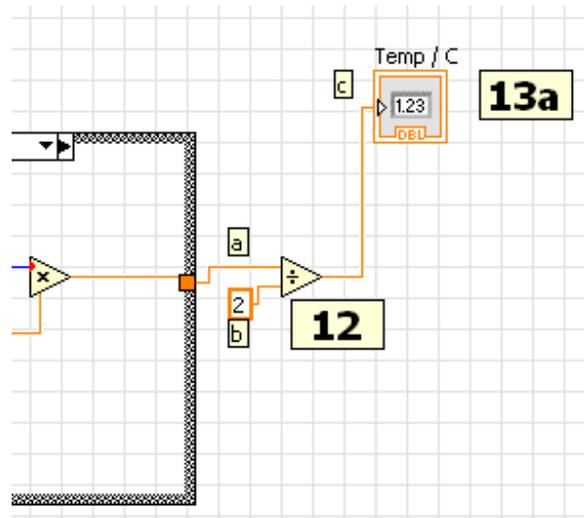
Der Temperaturmesswert hat eine Auflösung von  $0,5^{\circ}\text{C}$ , d.h. ein Bit hat die Wertigkeit von  $0,5^{\circ}\text{C}$ .

Um nun den endgültigen Temperaturwert zu bekommen, muss der Messwert also noch durch  $2$  geteilt werden.

Hierzu verwenden wir die Divisionsfunktion

#### BD\Programmierung\Numerisch\Dividieren

und platzieren diese rechts außerhalb des rechten Randes des Case-Rahmens, **Abb.5.1.24:**



**Abb.5.1.24: Die letzte Aufbereitung und die Anzeige des Messwertes (Block ① ② und ① ③ a)**

Den oberen Eingang der Divisionsfunktion verbinden Sie mit dem orangen Quadrat am rechten Rand des Case-Rahmens (Verbindung (a)).

An den unteren Anschluss der Divisionsfunktion schließen Sie eine Konstante mit dem Wert '2' an (Verbindung (b)).

Damit ist der letzte Berechnungsblock ① ② fertig.

Was jetzt eigentlich nur noch fehlt, ist eine geeignete Anzeige des Messwertes, denn alle notwendigen Berechnungen und Anpassungen sind abgeschlossen.

Schließen Sie daher am Ausgang der Divisionsfunktion ein Anzeigeelement an und nennen Sie dieses 'Temp / C'.

Als „**Sahnehäubchen**“ wollen wir auf dem Frontpanel zum Schluss noch das **aktuelle Tagesdatum** und die **aktuelle Uhrzeit** darstellen.

Hierzu bietet LabVIEW auch schon eine fertige Funktion an, die diese Daten aus „dem Inneren“ des PC/LapTops von der Systemuhr ermittelt:

### Datum-/Zeit-String lesen

#### BD\Programmierung\Timing\In Datum/Zeit

Die **Abb.5.1.25** zeigt die Details:

### Datum- /Zeit-String lesen (Funktion)

**Übergeordnete Palette:** [Timing-VIs und -Funktionen](#)

**Erfordert:** Base Package

Wandelt einen [Zeitstempel](#) oder einen numerischen Wert in eine D als Anzahl der Sekunden interpretiert, die seit Freitag, dem 01. Jar

[Beispiel](#)

+ Zum Blockdiagramm hinzufügen    🔍 Auf Palette suchen

**Datumsformat** wählt das Anzeigeformat von **Datum** au [Datum/Zeit formatieren](#) verändert.

0	<b>Short</b> —1/21/94
1	<b>Long</b> —Freitag, 21. Januar 1994
2	<b>Abbreviated</b> —Fr., 21. Jan 1994

#### Abb.5.1.25: „Wem die Stunde schlägt ...“

Die externe Verdrahtung dieser Funktion ist natürlich wieder denkbar einfach:

#### **Datumsformat(0):**

Hier legen Sie fest, in welcher Form das Datum später auf dem Frontpanel dargestellt werden soll (s. Tab. in Abb.5.1.26 untere Hälfte).

Wir schließen hier nichts an und wählen somit das 'Short'-Format (der Standardwert bei nicht belegtem Eingang ist ja der Wert '0').

#### **Aber Achtung:**

Hier ist noch ein kleiner Fehler in der LabVIEW-Hilfe:

Das in der Tabelle angegebene 'Short'-Format '1/21/94' ist das amerikanische Standardformat mit „Monat zuerst, '/' als Trennungszeichen und das Jahr zweistellig“.

In unserer „europäischen“ 2011er-LabVIEW-Version sieht 'Short' auf dem Frontpanel so aus, wie wir es gewohnt sind: '21.1.1994'.

#### **Zeitstempel:**

wird offen gelassen.

#### **Angabe von Sekunden? (F):**

Hier legt man fest, ob bei der Zeitanzeige auch die Sekunden mit angezeigt werden.

Wir legen aber großen Wert auf eine Sekunden-genaue Anzeige und schließen daher hier eine Konstante mit dem Wert 'T' an.

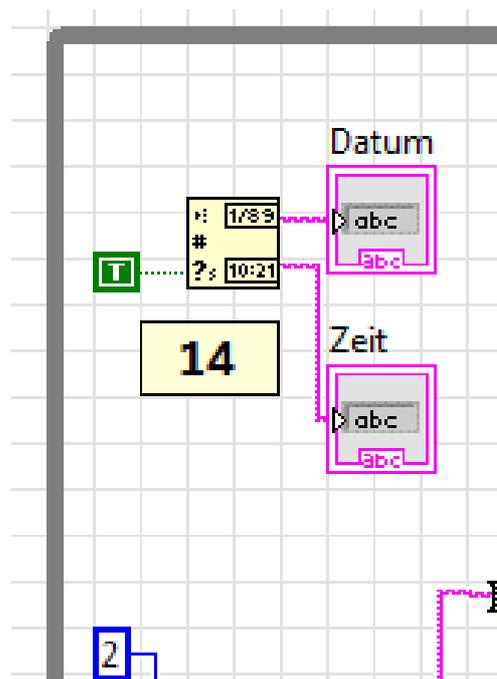
**Datum:**

An diesem Ausgang wird ganz einfach ein Anzeigeelement angeschlossen und dort wird später auf dem Frontpanel das Datum angezeigt.

**Zeit:**

An diesem Ausgang wird ganz einfach ein Anzeigeelement angeschlossen und dort wird später auf dem Frontpanel die aktuelle Uhrzeit angezeigt.

Und fertig ist unser Block **1 4**, Abb.5.1.26:



**Abb.5.1.26: „Die Zeit läuft ...“ (Block **1 4**)**

Speichern Sie das VI ab.

Kommen wir nun zum letzten Block unseres Blockdiagramms:

Wenn das VI beendet werden soll, klicken Sie auf dem Frontpanel einfach auf die **‘Stopp-Taste’** der While-Schleife: die While-Schleife wird beendet und das VI stoppt.

**Aber:**

Die LabVIEW-VISA-Funktionen halten hier noch eine „**kleine Gemeinschaft**“ für den Anwender bereit: sie geben nämlich nicht so einfach die serielle COM-Schnittstelle wieder frei !

Mit anderen Worten: obwohl das VI nicht mehr abläuft, hält LabVIEW die COM-Schnittstelle weiterhin besetzt und kein anderes Windows-Programm kann darauf zugreifen !

Die Lösung für dieses Problem ist jedoch ganz einfach:

Wir müssen LabVIEW beim Beenden des VIs anweisen, die Schnittstelle wieder ordnungsgemäß zu schließen und frei zu geben.  
Das machen wir mit der Funktion:

## VISA: Schließen

(BD\Instrumenten-I/O\Seruell\VISA: Schließen)

Abb.5.1.27 zeigt die Details:



**Abb.5.1.27: Die Schließen-Funktion: 'VISA: Schließen'**

Platzieren Sie daher diese Funktion rechts außerhalb der While-Schleife, **Abb.5.1.28:**





**Abb.5.1.29: Fehler oder keine Fehler ?**

Dazu ist die linke obere Ecke des Blockdiagramm-Fensters wesentlich:

Wenn dort der „dicke Startpfeil“ nicht zerbrochen dargestellt ist, so haben Sie aktuell keine Fehler gemacht und das LabVIEW-VI wäre startbereit.

Ist der „dicke Startpfeil“ jedoch zerbrochen dargestellt, so haben Sie mindestens einen kritischen Fehler gemacht und das LabVIEW-VI kann nicht gestartet werden.

In diesem Fall klicken Sie einfach auf den zerbrochenen Doppelpfeil: Sie erhalten jetzt eine ausführliche **Fehlerliste**, die Sie abarbeiten müssen.

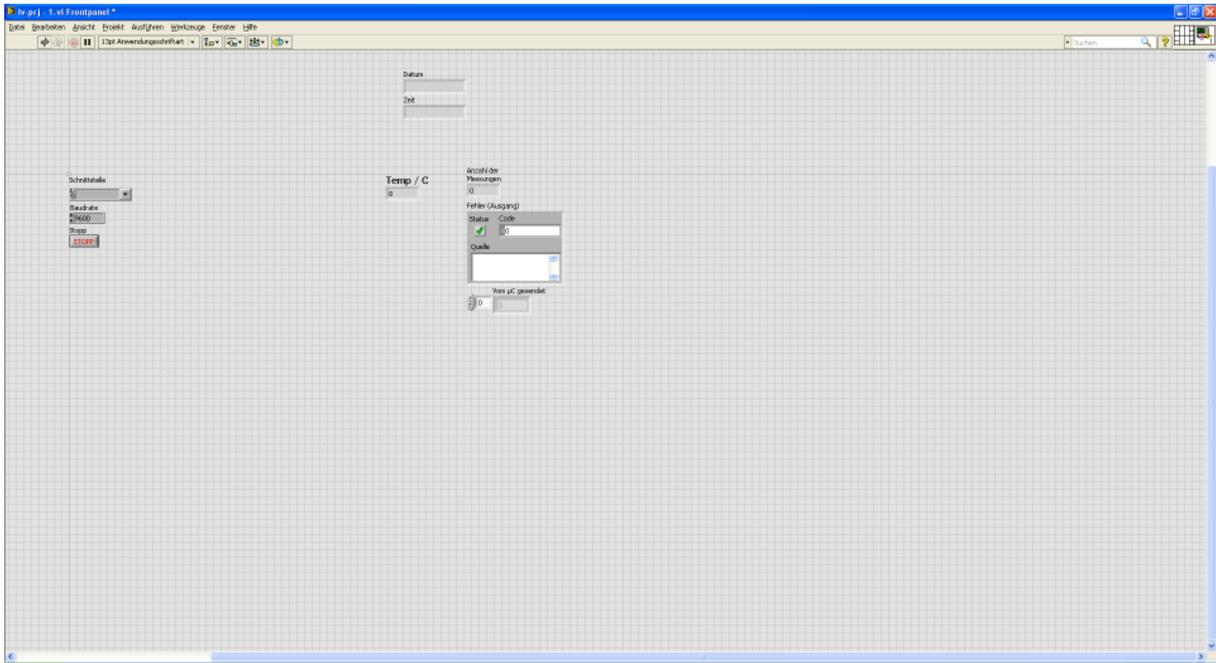
Und nun kommt **der zweite Höhepunkt** der LabVIEW-VI-Entwicklung .....

### *Die wunderschöne Gestaltung des Frontpanels*

## 5.2 Das Frontpanel

Nun können Sie Ihrem Gestaltungsdrang den freien Lauf lassen und ein „hinreißend geschmackvolles“ Frontpanel für den (End)Anwender entwerfen.

Schalten Sie dazu mit 'Strg+E' um auf das Frontpanel, **Abb.5.2.1**:



**Abb.5.2.1: Das von LabVIEW bereits „im Hintergrund“ erzeugte Frontpanel im Rohzustand**

Parallel zu Ihren Eingaben der Anzeige- und Bedienelemente im Blockdiagramm hat LabVIEW im Hintergrund bereits ein Frontpanel im Rohzustand erzeugt.

Allerdings: dieses Design lässt doch sehr zu wünschen übrig: LabVIEW hat recht willkürlich und ungeordnet alle Elemente auf dem Frontpanel verteilt.

Sie müssen jetzt also noch ein bisschen aufräumen, d.h. die Frontpanel-Elemente „schön und sinnvoll“ anordnen, größer und farbiger darstellen, etc.

Als erstes fügen wir allerdings noch zwei Anzeige-Elemente hinzu, denn wir wollen die gemessene Temperatur ja auf drei verschiedene Arten darstellen (s. Frontpanel-Darstellung zu Anfang des Projektes):

### **Hinzufügen eines Rundinstrumentes**

Sie finden dieses Anzeigeelement unter:

**FP\Modern\Numerisch\Rundinstrument**

Nennen Sie dieses Element um in 'Temperatur / C' und platzieren Sie es „irgendwo“ auf dem Frontpanel.

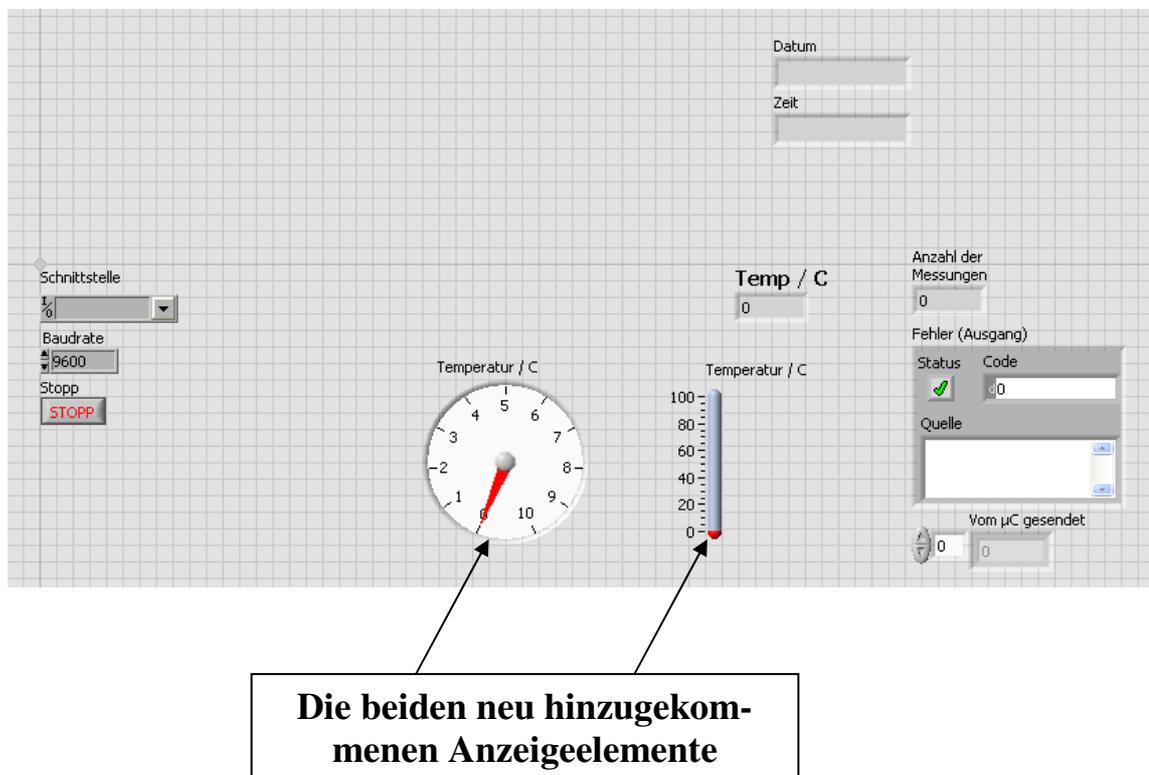
### Hinzufügen eines Thermometers

Sie finden dieses Anzeigeelement unter:

**FP\Modern\Numerisch\Thermometer**

Nennen Sie dieses Element um in 'Temperatur / C' und platzieren Sie es „irgendwo“ auf dem Frontpanel.

Das erweiterte Frontpanel sollte jetzt ungefähr so aussehen, **Abb.5.2.2**:



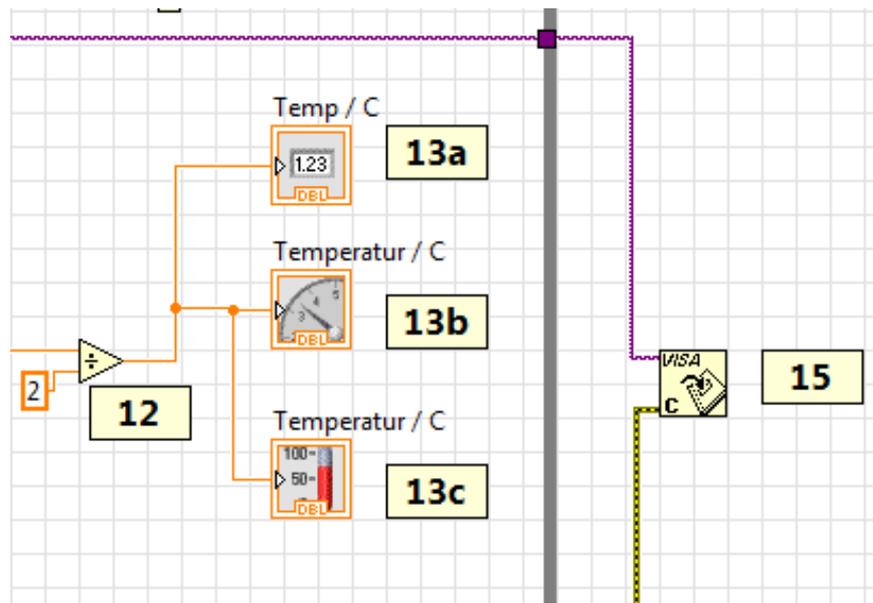
**Abb.5.2.2: Zwei neue Anzeigeelemente sind auf dem Frontpanel hinzugekommen**

Speichern Sie das VI ab und schalten Sie noch einmal kurz um auf das Blockdiagramm ('Strg+E').

„Irgendwo“ im Blockdiagramm finden Sie nun diese beiden neuen Anzeigeelemente „lose in der Gegend herum hängen“.

Platzieren Sie daher diese beiden Anzeigen in der Nähe von Block **1 3 a** und verbinden Sie diese ebenfalls mit dem Ausgang der Divisionsfunktion, Block **1 2**.

So entstehen die Böcke **1 3 b** und **1 3 c** im Blockdiagramm, **Abb.5.2.3**:



**Abb.5.2.3: Die neu hinzugekommen Anzeigeelemente, im Blockdiagramm korrekt verdrahtet, Blöcke 13a, 13b und 13c**

Alle drei Anzeigeelemente zeigen nun zwar immer denselben Temperaturwert an, aber dieses auf drei verschiedene Arten und Weisen: als reinen Zahlenwert, als Zeigerauslag auf einem Rundinstrument und in der „klassischen“ Thermometer-Darstellung.

Schalten Sie nun um auf das Frontpanel (Abb.5.2.2) und beginnen Sie mit der endgültigen Gestaltung des Benutzer-Interfaces.

Die **Bearbeitungshilfsmittel**, die Sie jetzt dazu benötigen sind:

- das Markieren von Elementen
- das Verschieben von Elementen
- das Vergrößern/Verkleinern von Elementen
- die Änderung von Schrift-Attributen: Größe, Farbe, Ausrichtung, ...
- das Ändern von Eigenschaften von Elementen.

***Nehmen Sie sich als Gestaltungsvorlage vielleicht den Frontpanel-Entwurf, den wir am Anfang des Projektes vorgestellt haben.***

(Und denken Sie daran: mit 'Strg+Z' können Sie alle Änderungen wieder rückgängig machen).

## 1. Schritt

Verschieben Sie die Elemente 'Schnittstelle', 'Baudrate' und 'Vom  $\mu\text{C}$  gesendet' in die linke obere Ecke.

Erweitern Sie das Anzeigeelement 'Vom  $\mu C$  gesendet' nach rechts hin um eine weitere Anzeigestelle, da der Mikrocontroller ja zwei Bytes sendet und diese hier angezeigt werden sollen.

Markieren Sie nun alle drei Elemente: mit 'Strg+0' können Sie nun ganz einfach alle Textattribute ändern. Stellen Sie z.B. ein:

- Größe: 18
- Ausrichten: zentriert
- Fett
- Schwarz

## 2. Schritt

Platzieren Sie die Elemente 'Stopp-Taster', 'Zeit' und 'Datum' in die rechte untere Ecke. Ändern Sie die Attribute des 'Stopp-Tasters' auf:

- Größe: 24
- Fett
- Rot

Löschen Sie bei diesem Taster den zusätzlichen Beschreibungstext über dem Taster.

Ändern Sie die Attribute der Felder 'Datum' und 'Zeit' auf:

- Größe: 18
- Ausrichten: zentriert
- Fett
- Schwarz

Platzieren Sie diese beiden Elemente „schön“ oberhalb des Stopp-Tasters.

## 3. Schritt

Platzieren Sie das Anzeigeelement 'Fehler (Ausgang)' in der rechten oberen Ecke und lassen Sie die Attribute dafür unverändert.

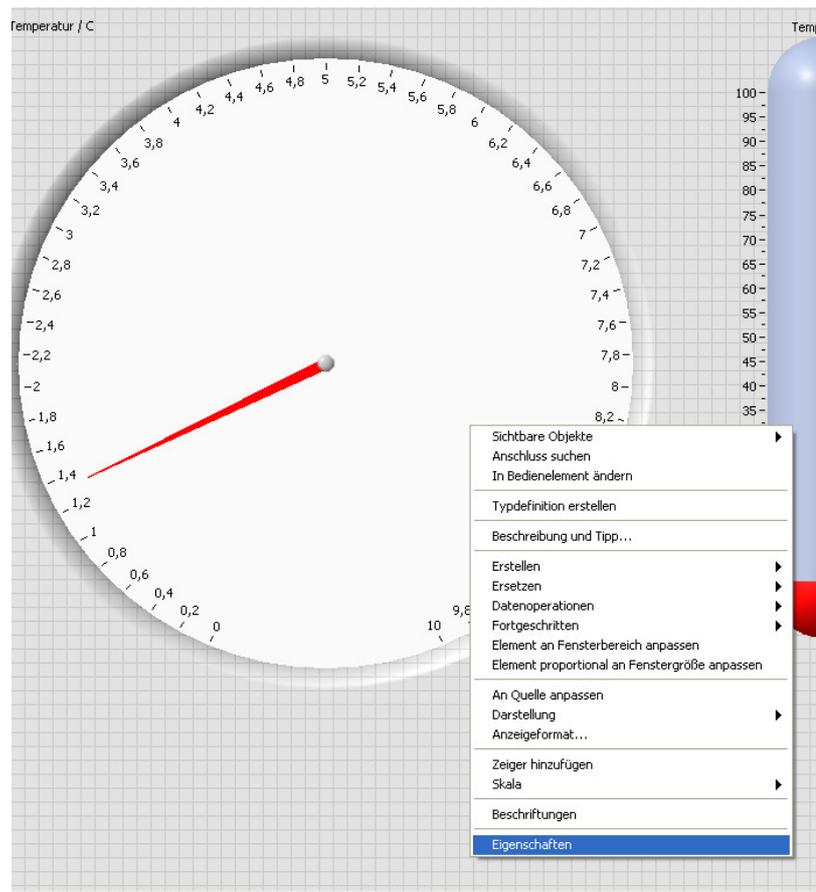
## 4. Schritt

Platzieren Sie die beiden Elemente 'Rundinstrument' und 'Thermometer' mittig auf dem Frontpanel und ziehen Sie diese auf maximale Größe auf.

Es müssen nun noch die Anzeigebereiche und die Beschriftungstexte angepasst werden.

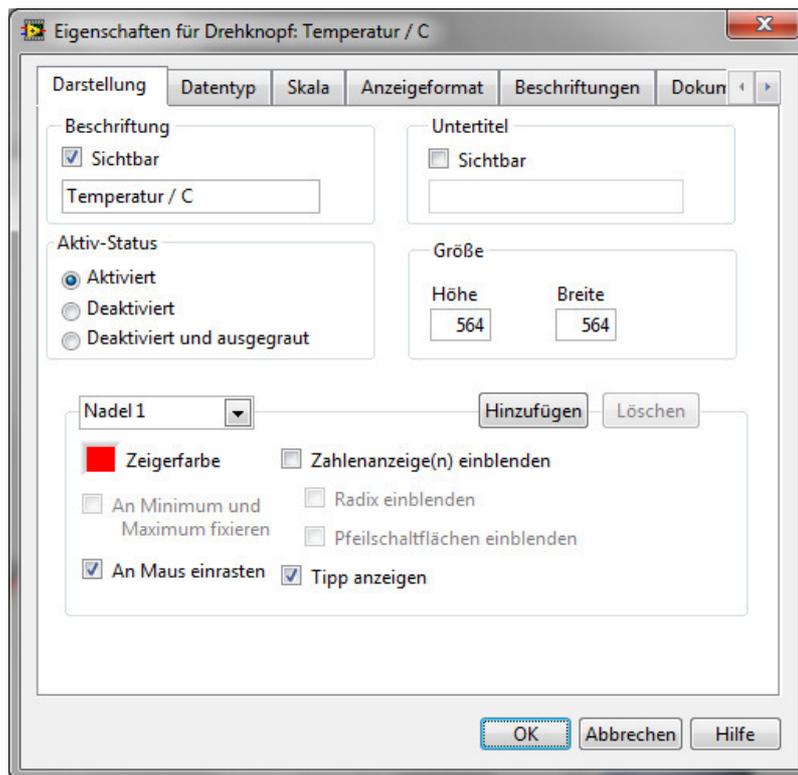
### Rundinstrument

Wählen Sie aus dem Kontextmenü zu dieser Anzeige den Punkt 'Eigenschaften', Abb.5.2.4:



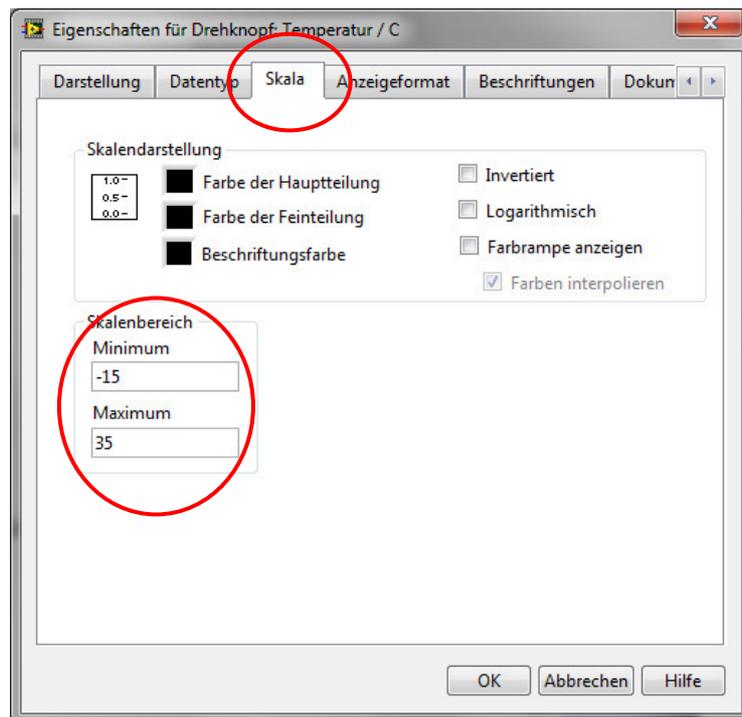
**Abb.5.2.4: Die Auswahl der 'Eigenschaften' für das Rundinstrument**

Es erscheint ein neues Fenster mit den Registerkarten, über die nun die Eigenschaften ganz konkret eingestellt und festgelegt werden können, **Abb.5.2.5**:



**Abb.5.2.5: Die Eigenschafts-Registerkarten**

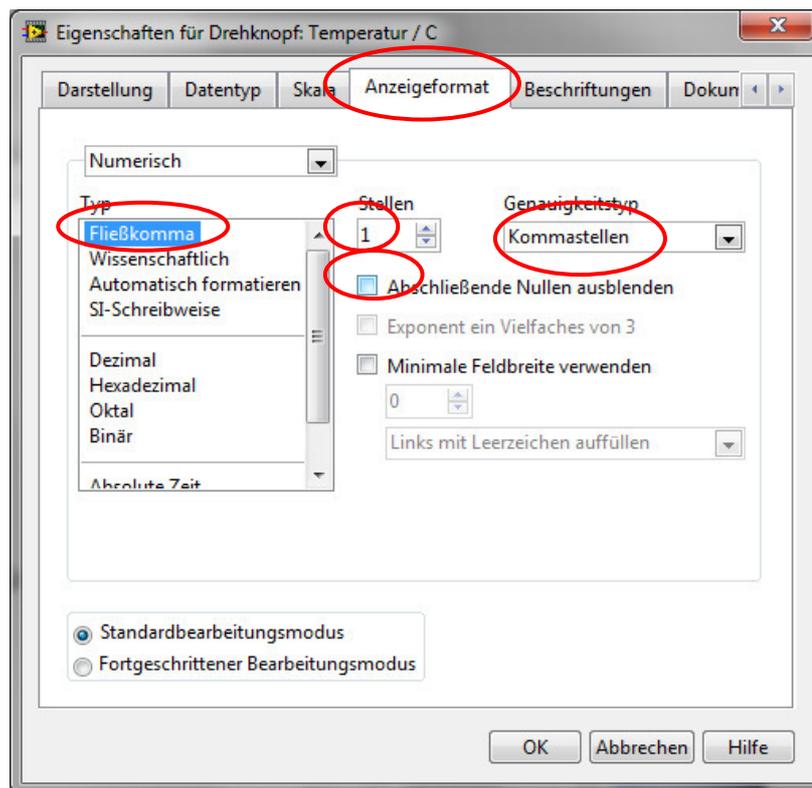
Wählen Sie als Erstes die Registerkarte 'Skala' und machen Sie die folgenden Festlegungen, **Abb.5.2.6:**



**Abb.5.2.6: Die Festlegung des minimalen und maximalen Anzeigewertes**

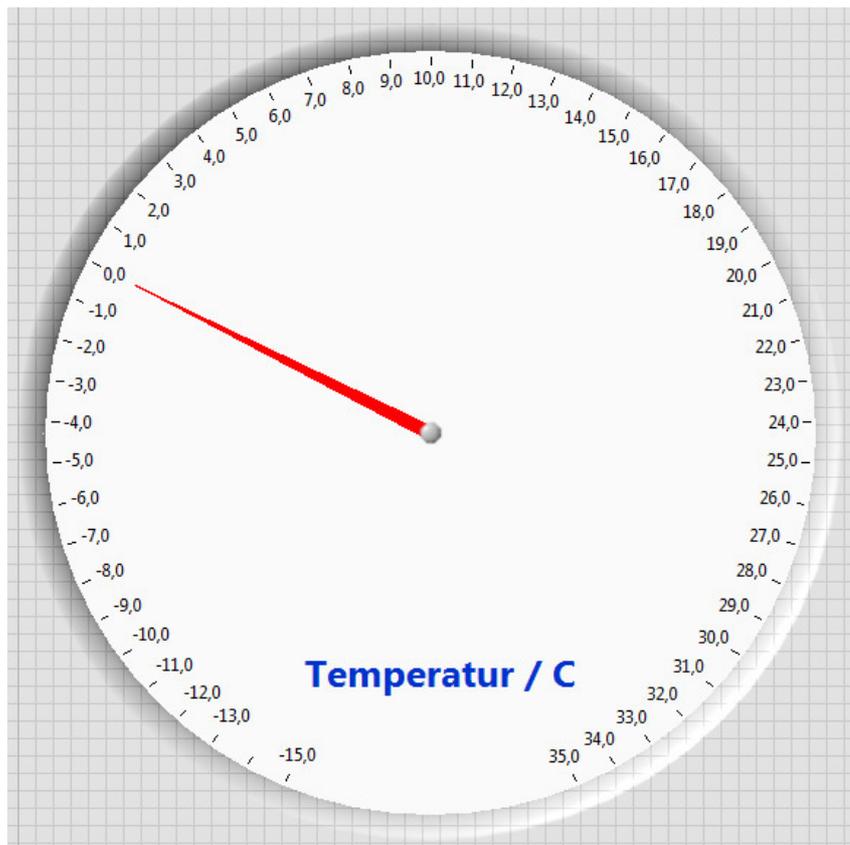
Durch diese Festlegungen werden Temperaturen im Bereich von  $-15^{\circ}\text{C}$  bis  $+35^{\circ}\text{C}$  angezeigt.

Wählen Sie nun die Registerkarte 'Anzeigeformat' und führen Sie auch hier die entsprechenden Einstellungen durch, **Abb.5.2.7**:



**Abb.5.2.7: Die Festlegung des Anzeigeformats**

Klicken Sie nun auf 'OK' und das Rundinstrument sollte nun so aussehen, **Abb.5.2.8:**

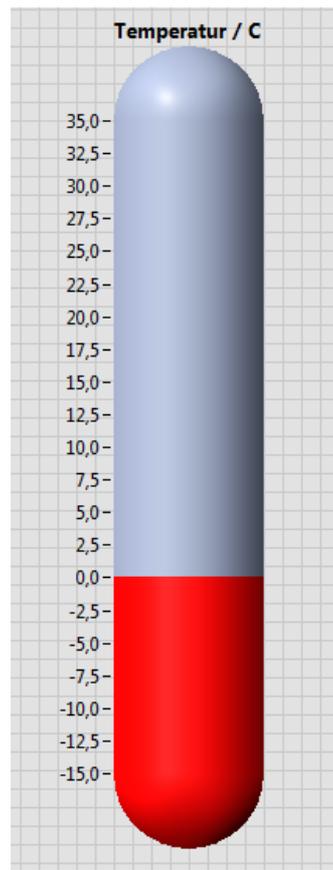


**Abb.5.2.8: Das fertig konfigurierte Rundinstrument**

Den Beschriftungstext 'Temperatur / C' haben wir in das Instrument verschoben und wie folgt formatiert:

- Größe: 36
- Fett
- Blau

Konfigurieren Sie jetzt auf die gleiche Art und Weise das Thermometer, **Abb.5.2.9:**



**Abb.5.2.9: Das fertig konfigurierte Thermometer**

## 5. Schritt

Platzieren das Anzeigeelement 'Anzahl der Messungen' zwischen dem Rundinstrument und dem Thermometer und formatieren Sie es nach Ihren Vorstellungen.

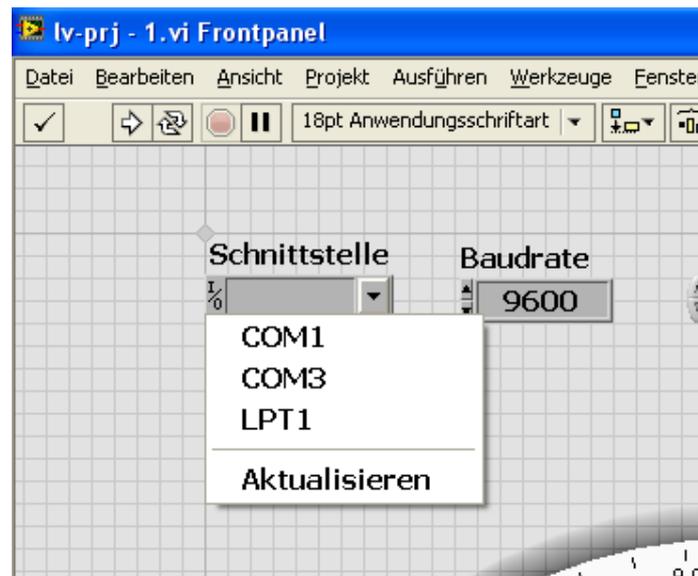
Platzieren das numerische Anzeigeelement 'Temp / C' rechts neben dem Thermometer und formatieren Sie auch diese Anzeige nach Ihren Vorstellungen.

Damit sollte Ihr Frontpanel nun anschaulich und übersichtlich gestaltet sein.

Ein allerletzter Punkt fehlt jetzt noch: die Auswahl der serielle Schnittstelle.

Fahren Sie dazu mit dem Cursor auf den kleinen Pfeil rechts am Bedienelement 'Schnittstelle'.

Wenn der Cursor die „Hand-mit-ausgestecktem-Finger“-Form hat, klicken Sie mit der linken Maustaste, **Abb.5.2.10:**



**Abb.5.2.10: Die Auswahl der gewünschten Schnittstelle**

Es erscheint ein kleines Fenster, in dem die zur Zeit am PC/LapTop vorhandenen Schnittstellen angezeigt werden (wir haben hier einen ganz modernen PC der bereits wieder serielle COM-Schnittstellen besitzt).

Jetzt kann die gewünschte serielle Schnittstelle einfach ausgewählt werden, hier z.B. COM3.

**Problem jedoch:**

Wenn Sie einen (nicht so) modernen PC/LapTop, ohne COM-Schnittstellen, besitzen, wird gar keine serielle Schnittstelle angezeigt, da LabVIEW die reinen USB-Schnittstellen nicht als serielle Schnittstellen erkennt bzw. akzeptiert.

Schließen Sie in diesem Falle einfach einen USB/Seriell-Adapter an eine USB-Buchse an und klicken Sie, nach einer kurzen Wartezeit, auf das Feld 'Aktualisieren'.

LabVIEW erkennt dann diesen Adapter als „echte“ serielle Schnittstelle und kann daher damit wie gewohnt arbeiten.

Wählen Sie nun diese Schnittstelle aus und damit ist dann Ihr LabVIEW-VI ablauffähig.

**Hinweis:**

Sollte der USB/Seriell-Adapter von LabVIEW nicht als COM-Schnittstelle erkannt werden, so speichern Sie Ihr VI, beenden LabVIEW und starten LabVIEW (über den MAX) erneut.

Dann sollte der Adapter auf jeden Fall richtig erkannt werden.

Und nun wird's spannend ...

### 5.3 Test und Ergebnis

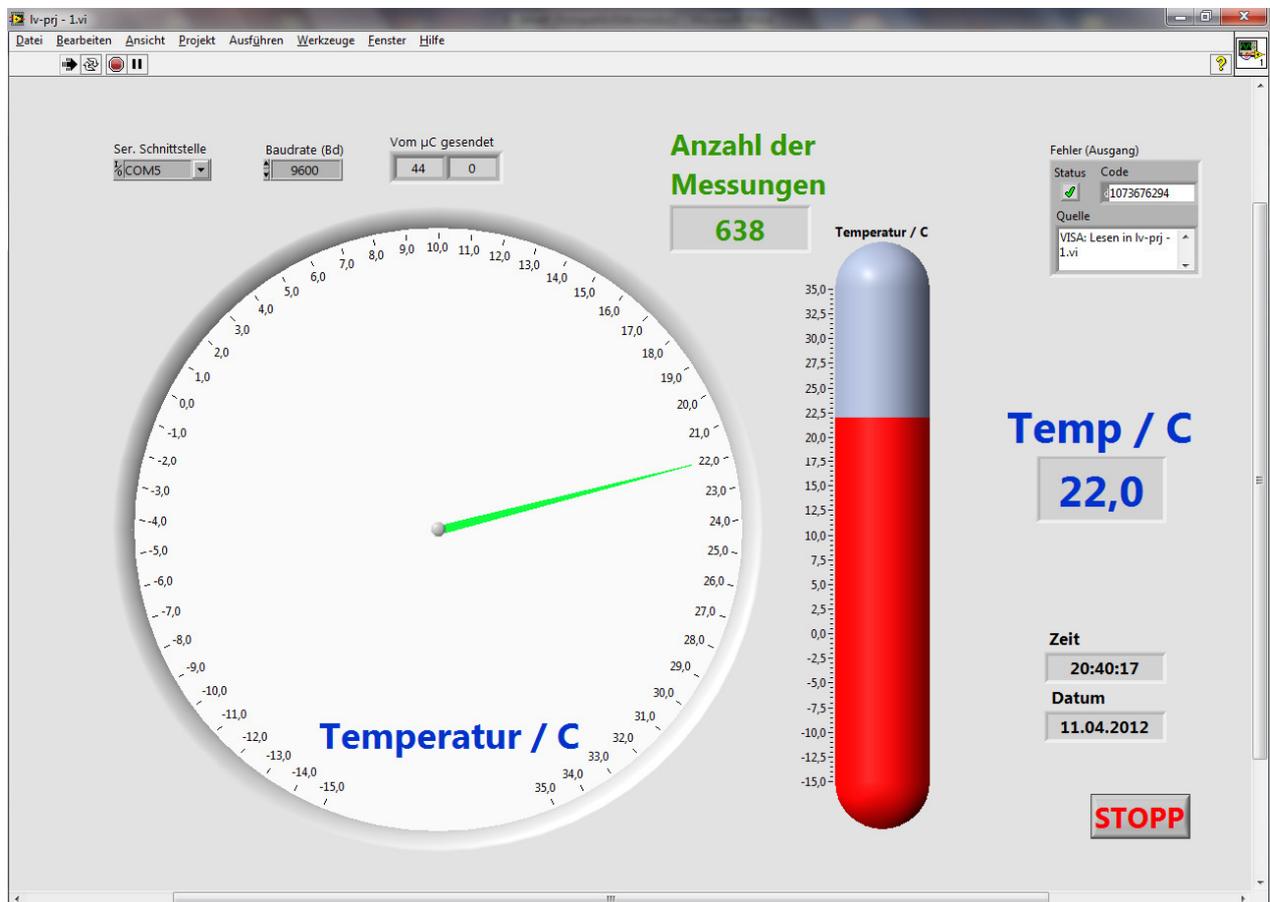
Schalten Sie um auf das Frontpanel und starten Sie das VI durch Klicken auf das 'Ausführen'-Ikon (s. Kap. 4.4).

(Auf 'Widerholt ausführen' brauchen Sie nicht zu klicken, da wir in unserem VI ja bereits eine (Endlos)-While-Schleife eingebaut haben).

#### **Hinweis:**

Das Mikrocontroller-Programm muss natürlich schon laufen und seine Messwerte an den PC/LapTop senden.

Die **Abb.5.4.1** zeigt das nun erzielte Ergebnis:



**Abb.5.4.1: Das VI in Aktion**

Im Sekundentakt sendet das Mikrocontroller-System die beiden Temperaturmesswerte, die zunächst als Rohwerte angezeigt werden.

Danach erfolgt die Umrechnung und die Darstellung auf den drei verschiedenen Anzeigeeinheiten.

Die fortlaufende Angabe der aktuellen Uhrzeit und des aktuellen Datums ergänzt die Frontpanel-Darstellung.

Sie stoppen das VI durch Klicken auf den **‘Stopp-Taster’**.

Lässt sich das VI aufgrund eines Fehler so nicht anhalten, so beenden Sie dessen Ausführung durch Klicken auf das **‘Ausführung abbrechen’**-Ikon (s. Kap. 4.4)

Läuft das VIs jedoch nicht so wie gewünscht, dann beginnt die Fehlersuche ....

Gehen Sie noch einmal Schritt für Schritt die Entwicklung des Blockdiagramms durch und prüfen Sie vor allen Dingen, ob der USB/Seriell-Adapter einwandfrei arbeitet und von LabVIEW erkannt wird.

Sie können z.B. ein ganz normales Terminal-Programm (z.B. HyperTerm) verwenden, um zu überprüfen, ob die beiden Bytes vom Mikrocontroller richtig empfangen werden und vor allen Dingen, auch den richtigen Inhalt haben.

## **6. Und wie geht's weiter ?**

Dieses kleine Projekt sollte Ihnen „einen ersten Geschmack auf den Einsatz von LabVIEW“ machen.

Wenn uns das gelungen ist, und Sie noch viel mehr über LabVIEW erfahren wollen und vor allen Dingen LabVIEW in der täglichen Praxis einsetzen möchten, so gibt es jetzt verschiedene Möglichkeiten, sein Wissen zu erweitern:

Anfang Juni 2012 erscheint in Elektor-Verlag der erste Band einer von uns herausgegebenen **Lehrbuchreihe** zum Thema **“LabVIEW für den Praktiker“**.

Hiermit wird ein fundierter Einstieg in LabVIEW mit vielen praktischen Beispielen, Projekten und Übungen (mit Musterlösungen) möglich.

Anfangen von den Grundlagen über den Betrieb von seriellen Schnittstellen bis hin zur graphischen Darstellung und Auswertung von Daten reicht zunächst das Spektrum in den ersten beiden Bände.

Weiterhin veranstalten wir in Zusammenarbeit mit dem Elektor-Verlag in verschiedenen Orten Deutschlands mehrtägige **Seminare und Workshops** zum Themenbereich **„LabVIEW meets  $\mu$ C“**.

Nähere Informationen dazu finden Sie im Kapitel ‘Literatur und Bezugsquellen’.

## **7. Anhänge**

## 7.1 Liste der Shortcuts

**Shortcuts** sind Tastenkombinationen, mit denen man bestimmte LabVIEW-Aktionen einfach und schnell ausführen kann.

Die zunächst wichtigsten Shortcuts sind:

‘**Strg+E**’: Umschalten zwischen Frontpanel und Blockdiagramm.

‘**Strg+Z**’: Rückgängig machen von zuvor durchgeführten Änderungen.

‘**Strg+H**’: Aufruf der Kontexthilfe.

‘**Strg+B**’: Löschen von überflüssigen Leitungen.

‘**Strg+0**’ (Null): Festlegungen zur jeweiligen Schriftart: Größe, Farbe, etc.

‘**Strg+S**’: Abspeichern des VIs.

‘**Strg+U**’: Aufräumfunktion für das Blockdiagramm.  
Identisch mit dem ‘Besen-Ikon’ in der Ikon-Leiste.

‘**Strg+T**’: Frontpanel und Blockdiagramm werden in Fenstern gleichzeitig nebeneinander dargestellt.

‘**Entf-Taste**’: auf der PC-Tastatur: Löschen von zuvor markierten Elementen.

## **8. Literatur, Seminare und Bezugsquellen**

### **Weiterführende Literatur:**

Wolfgang Georgi, Ergun Metin  
**„Einführung in LabVIEW“**  
5. überarbeitete und erweiterte Auflage, Januar 2012  
Carl Hanser Verlag GmbH & CO. KG  
ISBN-13: 978-3446423862

### **Das große deutschsprachige Hilfeforum zu LabVIEW:**

<http://www.labviewforum.de/index.php>

### **Mehrtägige Seminare und Workshops für LabVIEW-Anfänger, Neueinsteiger, Schüler, Auszubildende und Lehrer/Ausbilder beim Elektor-Verlag:**

[www.elektor.de](http://www.elektor.de)

### **LabVIEW-Projekte, Mikrocontroller-Systeme für Lehre und Ausbildung , u.v.a.m. von der Firma PalmTec:**

[www.palmtec.de](http://www.palmtec.de)

(u.a. das „**PT-First-Step-Board**“ aus diesem Projekt)

### **Weiterführende Informationen zu LabVIEW und Seminare für fortgeschrittene (Profi)Anwender bei National Instruments:**

[www.ni.com](http://www.ni.com)

## **9. Versions History**

23. Mai 2012:           Version 1.0 des Projektes veröffentlicht.