Prototyping VO+Ü
Vertiefung LabView

THEMEN

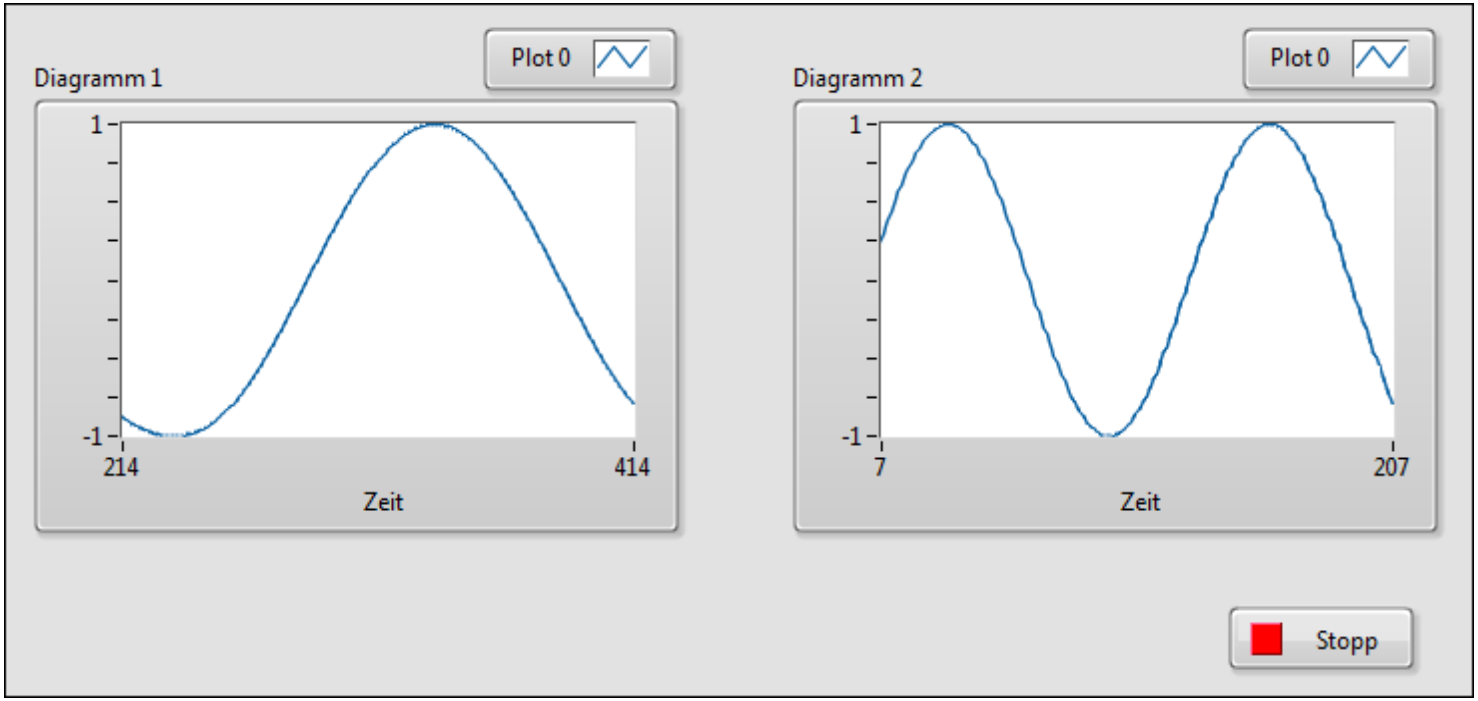
1. Kommunikation zwischen parallelen Schleifen, Variablen, Laufzeitprobleme, Eigenschafts- + Methodenknoten, FGV
2. Verzeichnispfade, FileIO: Binär-, Text-, TDMS-Dateien
3. Asynchrone Kommunikation, Queues, ereignisgesteuerte Programmierung, Gerätesteuerung, Gerätetreiber
4. myRio, Servoansteuerung, I2C-Bus, Sensorausweertung

VO1
Lösen von Datenflussproblemen mit Hilfe von Variablen

THEMEN

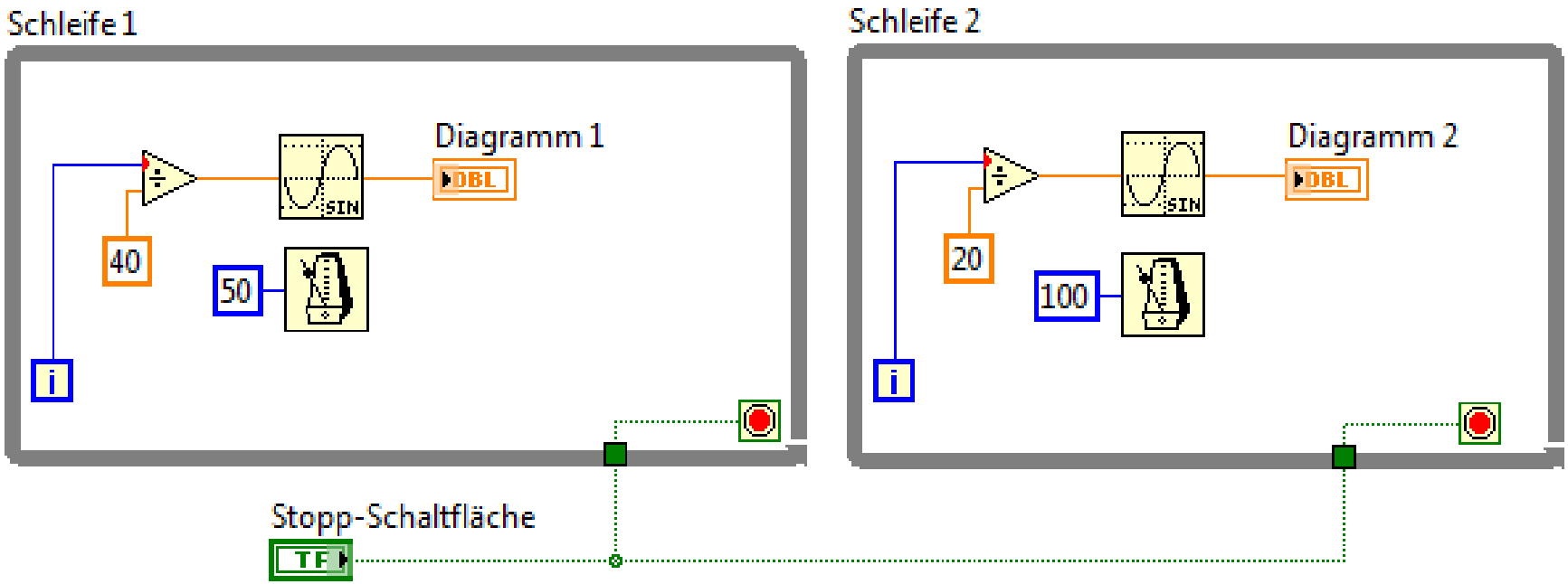
A. Kommunikation zwischen parallelen Schleifen
B. Schreiben in Bedienelemente und Auslesen aus Anzeigeelementen
C. Variablen
D. Laufzeitprobleme
E. Funktionale Globale Variablen (FGVs)

A. Kommunikation zwischen parallelen Schleifen



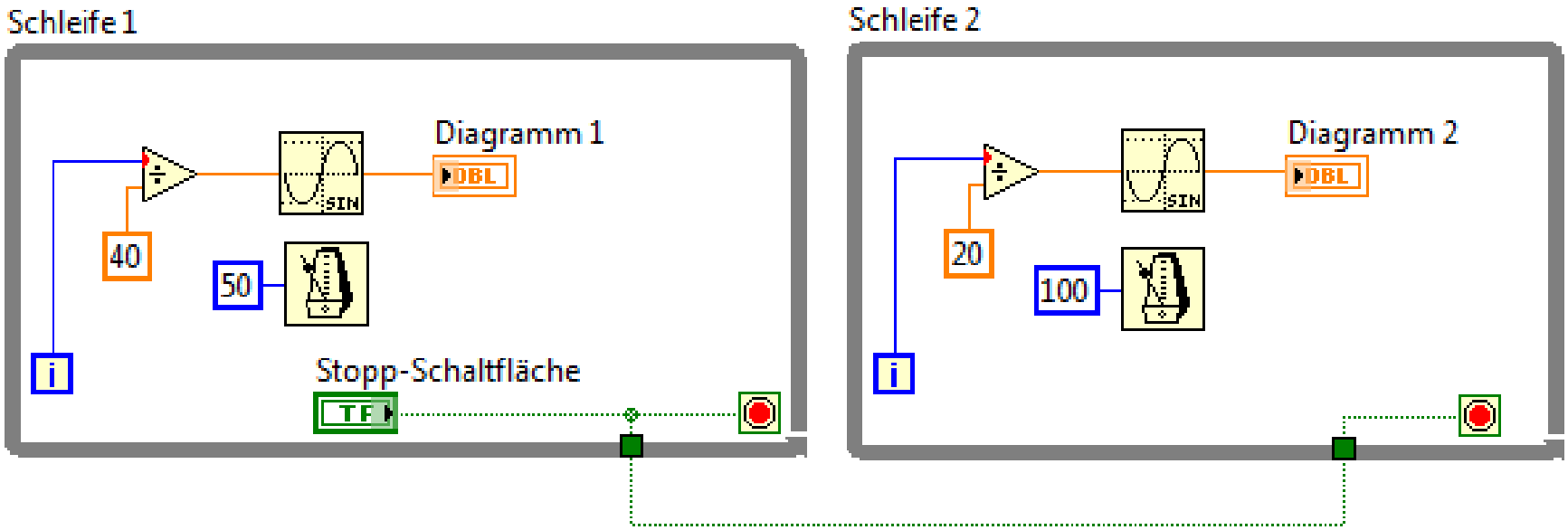
Beispiel: 2 Diagramme führen mehrere Aufgaben gleichzeitig aus

Wie werden die Schleifen in diesem Beispiel angehalten?



Der Datenaustausch zwischen parallelen Schleifen ist schwierig

Wie werden die Schleifen in diesem Beispiel angehalten?



Parallele Schleifen können Daten nicht über eine Direkt-verbindung austauschen

B. Schreiben in Bedienelemente und Auslesen von Anzeigeelementen

B-1. VI-Server-Architektur

VI-Server-Architektur

Eigenschaften und Methoden

VI-Klassenhierarchie

VI-Server-Architektur

Der VI-Server ermöglicht den programmatischen Zugriff auf LabVIEW

Mit dem VI-Server können Sie:

- Frontpanel-Objekte und VIs programmatisch steuern
- VIs dynamisch laden und aufrufen
- VIs über das Netzwerk ausführen
- Programmatisch auf die LabVIEW-Umgebung und den LabVIEW-Editor (für Skripte) zugreifen

Eigenschaften und Methoden

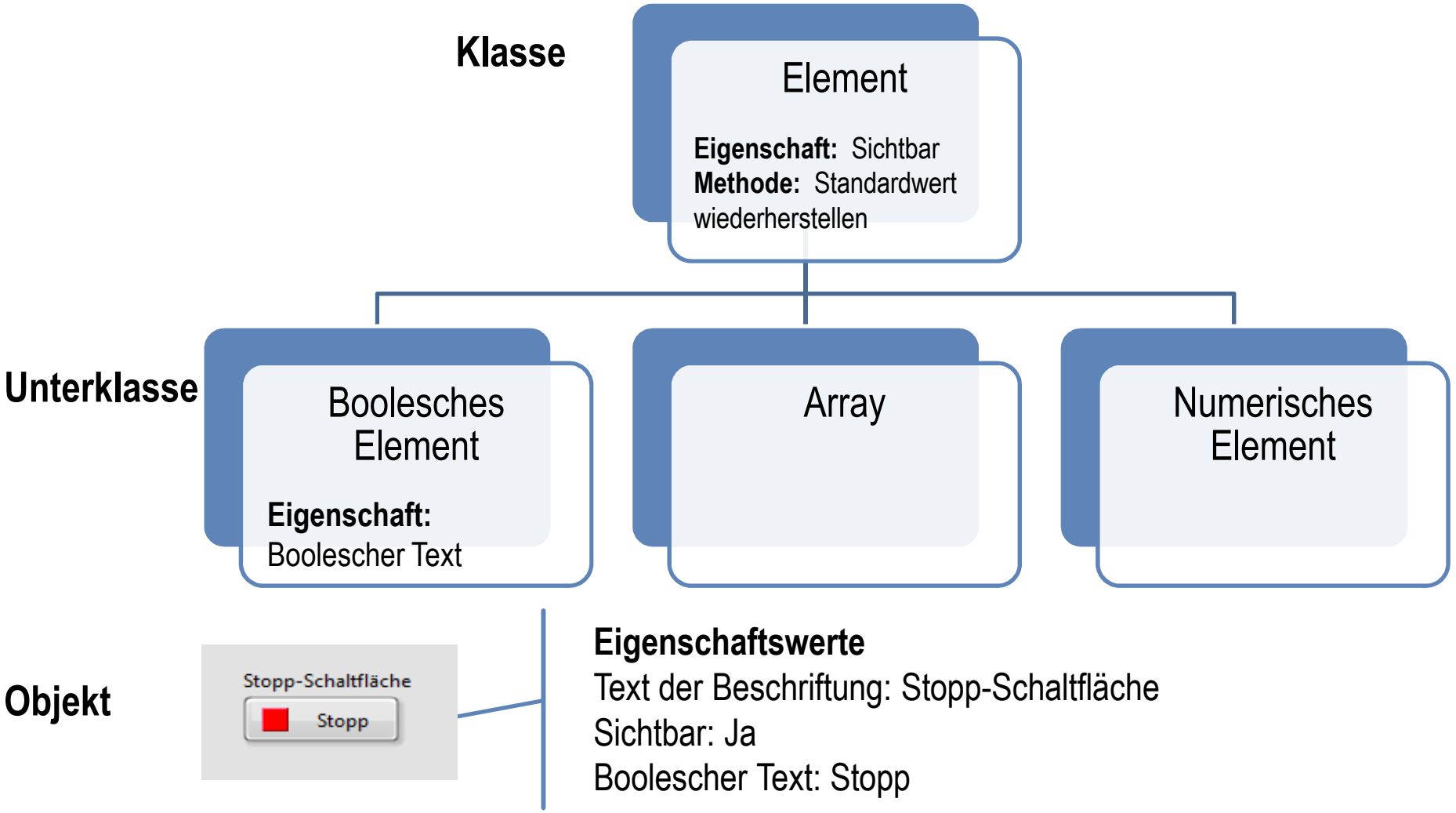
Eigenschaften: Attribute (1 Wert) des Objekts:
Lesen/Schreiben, nur Lesen, nur Schreiben

Zu den Eigenschaften zählen Farbe, Position, Größe, Sichtbarkeit, Beschriftung und Schriftart der Beschriftung

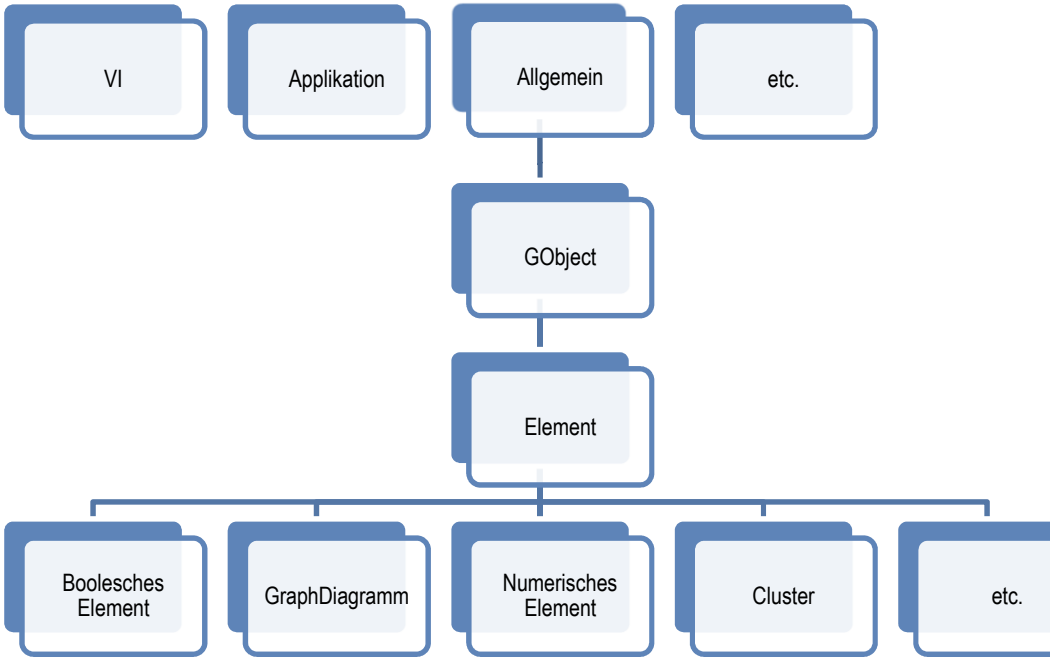
Methoden: Am Objekt ausgeführte Funktionen

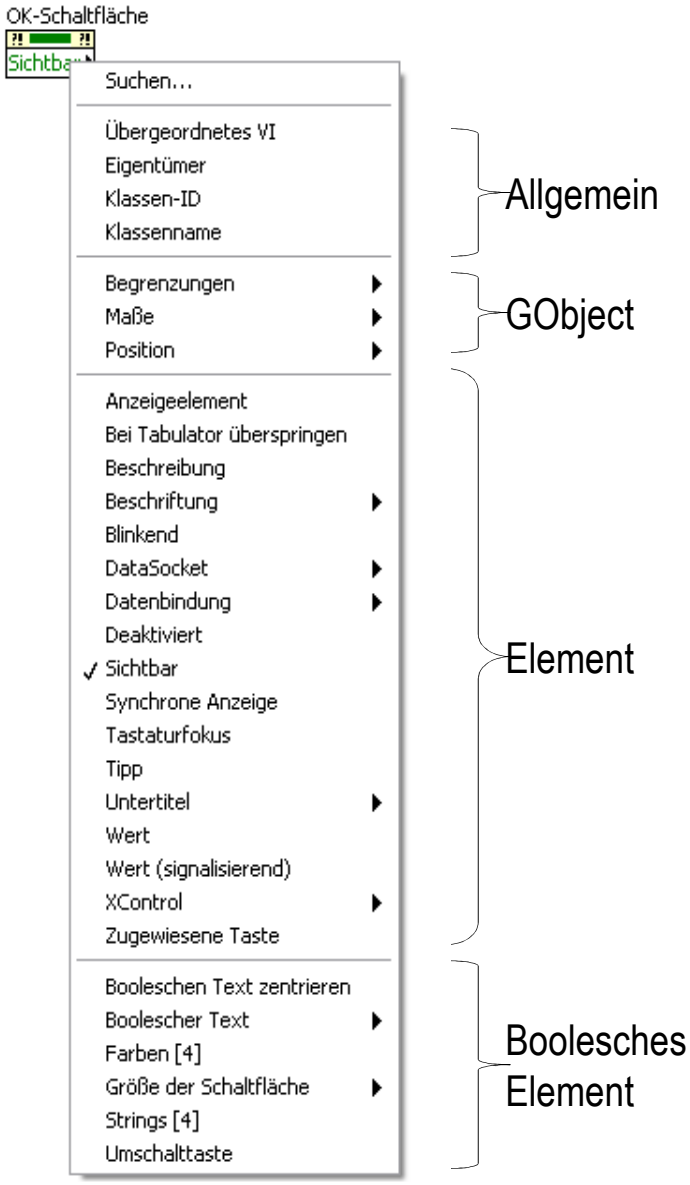
Zu den Methoden zählen z. B. das Wiederherstellen des Standardwerts und das Exportieren von Graphen-Abbildungen

Hierarchie der VI-Serverklassen



Hierarchie der VI-Serverklassen





B-2. Eigenschaftsknoten

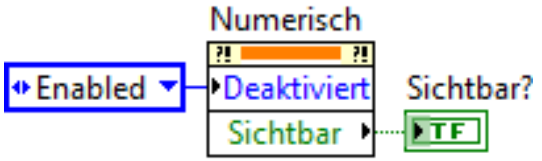
Definition

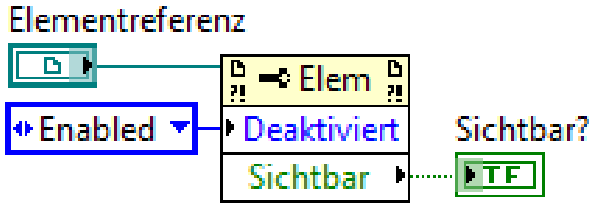
Erstellen von Eigenschaftsknoten

Ausführungsreihenfolge

Eigenschaftsknoten

- Greifen auf die Eigenschaften eines Objekts zu, z. B.:
  - Diagrammfarbe
  - Aktivierung/Deaktivierung von Elementen
  - Position von Elementen
- Änderungen programmatisch möglich
- Eigenschaften sind in der Kontexthilfe dokumentiert
- Arten von Eigenschaftsknoten:
  - Nicht sichtbar (implizit) verknüpft
  - Sichtbar (explizit) verknüpft



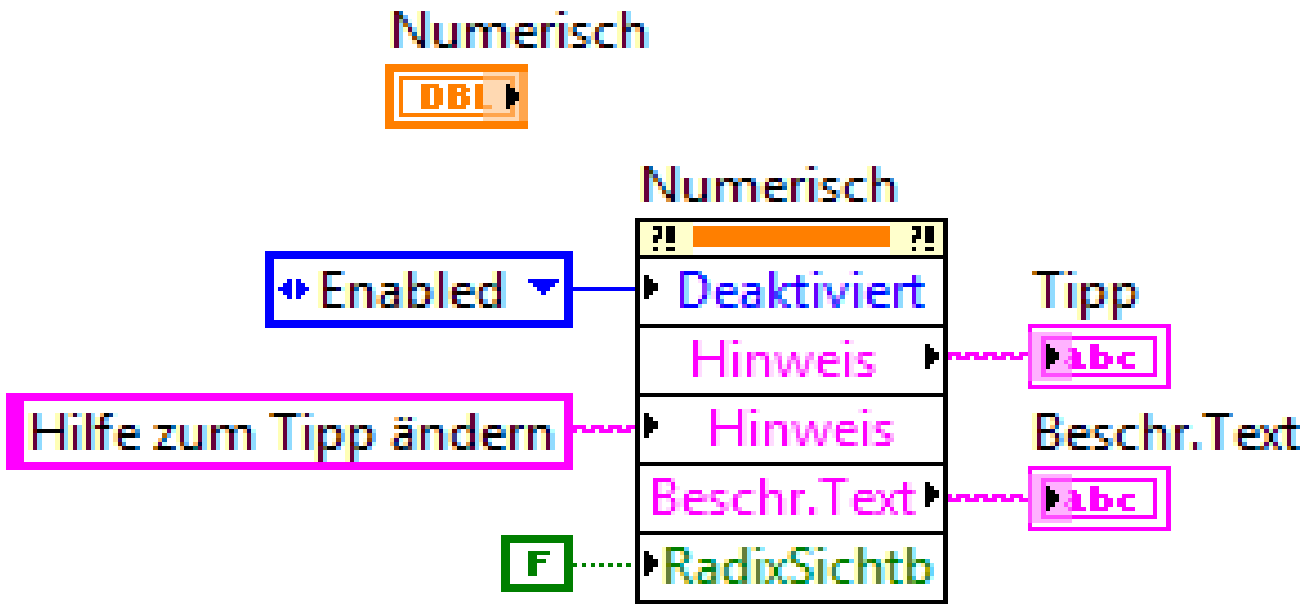


Erstellen von Eigenschaftsknoten

Erstellen eines Eigenschaftsknotens für ein Frontpanel-Objekt

Ausführungsreihenfolge

- Eigenschaftsknoten können mehrere Eigenschaften enthalten
- Eigenschaften werden von oben nach unten abgearbeitet



B-3. Methodenknoten

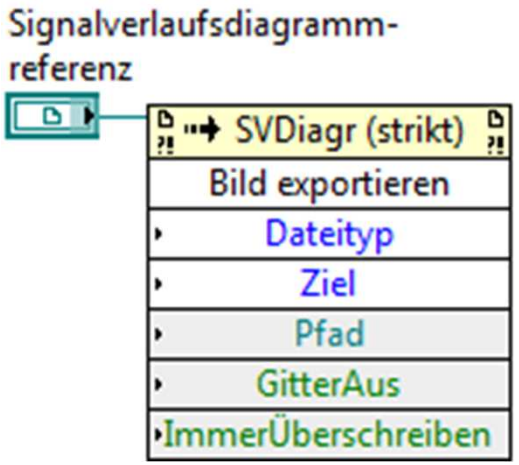
Definition

Methoden für Element

Methoden für VI

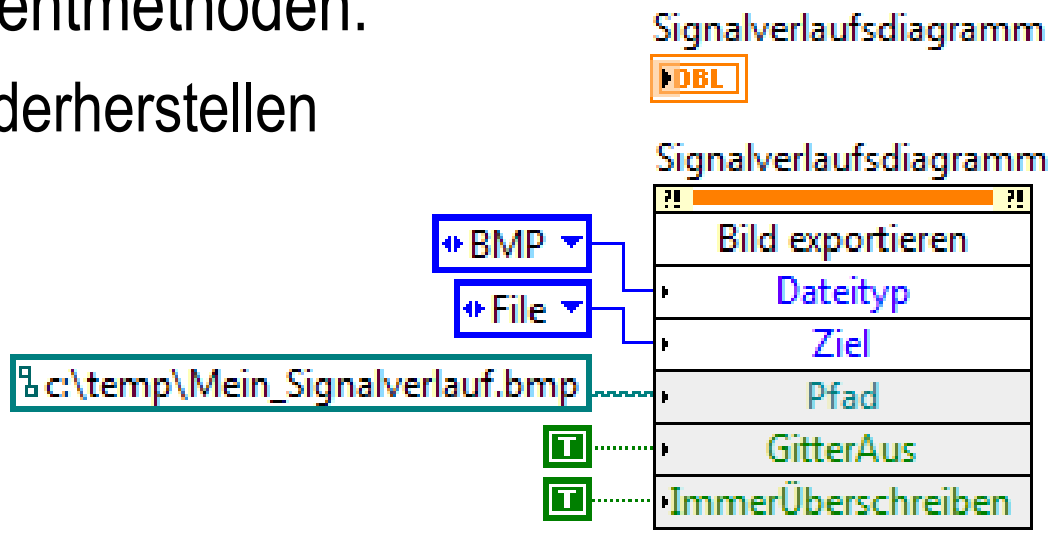
Methodenknoten

- Nehmen Handlungen an Objekten vor:
  - Abfragen der VI-Version
  - Ausdrucken des VI-Panels
  - Wiederherstellen aller Standardwerte
- Werden z. B. auf VIs oder Elemente angewandt
- Zu den meisten Methoden gibt es Parameter
- Methoden sind in der Kontexthilfe dokumentiert
- Arten von Methodenknoten:
  - Nicht sichtbar (implizit) verknüpft
  - Sichtbar (explizit) verknüpft



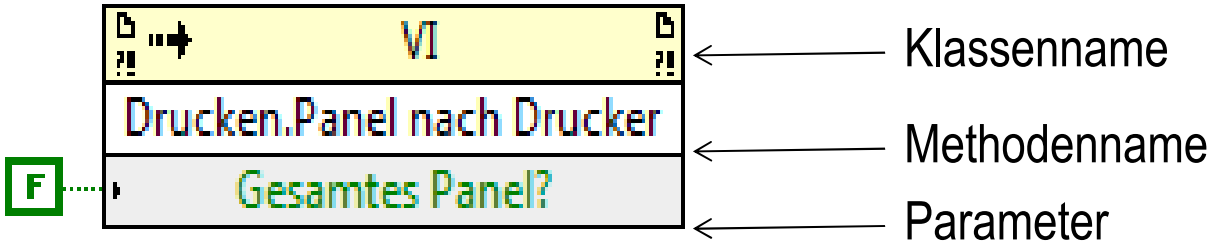
Elementmethoden

- Erstellen eines implizit verknüpften Methodenknotens:
  1. Rechtsklick auf das Element im Blockdiagramm und Auswahl von **Erstellen»Methodenknoten**
  2. Auswahl einer Methode aus dem Untermenü
- Beispiele für Elementmethoden:
  - Standardwert wiederherstellen
  - Bild exportieren



VI-Methoden

- Methodenknoten werden per VI-Server-Referenz mit dem aktuellen VI verknüpft
- Erstellen einer VI-Methode:
  1. Einfügen eines Methodenknotens in das Blockdiagramm
  2. Rechtsklick und Auswahl von **Klasse auswählen** zum Festlegen der Klasse
  3. Erneuten Rechtsklick und Auswahl von **Methode auswählen** zum Festlegen der Methode

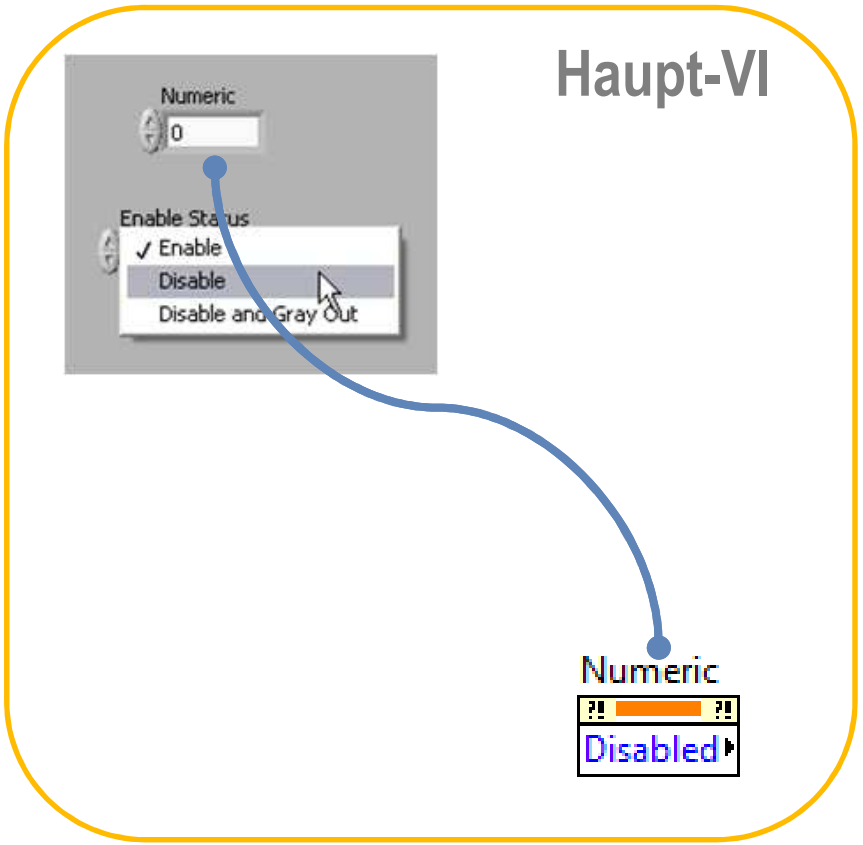


B-4. Elementreferenzen

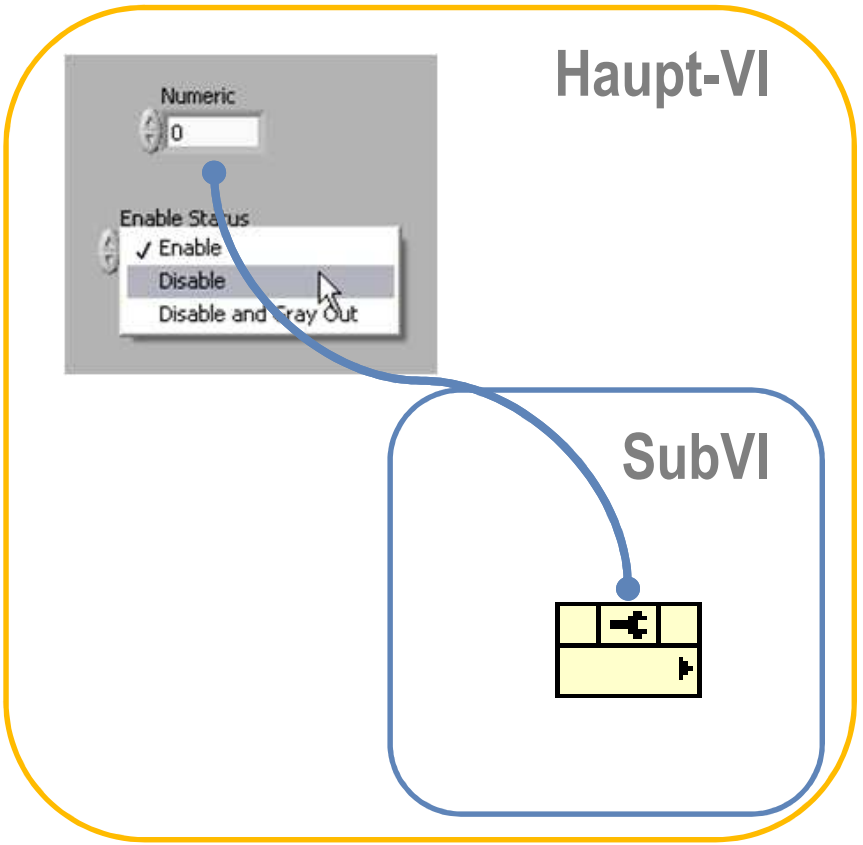
- Implizit und explizit verknüpfte Eigenschaftsknoten
- SubVIs erstellen
  - Mit der Option "SubVI erstellen"
  - Manuell
- VI-Server-Klasse auswählen

Elementreferenzen

Implizit verknüpfter Eigenschaftsknoten

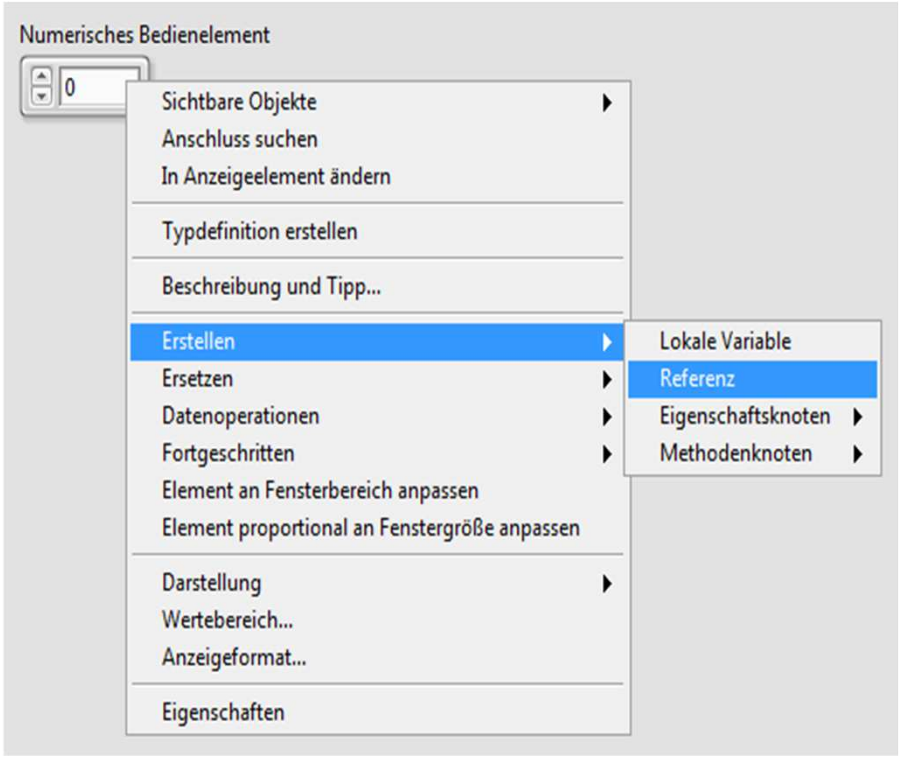


Explizit verknüpfter Eigenschaftsknoten



Elementreferenzen

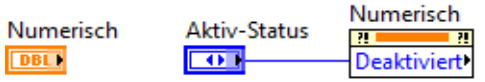
- Elementreferenzen stehen für Frontpanel-Objekte
- Werden mit allgemeinen Eigenschaftsknoten verbunden
- Werden an SubVIs übergeben



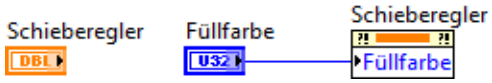
Erstellen von SubVIs

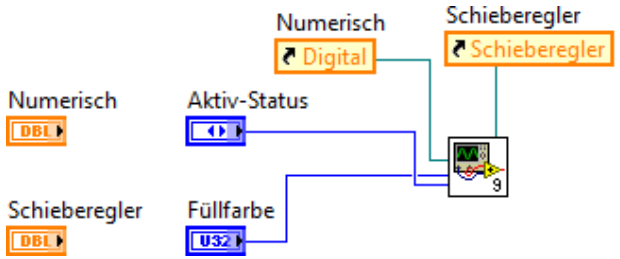
Zum Erstellen eines explizit verknüpften Eigenschaftsknotens in einem SubVI:

1. VI erstellen
2. Markieren des Blockdiagrammabschnitts, der das SubVI bilden soll



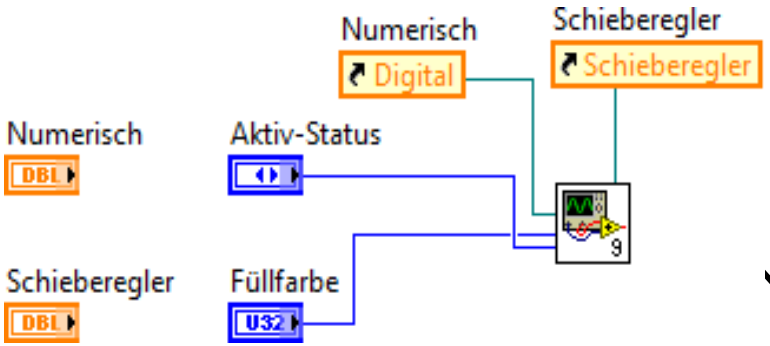
3. Auswahl von **Bearbeiten»SubVI erstellen** LabVIEW erstellt automatisch Elementreferenzen für das SubVI

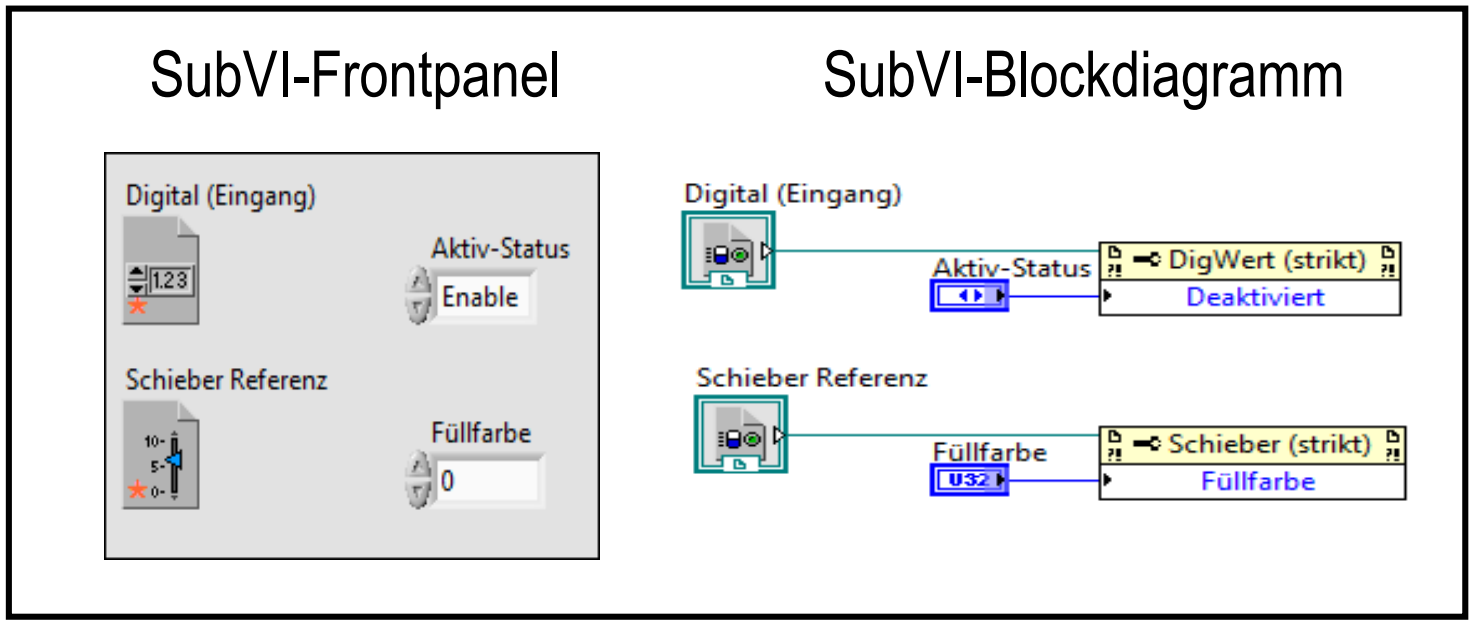




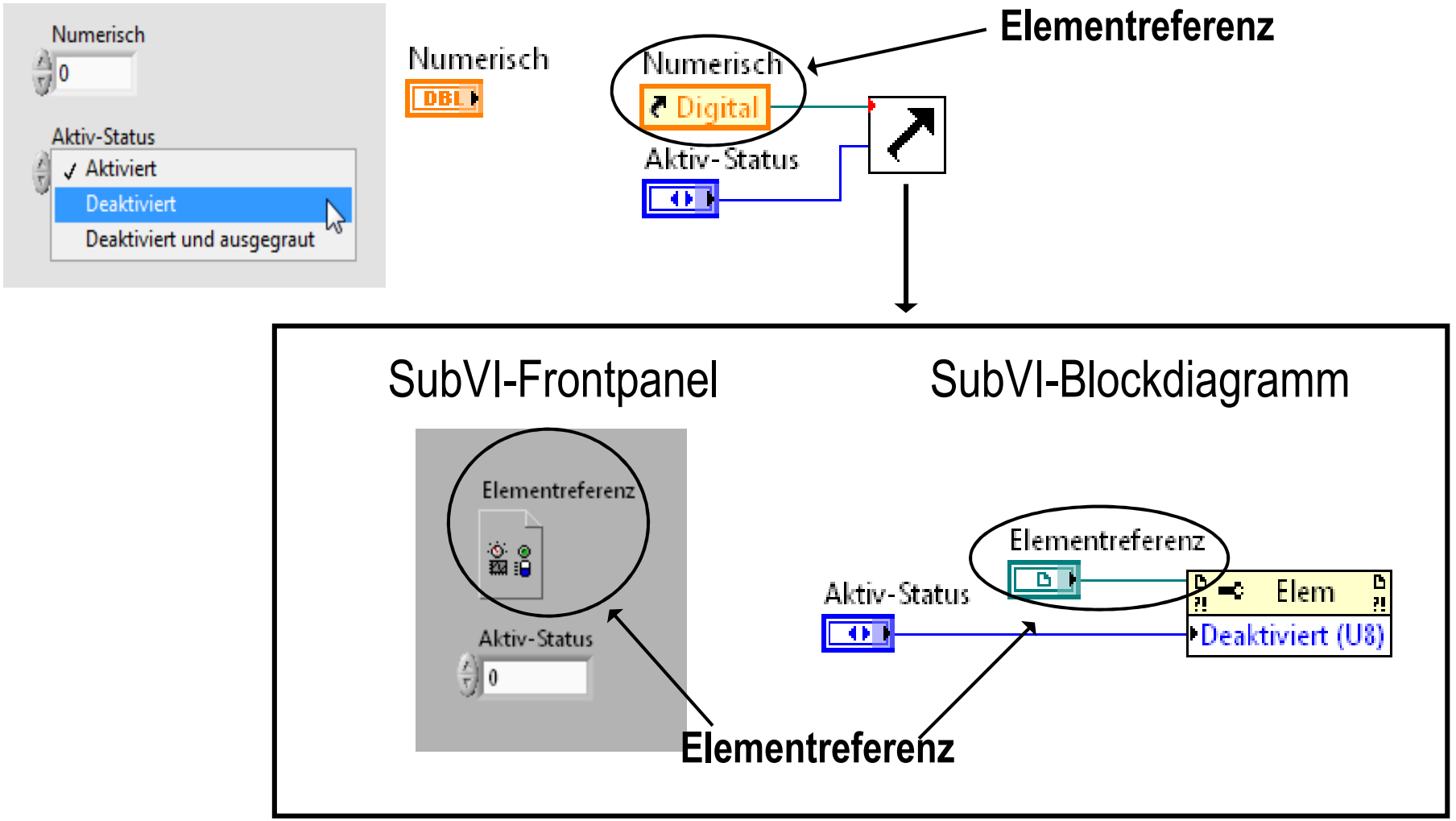
4. VI anpassen und speichern

Erstellen von SubVIs





Manuelles Erstellen von Elementreferenzen



Auswählen der VI-Serverklasse

- Nach dem Einfügen einer Elementreferenz in das SubVI-Frontpanel die VI-Server-Klasse des Elements angeben
  - Rechtsklick und Auswahl von **VI-Serverklasse**
  - Zur Angabe des Typs kann auch ein Element in eine Elementreferenz gezogen werden
- Die Klasse gibt den Typ der Elementreferenzen an, mit denen das SubVI arbeitet

C. Variablen

Variablen: Blockdiagrammelemente für den Zugriff auf und das Speichern von Daten entlegener Objekte

Variablenarten:

- Lokale Variablen: Speichern Daten in Bedien- und Anzeigeelementen des Frontpanels
- Globale Variablen: Speichern Daten in speziellen Speicherbereichen, auf die mehrere VIs zugreifen können
- Funktionale globale Variablen: Speichern Daten in Schieberegistern von While-Schleifen
- Umgebungsvariablen: Tauschen Daten zwischen verschiedenen Systemen in einem Netzwerk aus

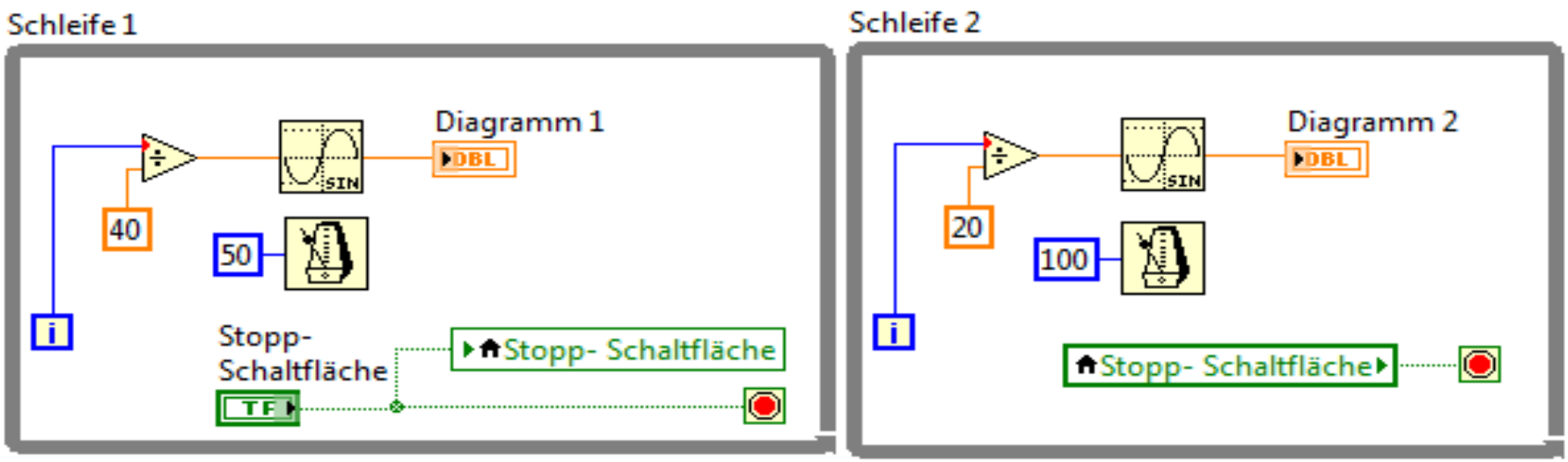
C-1. Lokale Variablen

Einsatzzweck

Lokale Variablen und boolesches Schaltverhalten

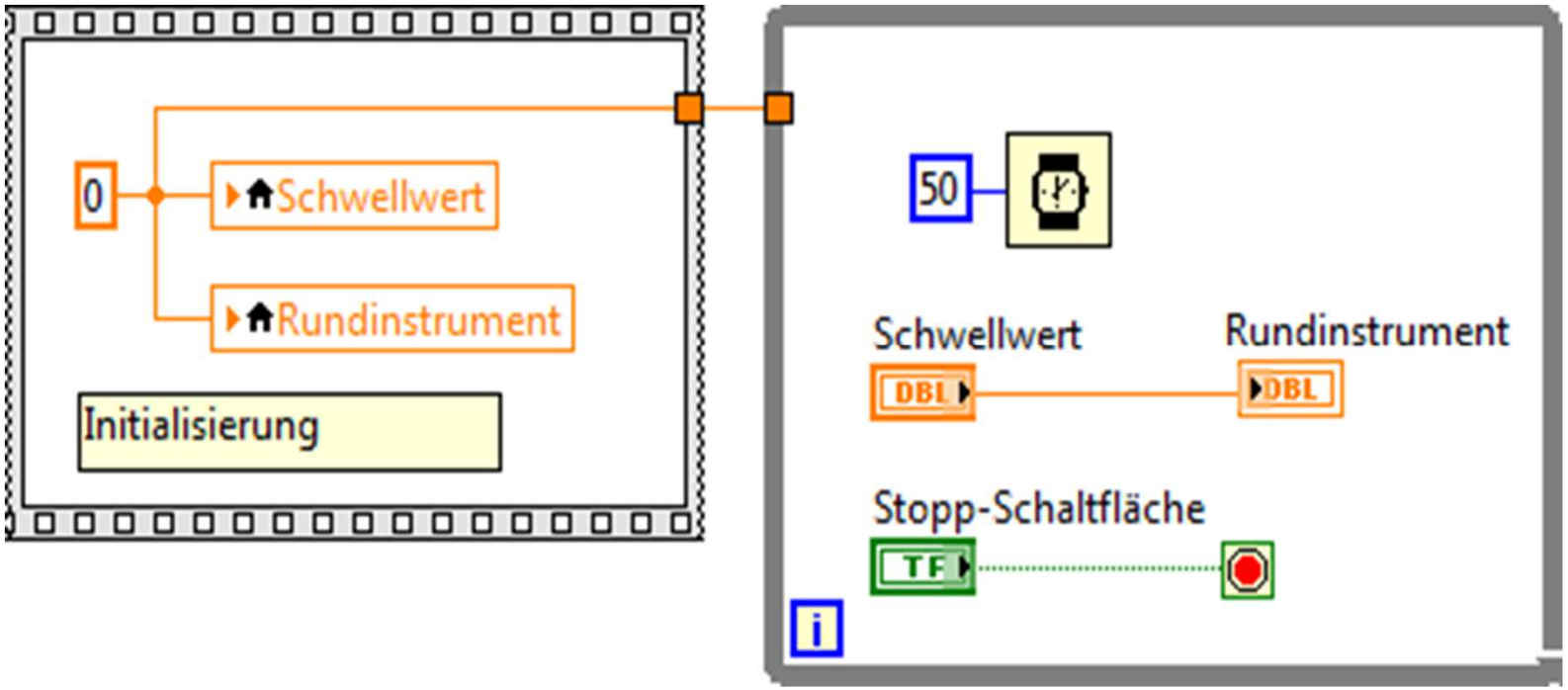
Erstellung lokaler Variablen

Lokale Variablen



Tauschen Daten innerhalb eines VIs aus

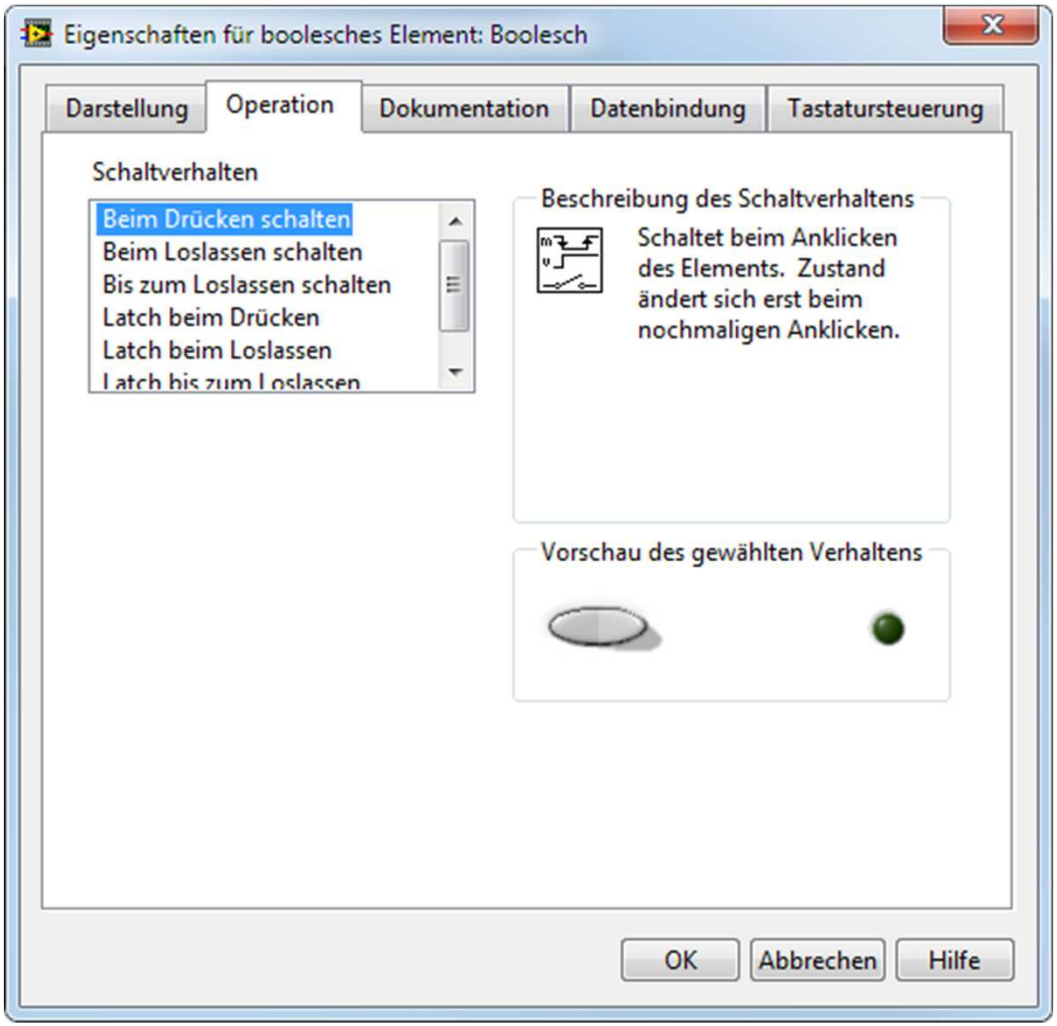
Lokale Variablen



Zum Ändern der Werte von Frontpanel-Elementen verwendet

Als Schalter, Öffner oder Taster konfiguriert

- Boolesche Bedienelemente, zu denen lokale Variablen gehören, müssen als Schalter, Öffner oder Taster konfiguriert sein
- Schaltverhalten ist nicht mit lokalen Variablen vereinbar



Erstellen lokaler Variablen

Erstellen und Verwenden lokaler Variablen

<Exercises>\LabVIEW Core 1\Demonstrations\Local Variables

DEMO

C-2. Globale Variablen

Einsatzzweck

Erstellung globaler Variablen

DEMO

C-3. Umgebungsvariablen

Einsatzzweck

Erstellung von Umgebungsvariablen

DEMO

D. Laufzeitprobleme

Definition

Vermeidung von Laufzeitproblemen

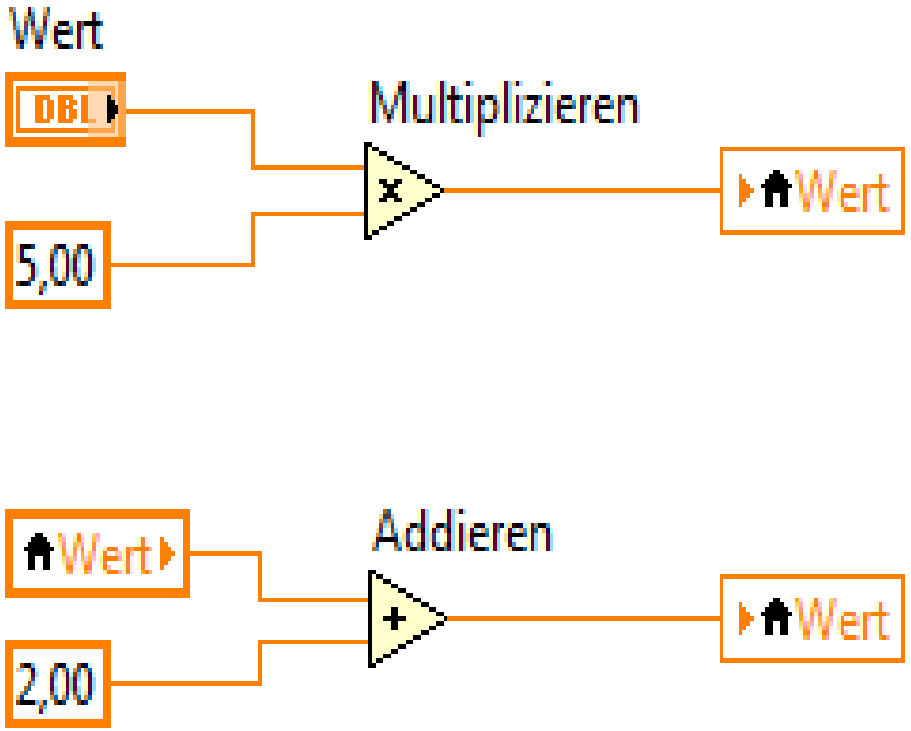
Überwachung gemeinsamer Ressourcen

Laufzeitprobleme

Ein Laufzeitproblem entsteht, wenn die Abfolge von Ereignissen oder die zeitliche Planung von Tasks ungewollt die Ausgangsgrößen beeinflusst

Laufzeitprobleme treten oft in Programmen auf, die mehrere Tasks parallel ausführen und Daten zwischen Tasks austauschen

Welchen Wert hat die Variable "Wert" als Letztes?



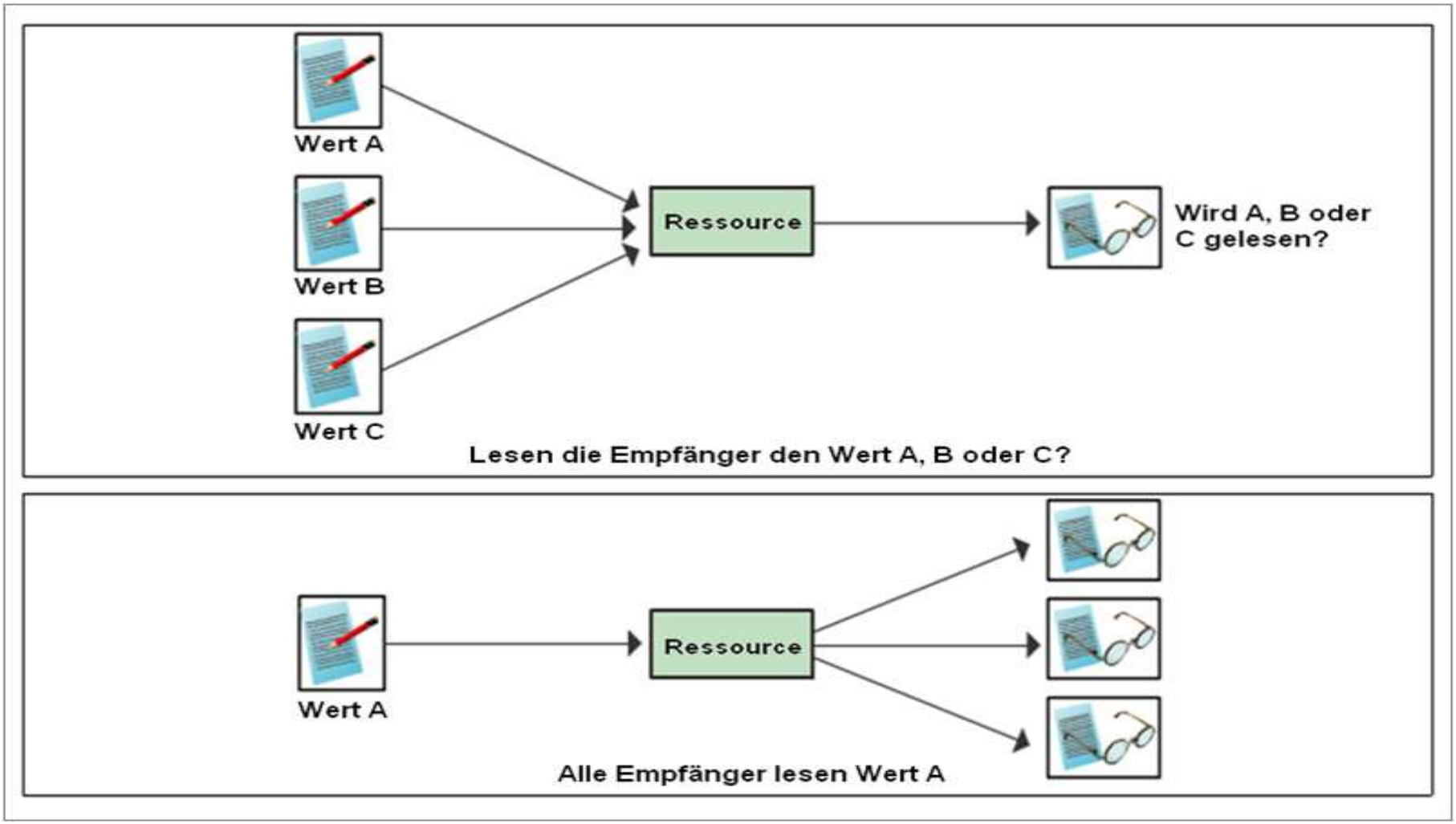
Vier Möglichkeiten:

- Wert = (Wert * 5) +2
- Wert = (Wert + 2) * 5
- Wert = (Wert * 5)
- Wert = Wert +2

Laufzeitprobleme

- Laufzeitprobleme sind schwer zu erkennen und zu beheben
- Beim Testen derartiger VIs kann Tausende Male das gleiche Ergebnis ausgegeben werden, während dennoch die Möglichkeit eines anderen Ergebnisses besteht
- Laufzeitprobleme lassen sich vermeiden durch:
  - Überwachung gemeinsamer Ressourcen
    - Eine Datenquelle und mehrere Datensenken
  - Eine gut strukturierte Anweisungsreihenfolge
  - Sparsame Verwendung von Variablen

Überwachung gemeinsamer Ressourcen

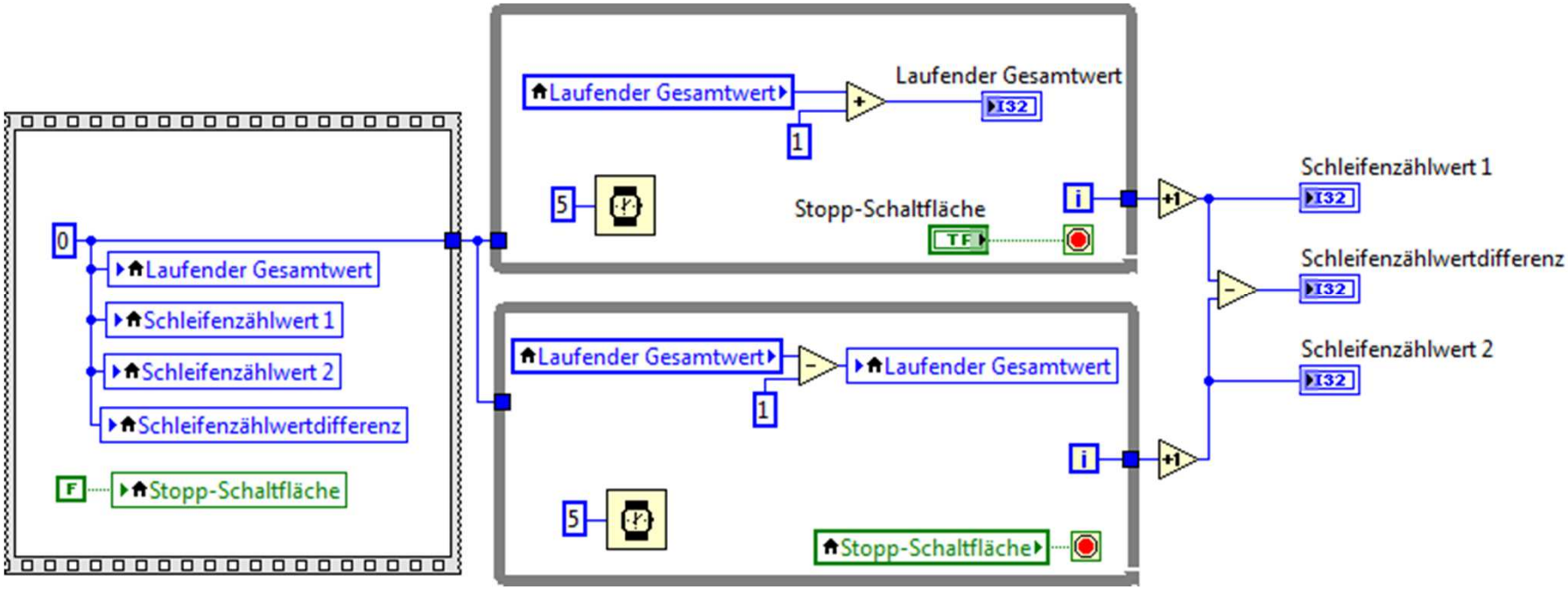


E. Entwurfsmuster für funktionale globale Variablen

- Motivation
  - Problem: Laufzeitprobleme beim Lesen-Modifizieren-Schreiben
  - Lösung: Verwendung des SubVIs mit der funktionalen globalen Variablen
- Funktionale globale Variable (FGV)
  - Definition
  - Grundlegender Aufbau
- Andere Anwendungsfälle

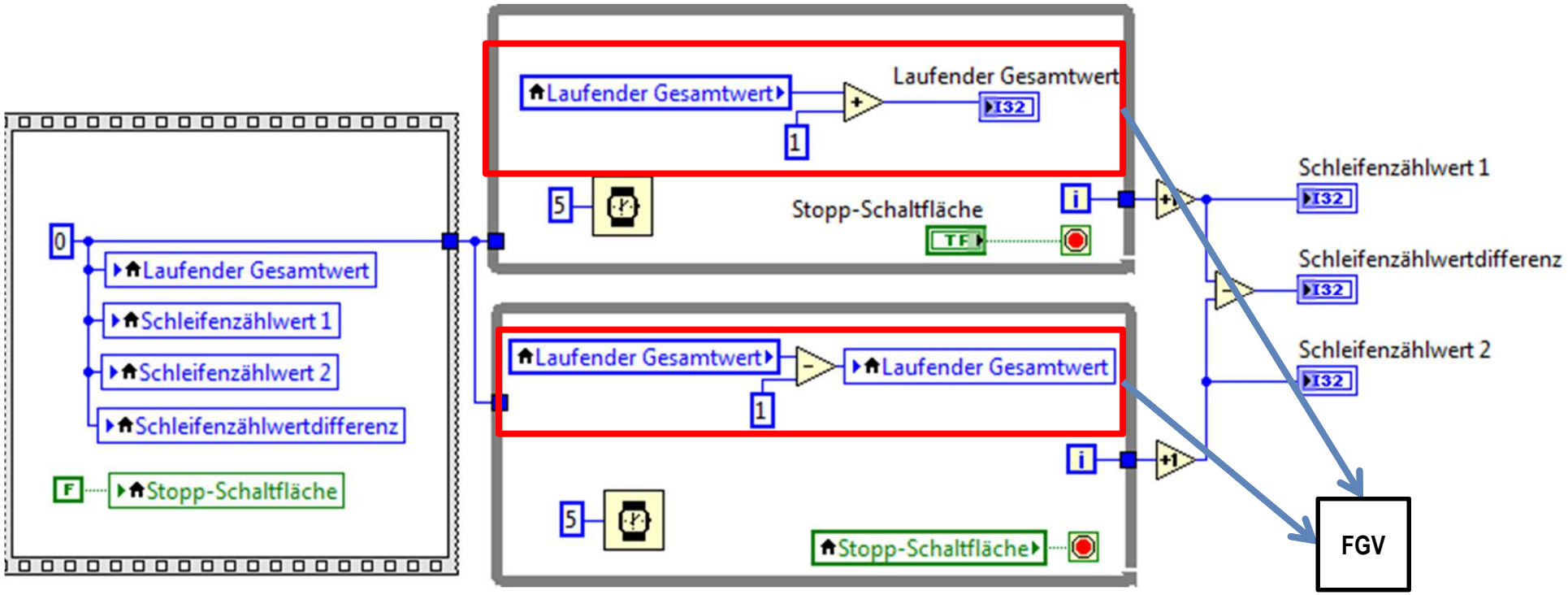
Problem: Laufzeitproblem (Lesen-Modifizieren-Schreiben)

- Das abgebildete Blockdiagramm hat ein Laufzeitproblem beim Lesen, Modifizieren und Schreiben
- Durch gleichzeitig stattfindende Vorgänge können falsche Daten ausgegeben werden



Lösung: Verwenden eines SubVIs basierend auf dem FGV-Entwurfsmuster

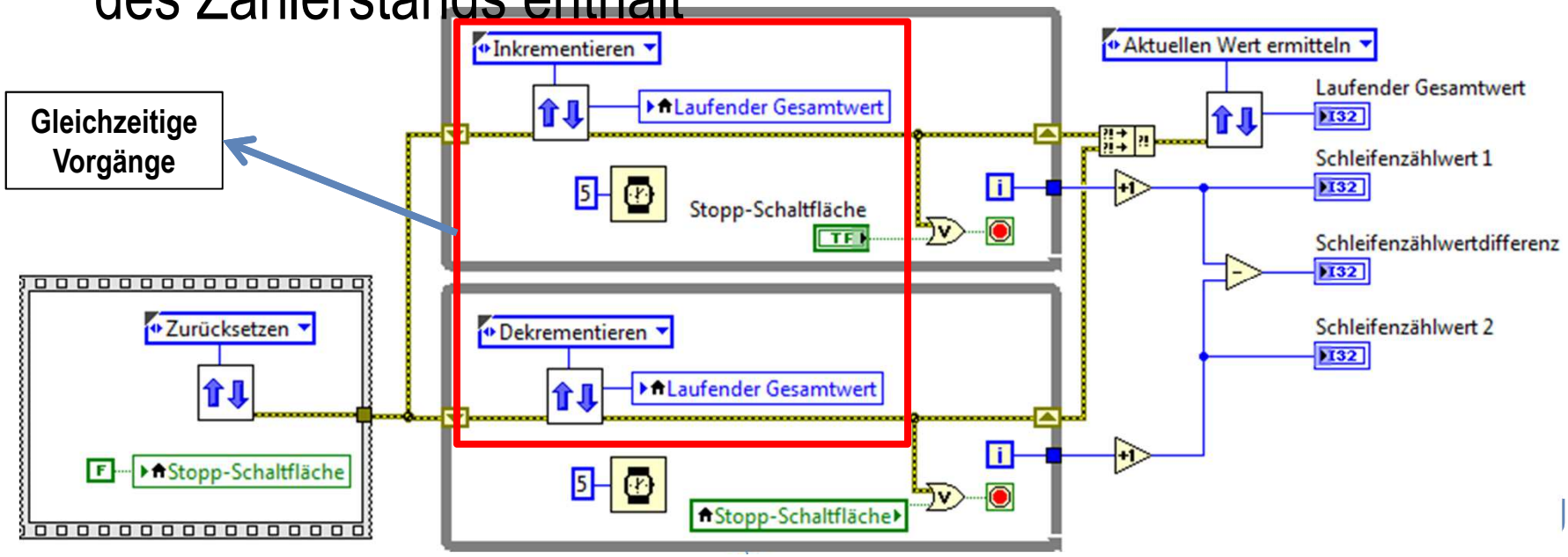
Fügen Sie die gemeinsam genutzte globale Ressource und alle Funktionen, die mit ihr arbeiten, in ein FGV-SubVI ein



Beispiel-VI "Bidirektionaler Zähler"

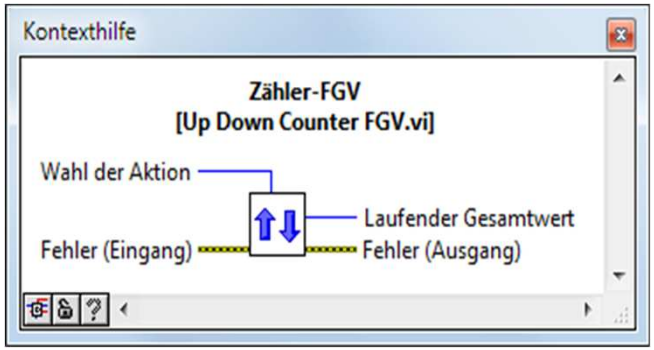
Das VI "Bidirektionaler Zähler" ist ein FGV-VI, das:

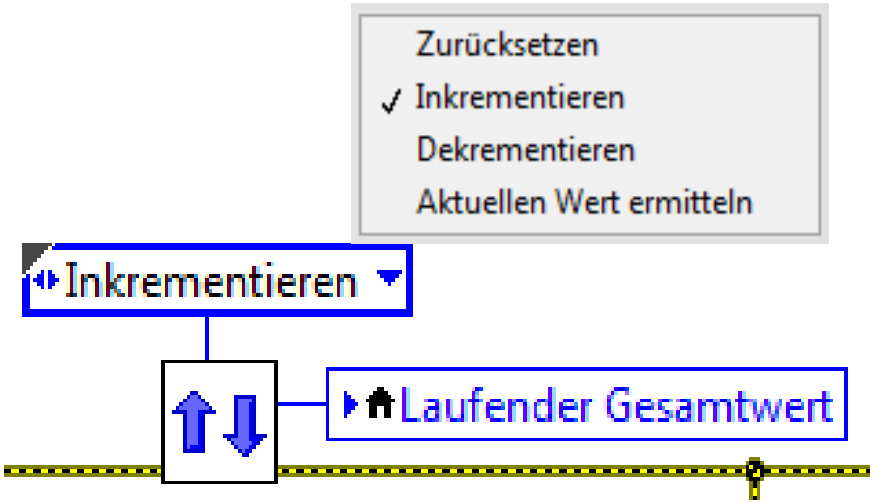
- Laufzeitprobleme beim gleichzeitigen Lesen, Modifizieren und Schreiben verhindert
- Methoden zum Zurücksetzen des Zählers und zum Ändern des Zählerstands enthält



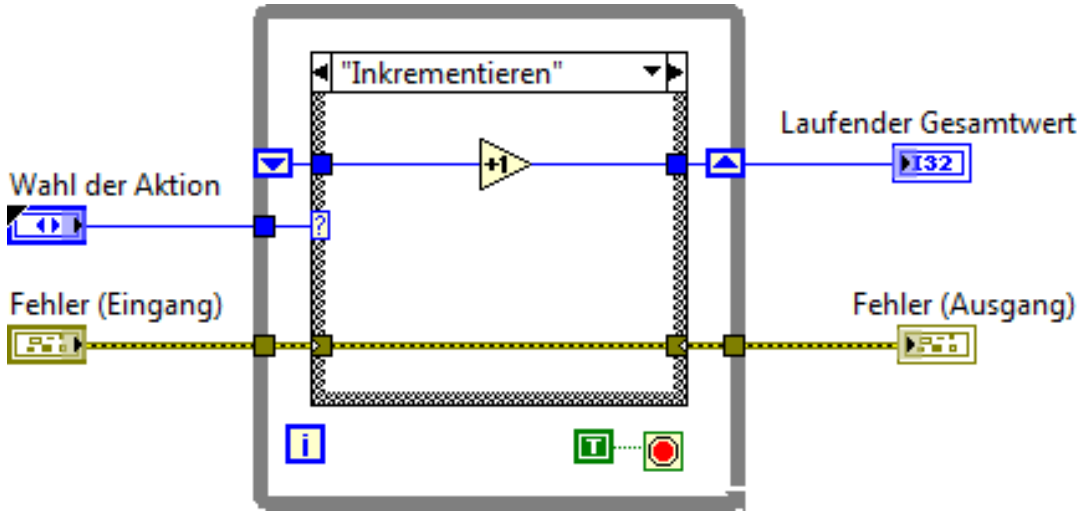
Bidirektionaler Zähler

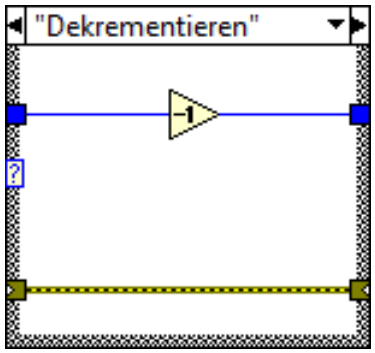
- **Wahl der Aktion:** Enum zur Auswahl des Betriebsmodus:
  - Zurücksetzen
  - Inkrementieren
  - Dekrementieren
  - Aktuellen Wert ermitteln
- **Laufender Gesamtwert:** Gibt den aktuellen Wert des VIs aus

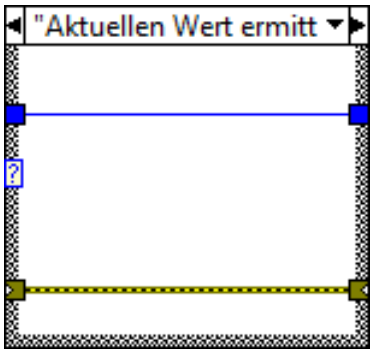


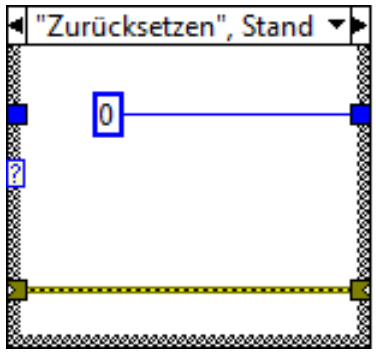


Aufbau des VIs "Bidirektionaler Zähler"









Funktionale globale Variable

Demo zum Umsetzen und Aufrufen einer FGV

Funktionale globale Variable

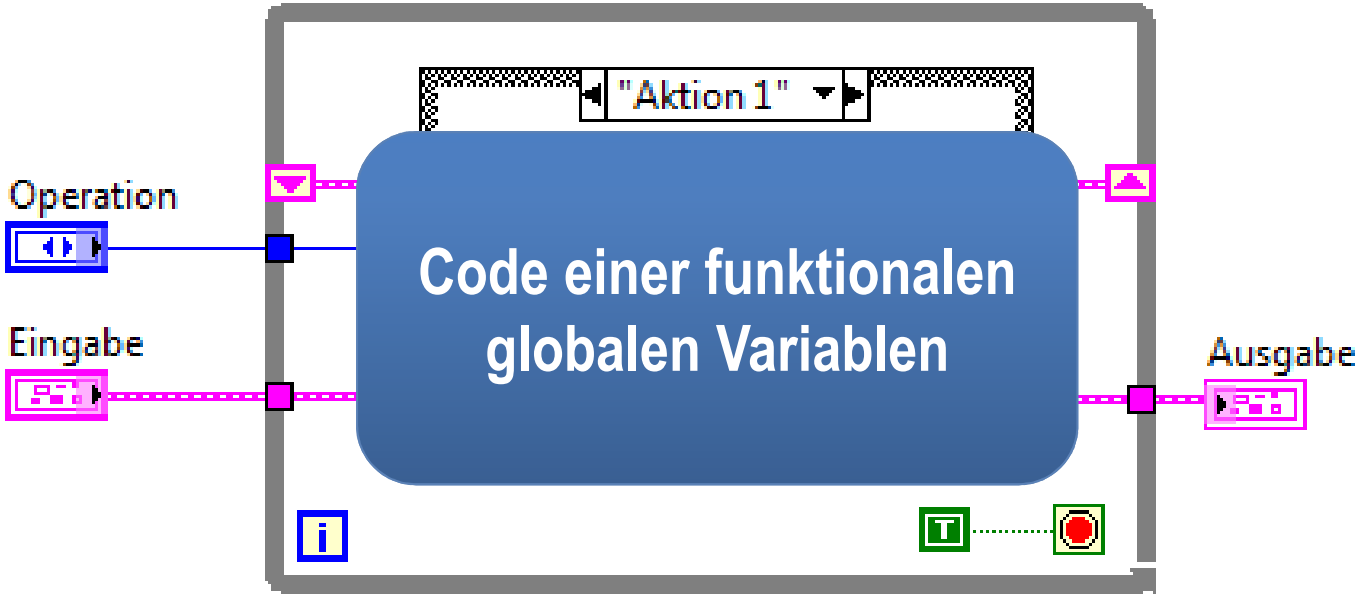
Funktionale globale Variable: Ablaufvariantes VI mit nicht initialisierten Schieberegistern zum Speichern globaler Daten. Ermöglicht oft Aktionen an den Daten.

- **Funktional**
  - Kann Berechnungen durchführen oder mit Daten arbeiten
- **Global**
  - Macht Daten innerhalb einer Applikationsinstanz verfügbar
- **Variable**
  - Speichert Daten
  - Funktioniert hinsichtlich des Datenaustauschs wie eine Variable

Grundlegender Aufbau einer FGV

Allgemeiner Aufbau einer funktionalen globalen Variablen:

- Nicht initialisiertes Schieberegister mit einer einmal durchlaufenden While-Schleife
- Case-Struktur



Andere Anwendungsfälle