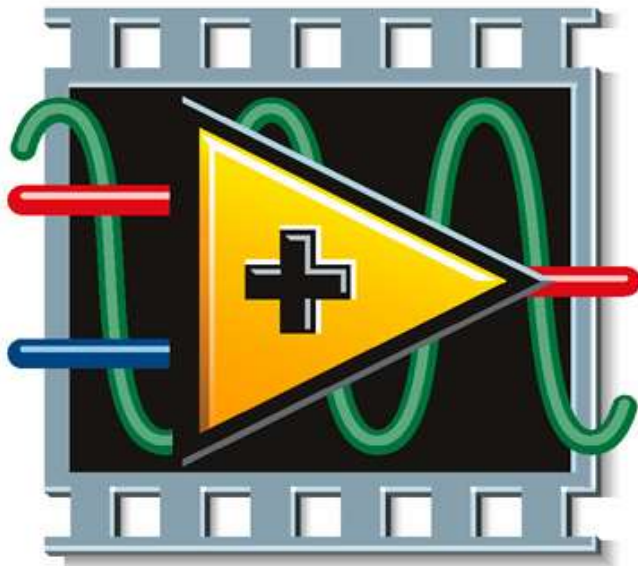


Alessandro Carlassare, M. Sc.
Prof. Dr.-Ing. Volker K. S. Feige

Teilpraktikum „Signalverarbeitung“

des Moduls "Sensorsysteme & Signalverarbeitung"

zur Prüfungsordnung B. ENG. ELEKTRO- UND INFORMATIONSTECHNIK (PO) 2016



Kurzeinführung in die LabVIEW-Programmierung

INHALTSVERZEICHNIS

1.	Einleitung	3
1.1	Vorbemerkung	3
1.2	Was ist LabVIEW?	3
2.	Erstellen eines LabVIEW-Projekts	5
3.	Frontpanel.....	8
3.1	Allgemeines.....	8
3.2	Ausgewählte Bedienelemente.....	9
3.3	Die Menüleiste des Frontpanels	10
4.	Das Blockdiagramm.....	12
4.1	Allgemeines.....	12
4.2	Darstellung der Datentypen im Blockdiagramm	13
4.3	Die Menüleiste des Blockdiagramms	15
5.	Hinweise zur Programmierung mit LabVIEW.....	16
5.1	Kontrollstrukturen [2].....	16
5.1.1	CASE-Anweisung („Verzweigung“)	16
5.1.2	FOR-Schleife.....	17
5.1.3	WHILE-Schleife.....	18
5.1.4	Sequenz	19
5.2	SubVI.....	20
6.	Einige grundlegende LabVIEW-Funktionen	24
7.	Vorbereitung und Aufgabenstellung	28
	Literaturverzeichnis.....	29

1. Einleitung

1.1 Vorbemerkung

Die vorliegende Anleitung soll einen raschen Einstieg in die Programmierung mit LabVIEW im Rahmen des Praktikums „Signalverarbeitung“ ermöglichen. Es sei darauf hingewiesen, dass Kenntnisse einer „klassischen“ höheren Programmiersprache wie C, C++, Visual Basic, Java oder PASCAL eine erfolgreiche Einarbeitung in die Thematik erleichtern. Bei der Erstellung der Praktikumsunterlagen wurden insbesondere elementare Kenntnisse sowohl der Softwaretechnik als auch der Mathematik, wie sie z. B. in den Grundlagenvorlesungen gelehrt werden, als bekannt vorausgesetzt. Zur Aufarbeitung evtl. vorhandener Wissenslücken sei an dieser Stelle auf das Schrifttum verwiesen.

1.2 Was ist LabVIEW?

LabVIEW ist die Abkürzung für „Laboratory Virtual Instrument Engineering Workbench“ und bezeichnet eine Programmierumgebung der Firma National Instruments, die insbesondere für die Erstellung von Anwendungen der Mess-, Steuer- sowie Regelungstechnik in den 1980er Jahren entwickelt wurde. LabVIEW grenzt sich von den sonst üblichen Programmiersprachen wie z. B. C++ oder Visual Basic dadurch ab, dass die Software nicht mithilfe eines textbasierten Quellcodes „geschrieben“, sondern unter Verwendung einer als „G“ bezeichneten, datenflussorientierten graphischen Programmiersprache entwickelt wird. Eine in LabVIEW erstellte Anwendung wird im Allgemeinen als „Virtuelles Instrument“ (engl. „Virtual Instrument“) oder kurz als „VI“ bezeichnet. Ein VI besteht stets aus einer Benutzerschnittstelle (Frontpanel) und dem eigentlichen Programmcode, der als Blockdiagramm bezeichnet wird. Die Programmierung eines VIs erfolgt dadurch, dass die einzelnen Elemente durch „Leitungen“ miteinander verknüpft werden. Die für den Anwender relevanten Bedienelemente werden auf dem Frontpanel platziert. Die Entwicklungsumgebung generiert im Blockdiagramm automatisch dazugehörige Terminals, die als Datenquelle oder -senke fungieren. Die Elemente im Blockdiagramm können die Elemente im Frontpanel ansteuern und mittels Operatoren (auch „Knoten“ genannt) können Berechnungen oder weitere Verarbeitungen der Daten erfolgen. Ein sehr einfaches Beispiel für ein VI ist in Abbildung 1 dargestellt. Die Anwendung dient dazu, zwei Variablen miteinander zu addieren und das Ergebnis der Operation anzuzeigen. Im linken Fenster ist die Benutzeroberfläche (Frontpanel) zu sehen, das rechte Fenster zeigt das dazugehörige Blockdiagramm. Hierbei dienen die beiden mit „Zahl a“ und „Zahl b“ bezeichneten Eingabefelder jeweils als Datenquelle. Das Ausgabefeld

„Ergebnis“ stellt die Datensenke dar, wodurch das Ergebnis der Operation im Frontpanel dargestellt wird. Das Datenflusskonzept von LabVIEW sieht vor, dass zunächst die beiden Eingabefelder abgefragt werden, im Anschluss daran werden die Daten zu dem Operationsknoten transportiert. Liegen alle erforderlichen Daten am Summationsknoten an, so wird die Addition durchgeführt und das Ergebnis an den nächsten Knoten, in diesem Fall das Ausgabefeld, weitergegeben. Eine besondere Eigenschaft des Datenflusskonzepts ist jedoch, dass die Reihenfolge, in der die Befehle ausgeführt werden, nicht immer in eindeutiger Weise vorhersehbar ist. LabVIEW stellt jedoch einige Funktionen zur Verfügung, die die Abläufe kontrollieren bzw. beeinflussen kann.

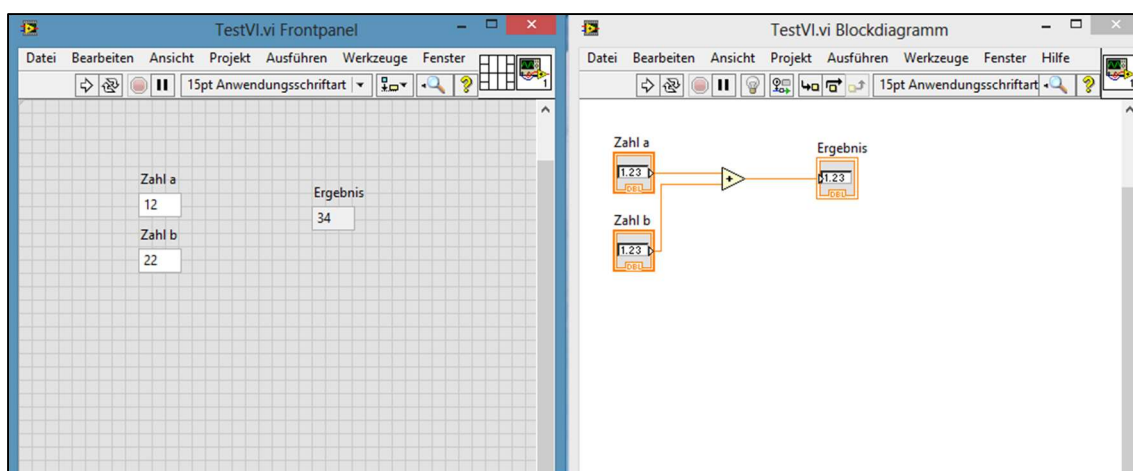


Abbildung 1: Benutzeroberfläche der LabVIEW-Entwicklungsumgebung

Mithilfe von LabVIEW können selbstverständlich auch wesentlich komplexere Anwendungen als das soeben vorgestellte Beispiel erstellt werden. Die Entwicklungsumgebung stellt umfangreiche Bibliotheken zur Datenerfassung, -verarbeitung, -visualisierung sowie zur Datenspeicherung bereit. Ein wesentlicher Vorteil von LabVIEW ist, dass auch die Einbindung von Hardware in der Regel problemlos möglich ist, da eine Vielzahl von Protokollen unterstützt werden. Dies hat dazu geführt, dass sich die Software als Quasi-Standard im Bereich der industriellen Fertigung, beispielsweise zur Automatisierung von Abläufen im Prüffeld als auch im Bereich der Forschung- und Entwicklung etabliert hat.

2. Erstellen eines LabVIEW-Projekts

Grundsätzlich ist es in LabVIEW möglich, einzelne VIs zu erstellen und diese selbstständig auszuführen. Da eine LabVIEW-Anwendung durchaus aus mehreren VIs bzw. SubVIs (s. hierzu Abschnitt 5.2) bestehen kann, ist es sinnvoll ein neues LabVIEW-Projekt zu erstellen. In ein LabVIEW-Projekt können die erstellten VIs und die weiteren benötigten Dateien integriert werden. Dies hat unter anderem den Vorteil, dass Dateien innerhalb eines Projekts untereinander referenziert werden, wobei die Dateien nicht zwingend innerhalb des gleichen Verzeichnisses abgelegt werden müssen. Das Anlegen eines Projekts ist ferner zwingend erforderlich, falls Anwendungen an weitere Geräte (engl. „targets“) wie Echtzeitsysteme oder FPGAs verteilt (engl. „deployed“) werden sollen. Zum Erstellen eines neuen Projekts muss zunächst die Entwicklungsumgebung¹ gestartet werden. Dies geschieht, wie bei Windows-Anwendungen üblich, durch Doppelklicken auf die dazugehörige Verknüpfung (z. B. „NI LabVIEW 2014“) auf dem Desktop oder über den entsprechenden Eintrag im Startmenü. Bei neueren Windows-Versionen findet man die Verknüpfung im Bereich „Alle Apps“. Das Symbol zum Starten von LabVIEW ist in Abbildung 2 dargestellt.



Abbildung 2:
LabVIEW-Symbol

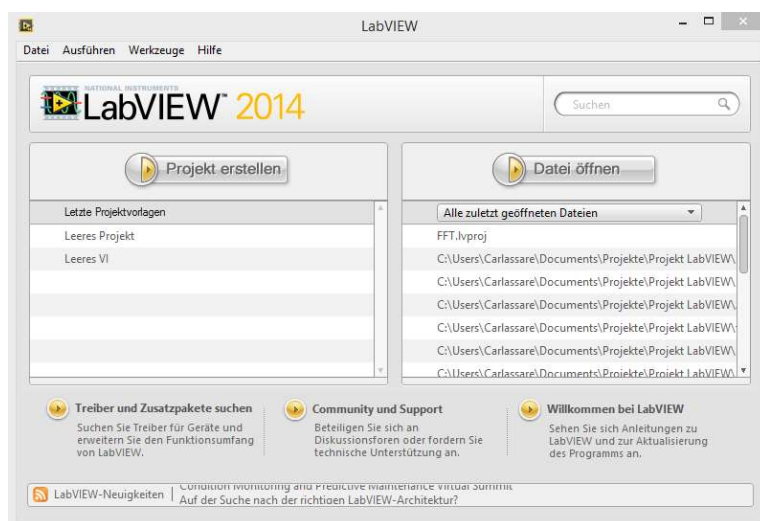


Abbildung 3: LabVIEW, Startbildschirm

Sobald die Programmierungsumgebung LabVIEW geladen ist, erscheint das in Abbildung 3 dargestellte Startfenster. Ein neues Projekt kann durch Anklicken des Buttons „**Projekt erstellen**“ auf der linken Seite des Menüs erstellt werden. Im darauffolgenden Menü „Projekt erstellen“

¹ Für die Erstellung der Screenshots wurde die Version „LabVIEW 2014“ unter dem Betriebssystem Microsoft Windows 8.1 verwendet. Neuere bzw. ältere Versionen von LabVIEW und/oder Windows weichen ggf. in einigen Details geringfügig ab, die grundsätzliche Vorgehensweise bleibt jedoch unverändert.

(s. Abbildung 4) können diverse Optionen zum Erstellen eines neuen Projekts gewählt werden. Es stehen auch bereits vorkonfigurierte Projekte, sogenannte Templates, zur Verfügung. Im Rahmen des Praktikums ist stets der Startpunkt „**Leeres Projekt**“ auszuwählen. Durch anschließendes Anklicken der Schaltfläche „**Fertigstellen**“ wird das entsprechende Projekt angelegt.

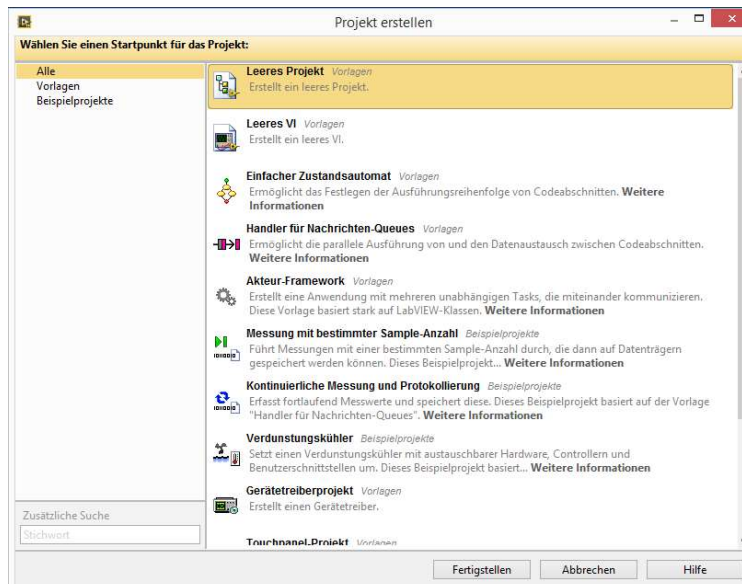


Abbildung 4: Auswahlmennü "Projekt erstellen"

LabVIEW erstellt nun ein neues Projekt entsprechend der gewählten Einstellungen. Anschließend erscheint der „Projekt-Explorer“ (vgl. Abbildung 5) auf dem Bildschirm. Mithilfe dieses Tools ist es möglich, neue LabVIEW-Projekte zu erstellen oder vorhandene Projekte zu verwalten.

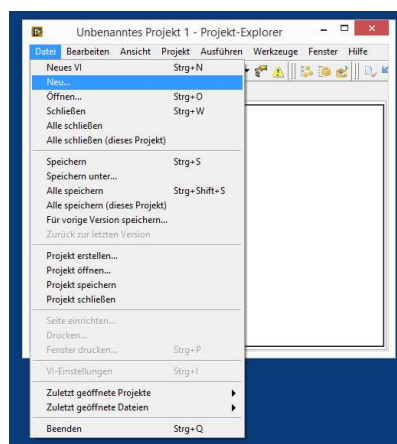


Abbildung 5: Projekt-Explorer

Mithilfe des Projekt-Explorers können ebenfalls neue VIs angelegt werden. Dies geschieht durch Anklicken des Menüpunkts „**Datei**“ und durch anschließendes Auswählen des Menüeintrags „**Neu...**“. Im darauffolgenden Dialogfenster „**Neu**“ ist der Eintrag „**Leeres VI**“ sowie die Option „**Zu Projekt hinzufügen**“ durch Auswählen der Checkbox im unteren Bereich auszuwählen. Durch Bestätigen mit „**OK**“ wird das VI neu erstellt und dem Projekt hinzugefügt.

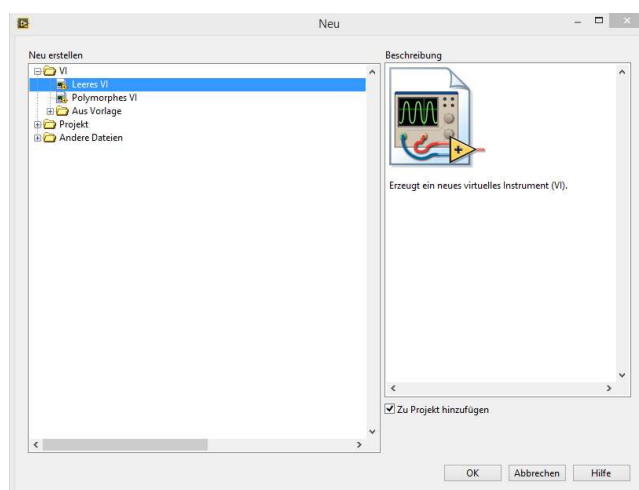


Abbildung 6: Neues VI hinzufügen, Dialog

Das VI ist nun im Projekt-Explorer, in der Regel unter der Standardbezeichnung „Unbenannt“, zu finden. Es erscheinen ferner zwei neue Fenster. Das Fenster mit dem weißen Hintergrund ist für die Bearbeitung des Blockdiagramms vorgesehen, das Fenster mit dem grauen, karierten Hintergrund dient der Erstellung des Frontpanels (Benutzerschnittstelle). Als nächster Schritt sollten Projekt und VI benannt und gespeichert werden. Dies kann durch Anklicken des Menüs „Datei“ und Auswahl des Eintrags „Alle speichern“ im Projekt-Explorer erfolgen. In den folgenden beiden Dialogen werden Sie nacheinander dazu aufgefordert, jeweils einen Namen für das gesamte Projekt sowie für das soeben erstellte VI zu vergeben. Empfehlenswert ist die Verwendung von „sprechenden“ Dateinamen, aus denen sich die Funktion des jeweiligen VIs unmittelbar ableiten lässt. Nach Eingabe des gewünschten Dateinamens kann das Dialogfeld jeweils durch Anklicken der Schaltfläche „OK“ geschlossen werden. Projekt und VI sind damit gespeichert.

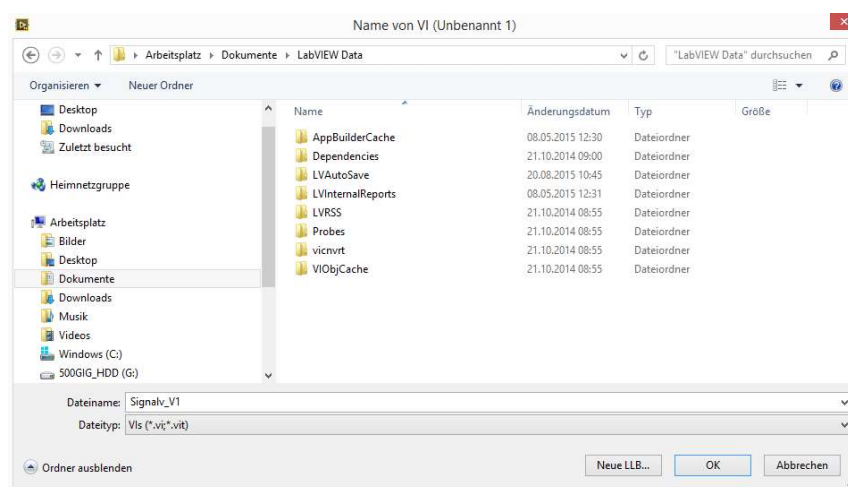


Abbildung 7: Dialog zum Speichern des VIs

3. Frontpanel

3.1 Allgemeines

Wie bereits eingangs erwähnt, besteht jedes VI aus einer Benutzerschnittstelle (Frontpanel) sowie der dazugehörigen Logik (Blockdiagramm). Im Frontpanel erfolgen die Eingaben durch den Benutzer sowie die Ausgaben der Datenwerte, die ggf. zuvor im Blockdiagramm verarbeitet wurden und dem Benutzer angezeigt werden sollen.

LabVIEW stellt eine große Anzahl von Bedienelementen zur Verfügung. Zum Anzeigen der verfügbaren Elemente klicken Sie mit der rechten Maustaste in einen leeren Bereich des Frontpanel-Fensters. Es erscheint ein nach Kategorien sortiertes Palettenmenü. Die gewünschten Elemente können mit der Maus nach dem „Drag & Drop“-Prinzip auf die Frontpanel-Oberfläche gezogen werden.

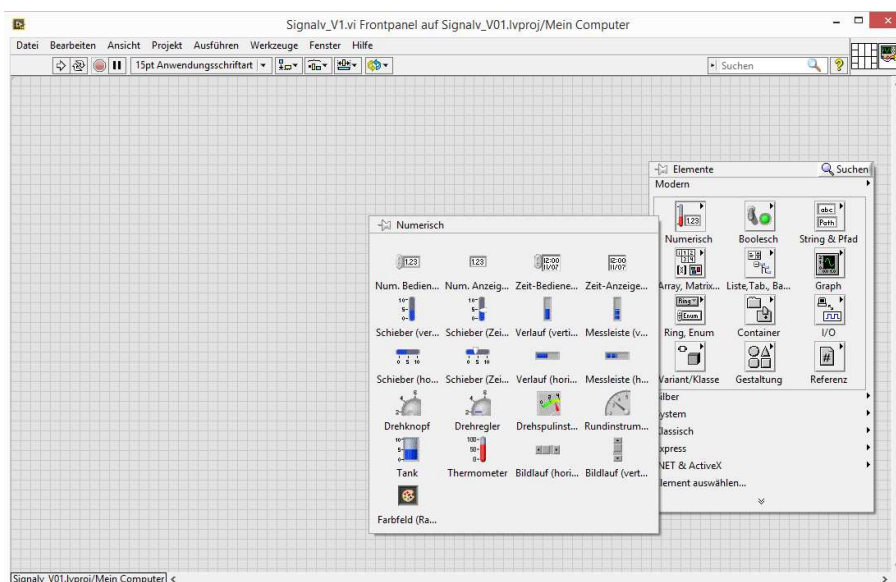
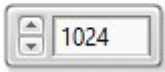


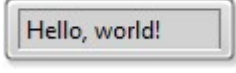
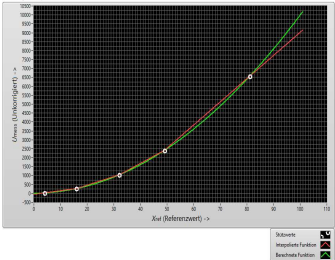



Abbildung 8: Palettenmenü mit geöffneter Unterkategorie "Numerisch"

Sofern der Name des gewünschten Elements bekannt ist, kann mithilfe des Buttons „Suchen“ oben rechts gezielt nach dem jeweiligen Element gesucht werden. Die Ein- und Ausgabeelemente sind nach Datentyp sortiert. Die zur Verfügung stehenden Kategorien sind in Abbildung 8 dargestellt. Im Folgenden werden einige wichtige Elemente, die im Rahmen dieses Praktikums benötigt werden kurz erläutert. Daneben sei auf das Schrifttum [1][2] sowie auf die umfangreiche Hilfefunktion von LabVIEW hingewiesen, die innerhalb des Programms durch Betätigen der Taste „F1“ auf der Tastatur aufgerufen werden kann. Ferner sei angemerkt, dass nahezu jedes Element über Eigenschaften verfügt, die durch Aufrufen des entsprechenden Menüs (Rechtsklick auf das gewünschte Element im Frontpanel, dann den Menüpunkt „Eigenschaften“ auswählen). Eine detaillierte Auflistung der Eigenschaften sämtlicher Elemente ist aufgrund der großen Anzahl an dieser Stelle nicht möglich.

3.2 Ausgewählte Bedienelemente

<p>Anzahl Punkte</p> 	<p><u>Numerisches Bedienelement:</u> Dient der Eingabe von Zahlenwerten in den Datentypen „Unsigned“, „Integer“ oder „Float“ oder „Komplex“ im Dual-, Oktal, Dezimal- oder Hexadezimalzahlensystem. Über die neben dem Eingabefeld befindlichen, optionalen Buttons (Pfeile) kann die Zahl in- oder dekrementiert werden. Der gültige Eingabebereich kann beliebig vorgegeben werden.</p>
<p>U_{mess,korr}</p> 	<p><u>Numerisches Anzeigelement:</u> Dient der Ausgabe von Zahlenwerten analog zum numerischen Bedienelement. Die Anzeigeparameter wie Zahlenformat, Anzahl der Nachkommastellen, etc. können beliebig angepasst werden.</p>
<p>Eingabe</p> 	<p><u>„String“-Bedienelement:</u> Dient der Eingabe beliebiger Zeichenketten („Strings“).</p>
<p>Ausgabe</p> 	<p><u>„String“-Ausgabeelement:</u> Dient der Ausgabe beliebiger Zeichenketten („Strings“). Unter Umständen ist die Ausgabe von Zahlenwerten im „String“-Format sinnvoll, da auf diese Weise eine umfangreichere Bearbeitung vor der Ausgabe möglich ist. So kann beispielsweise ein Präfix oder ein Suffix hinzugefügt werden, was sich insbesondere bei der Visualisierung physikalischer Größen (mit Einheiten) als zweckmäßig erweisen kann.</p>
<p>Darstellung der Integration</p> 	<p><u>XY-Graph:</u> Dient der Visualisierung von funktional zusammenhängenden Zahlenwerten wie Zeitverläufe, Spektren, Messreihen, etc. Die Darstellung erfordert eine Konvertierung der Daten in das „Cluster“-Format.</p>
<p>Start</p> 	<p><u>„Virtueller“ Druckschalter:</u> Dient als Eingabeelement für eine Bool'sche Variable. Schalter existieren in verschiedenen Ausführungen und können sowohl bzgl. des Aussehens als auch hinsichtlich des Schaltverhaltens umfassend angepasst werden.</p>


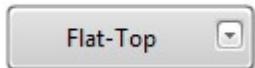
<p>Bereit</p> 	<p><u>„Virtuelle“ LED:</u> Mithilfe der LEDs kann der Zustand von Bool'schen Variablen visualisiert werden.</p>
<p>Fensterfunktion</p> 	<p><u>„Ring“-Menü:</u> Bietet ein „Dropdown“-Menü, dessen Einträge beliebig konfiguriert werden können. Jedem Eintrag wird ein ganzzahliger Wert zugewiesen, der z. B. als Argument für eine CASE-Anweisung verwendet werden kann.</p>







Tabelle 1: Bedienelemente Frontpanel, Auswahl

3.3 Die Menüleiste des Frontpanels

Im oberen Bereich des Frontpanel-Fensters ist eine Menüleiste („Toolbar“) zu finden, mit dessen Hilfe auf einige, häufig benötigte Funktionen zugegriffen werden kann. Die einzelnen Funktionen sind in Abbildung 9 dargestellt und werden in Tabelle 2 näher erläutert.



Abbildung 9: Frontpanel, Menüleiste

	VI ausführen [Tastaturkürzel: „ Strg + R “]
	VI wird ausgeführt
	VI kann aufgrund eines Fehlers nicht ausgeführt werden. Anklicken des Symbols öffnet eine Fehlerliste.
	VI kontinuierlich ausführen
	Ausführung des VIs abbrechen [Tastaturkürzel: „ Strg + . “]
	Ausführung des VIs unterbrechen

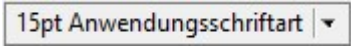

	Öffnet ein Menü zur Formatierung der Schrift.
	Schaltflächen zur Anpassung der Bedienelemente (ausrichten, Größe anpassen, Abstände angleichen, Ebenen, Gruppierung, etc.)

Tabelle 2: Frontpanel, Menüleiste, Erläuterungen zu den Symbolen

4. Das Blockdiagramm

4.1 Allgemeines

Das Blockdiagramm entspricht dem Quellcode textbasierter Programmiersprachen. Zu den Objekten, die im Blockdiagramm zur Programmierung verwendet werden gehören u. a. Terminals, SubVI, Funktionen, Konstanten, Strukturen und Leitungen. Da Frontpanel und Blockdiagramm in getrennten Fenstern ausgeführt werden, kann mithilfe der Windows-Taskleiste in die jeweils andere Ansicht gewechselt werden. Ein schnelles Umschalten erreicht man ferner durch Betätigen der Tastenkombination „**Strg+E**“.

Um den Zusammenhang zwischen Blockdiagramm und Frontpanel zu erläutern wird das Beispiel aus Abbildung 1 erneut aufgegriffen. Man erkennt, dass zu jedem Bedienelement auf dem Frontpanel („Zahl a“, „Zahl b“ und „Ergebnis“) ein Terminal mit identischer Bezeichnung im Blockdiagramm existiert. Diese Terminals dienen als Datenquelle bzw. -senke und werden selbstständig von LabVIEW angelegt, sobald ein Element dem Frontpanel hinzugefügt wird. Um die gewünschte Funktionalität herzustellen, muss ferner ein Element ausgewählt werden, das die vorhandenen Terminals auf geeignete Weise verknüpft. Im Beispiel aus Abbildung 1 ist dies der Operator „Addition“. Hierzu klickt man analog zum Frontpanel mit der rechten Maustaste auf eine freie Stelle auf der Oberfläche des Blockdiagramms – ein Palettenmenü erscheint.

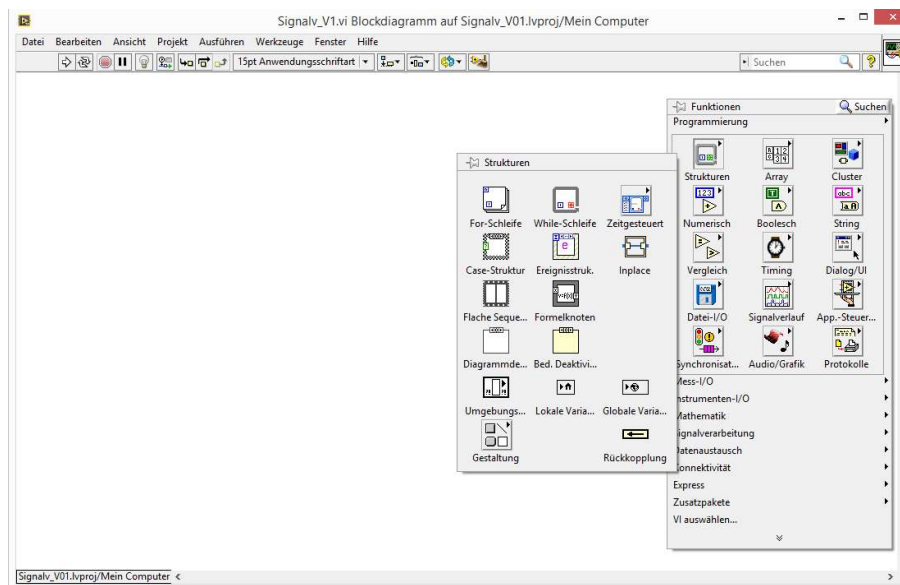


Abbildung 10: Palettenmenü mit geöffneter Unterkategorie "Strukturen"

Die Elemente sind thematisch gruppiert: Grundlegende Programmierelemente wie Kontrollstrukturen, einfache Operatoren, Konstanten, etc. findet man unter der Rubrik „Programmierung“. Spezielle Funktionen sind unter den übrigen Rubriken zu finden. Komplexe mathematische Funktionen, so z. B. aus der Trigonometrie, aus der Statistik oder auch Integraltransformationen sind unter der Rubrik „Mathematik“ zu finden. Ist die genaue Rubrik

nicht bekannt, so empfiehlt sich innerhalb des Blockdiagramms ebenfalls die Verwendung der Suchfunktion. Wie im Frontpanel, kann das gewünschte Element per „Drag & Drop“ zum Blockdiagramm hinzugefügt werden.

Sind alle benötigten Elemente im Blockdiagramm platziert worden, können die Anschlüsse „verdrahtet“ werden. Dazu führt man den Mauszeiger über den entsprechenden Anschluss, dieser ändert daraufhin selbstständig sein Design. Durch Klicken und festhalten der linken Maustaste kann der gewählte Anschluss nun mit dem Zielanschluss verbunden werden. Sofern sämtliche Anschlüsse verbunden wurden, kann das VI über den Button „Ausführen“ (vgl. Tabelle 2) gestartet werden.

4.2 Darstellung der Datentypen im Blockdiagramm

Grundsätzlich unterscheidet LabVIEW die folgenden Datentypen [1]:

- | | | |
|--------------|-------------|-----------------|
| - Numerische | - Boolesche | - Ring & Enum |
| - Array | - Pfad | - Signalverlauf |
| - String | - Referenz | - Cluster |

Die verwendeten Datentypen sind im Blockdiagramm anhand von Farbe und Beschaffenheit der Leitungen sowie der übrigen Objekte zu erkennen. In Abbildung 1 sind sämtliche abgebildeten Objekte orangefarben, was auf den Datentyp „Numerisch“ in Gleitkommadarstellung hinweist. Abbildung 11 zeigt, wie die verschiedenen Datentypen in LabVIEW dargestellt werden. Die numerischen Datentypen sind dabei in Abhängigkeit von der Zahlendarstellung nochmals unterteilt. Eine Übersicht zu den verschiedenen Untertypen des Datentyps „Numerisch“ inkl. Darstellungsbereich ist im Anhang zu finden.

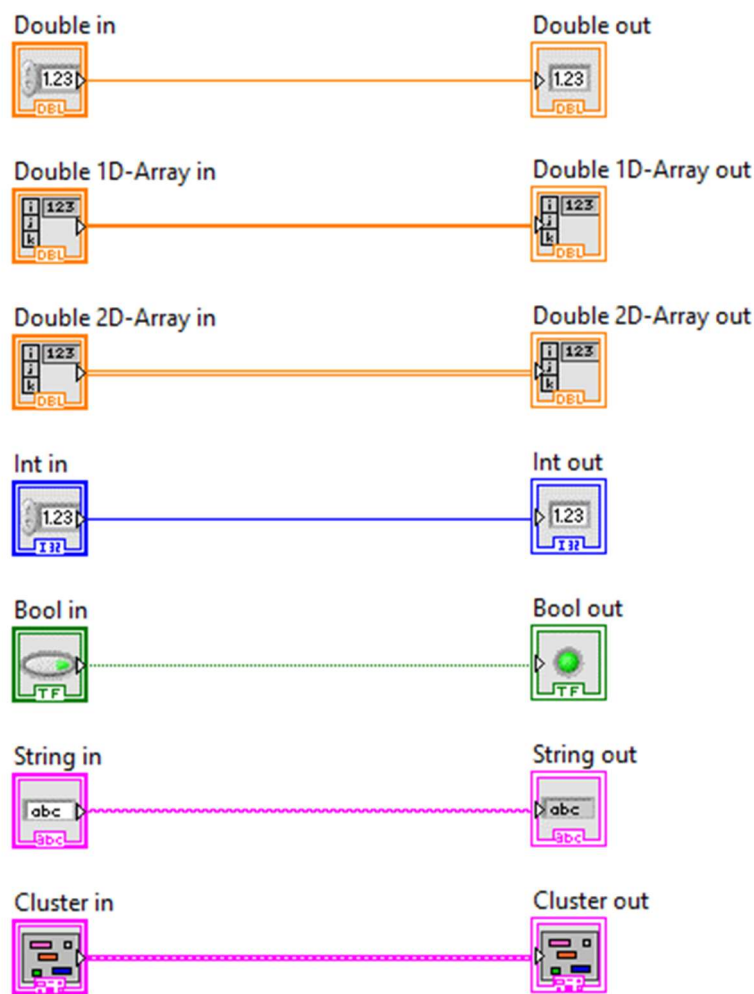


Abbildung 11: Darstellung ausgewählter Datentypen in LabVIEW

Hinsichtlich der Verarbeitung unterschiedlicher Datentypen sei an dieser Stelle noch angemerkt, dass viele Funktionen in LabVIEW polymorph sind, sich also selbstständig an den zu verarbeitenden Datentyp anpassen können. So können die Vergleichsoperatoren (z. B. Prüfung auf Gleichheit) gleichermaßen sowohl numerische Daten als auch Zeichenketten oder weitere Datentypen miteinander vergleichen. Auf der anderen Seite existieren auch Funktionen, die zwar keine Polymorphie aufweisen, jedoch eine automatische Typenumwandlung (engl. „coercion“) in einen geeigneten Datentyp durchführen [2]. Findet an einem Knoten eine automatische Datenkonvertierung statt, so wird das entsprechende Terminal („Anschlussklemme“) im Blockdiagramm mit einem roten Dreieck gekennzeichnet. Daneben existieren auch Funktionen, die eine manuelle Datenkonvertierung ermöglichen. Sollte ein Datentyp nicht mit einer Funktion oder einem Ausgabefeld kompatibel sein, so wird die entsprechende Leitung in schwarz dargestellt und mit einem Fehlersymbol (rotes „X“) markiert. Zeigt man mit der Maus auf das Fehlersymbol, so wird eine Meldung mit Hinweis auf den zulässigen Datentyp

angezeigt. Eine entsprechende Meldung erscheint auch beim Versuch, das VI mit der fehlerhaften Verbindung auszuführen. Sämtliche Datentypen können gleichermaßen als „Skalar“ oder als n -dimensionales Array vorkommen. Viele Funktionen können sowohl auf eine einfache Variable als auch auf n -dimensionale Arrays angewendet werden.

4.3 Die Menüleiste des Blockdiagramms

Analog zum Frontpanel ist auch im oberen Bereich des Blockdiagramm-Fensters eine Menüleiste zu finden. Abbildung 12 zeigt die Menüleiste des Blockdiagramms. Ein Teil der Schaltflächen ist identisch mit jenen im Blockdiagramm, die Erläuterungen zu diesen Befehlen sind in Tabelle 2 zusammengefasst, die Bedeutung der übrigen Buttons ist in Tabelle 3 erläutert.

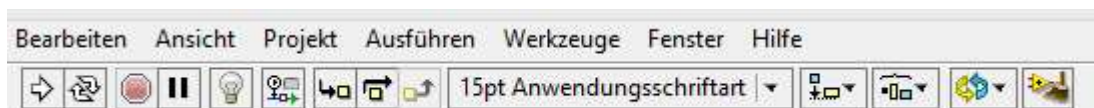


Abbildung 12: Blockdiagramm, Menüleiste



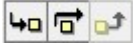

	„Highlight-Funktion“: Bei Aktivierung wird der Datenfluss zwischen den einzelnen Knoten graphisch dargestellt, so dass die Ausführung des VIs im Blockdiagramm nachvollzogen werden kann. Es ist jedoch anzumerken, dass das VI in diesem Fall verzögert ausgeführt wird, so dass z. B. die Abarbeitung von Iterationsschritten in Schleifen deutlich mehr Zeit in Anspruch nehmen kann.
	„Verbindungswerte speichern“: Speichert die über die Leitungen übertragenen Daten, so dass diese z. B. mithilfe der „Sonden“-Funktion überprüft werden können.
	„Hineinspringen“, „Überspringen“ und „Herausspringen“ dienen zur schrittweisen Ausführung von VIs.
	„Aufräumen“: Die ausgewählten Elemente werden von LabVIEW automatisch angeordnet, Verbindungen zwischen den Elementen werden ebenfalls angepasst. Kann ggf. die Übersichtlichkeit verbessern. Sofern nichts ausgewählt wird, führt LabVIEW die Funktion bzgl. des gesamten Blockdiagramms aus.

Tabelle 3: Blockdiagramm, Menüleiste, Erläuterungen zu den Symbolen

5. Hinweise zur Programmierung mit LabVIEW

5.1 Kontrollstrukturen [2]

Die aus den „klassischen“ Programmiersprachen bekannten Kontrollstrukturen sind auch in LabVIEW verfügbar. Bedingt durch das graphische Programmierkonzept, weicht die Handhabung dieser Strukturen stark von den textbasierten Programmiersprachen ab. Im Folgenden soll kurz auf die wichtigsten Kontrollstrukturen in LabVIEW sowie auf eventuelle Besonderheiten eingegangen werden. Eine Gemeinsamkeit aller hier vorgestellten Kontrollstrukturen ist, dass sie auf dem gleichen Konzept basieren: Die Kontrollstruktur besteht aus einem Rahmen, der um diejenigen Elemente herum platziert wird, deren Ausführung durch die Kontrollstruktur beeinflusst werden soll.

5.1.1 CASE-Anweisung („Verzweigung“)

Die „CASE“-Anweisung ermöglicht eine Fallunterscheidung. Das dazugehörige graphische Element besteht im Wesentlichen aus einem Rahmen mit einem „Selector Terminal“ und einem „Case Selector Label“. Das „Selector Terminal“ ist auf der linken Seite des CASE-Fensters positioniert. Der am „Selector Terminal“ anliegende Wert bestimmt, welcher Fall der CASE-Struktur ausgeführt werden soll. Das „Selector Terminal“ akzeptiert Daten vom Typ „Integer“, „Real“, „Boolean“, „String“ oder „Enum“. Die auszuführenden Elemente werden innerhalb des Rahmens platziert, der den jeweiligen Fall repräsentiert.

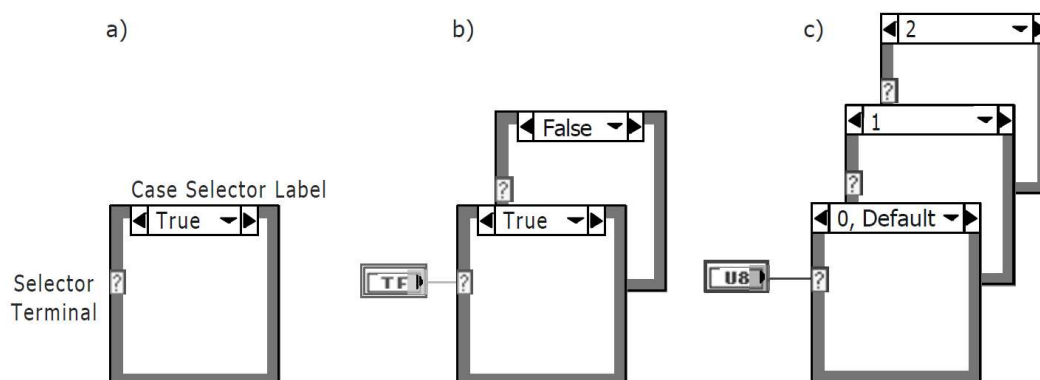


Abbildung 13: Konzept der CASE-Anweisung in LabVIEW [2]

In Abbildung 13 ist das Konzept der CASE-Anweisung dargestellt. Teil a) zeigt den Rahmen unmittelbar nach der Platzierung. Teil b) zeigt eine solche Anweisung unter

Verwendung einer Bool'schen Variablen am „Selector Terminal“, im Teil c) wurde eine Variable des Typs „Integer“ verwendet. Für jeden Fall der CASE-Struktur existiert ein separates Fenster, das in Abhängigkeit des am „Selector Terminal“ anliegenden Werts ausgeführt wird. Im Blockdiagramm wird, entgegen der prinzipiellen Darstellung in Abbildung 13, lediglich ein Fall angezeigt. Ein Durchschalten der vorhandenen Fälle kann mithilfe der Pfeilsymbole links und rechts des „Case Selector Label“ erfolgen. Die mithilfe einer Bool'schen Variablen gesteuerte CASE-Anweisungen besitzen zwei mögliche Fälle und entsprechen somit der als „IF-Anweisung“ bezeichneten Kontrollstruktur in C/C++. Wird eine numerische Variable abgefragt, so stellt LabVIEW bis zu $2^{15}-1$, also insgesamt 32767 Fälle zur Verfügung, wobei zunächst nur zwei Fälle generiert werden. Weitere Fälle können durch Rechtsklicken auf das „Case Selector Label“ und Auswahl des entsprechenden Menüpunkts neu hinzugefügt, aus einem bereits vorhandenen Fall kopiert oder bei Bedarf auch gelöscht werden. Die Datenübergabe zwischen dem CASE-Fenster und dem übrigen Blockdiagramm erfolgt mithilfe von „Tunnel“ genannter Knotenpunkte, die bei Verbindung zweier, jeweils innerhalb sowie außerhalb der Struktur befindlichen Terminals selbstständig angelegt werden.

Abbildung 14 zeigt ein einfaches Beispiel einer CASE-Anweisung: Zwei Zahlen sollen wahlweise addiert oder subtrahiert werden. Die Abfrage erfolgt über eine Bool'sche Variable. Links ist der Fall „TRUE“ (Addition), rechts der Fall „FALSE“ (Subtraktion) ausgewählt. Unmittelbar auf dem Rahmen sind die „Tunnel“ (hier orangefarben) und das „Selector Terminal“ (hier grün) zu erkennen.

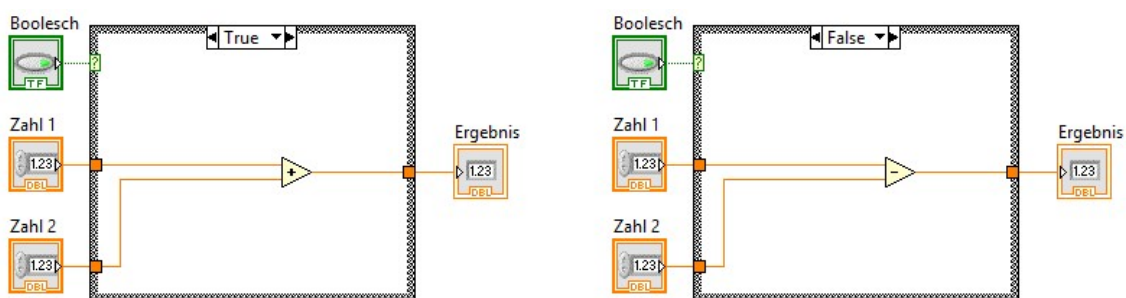


Abbildung 14: CASE-Anweisung, Anwendungsbeispiel

5.1.2 FOR-Schleife

Die FOR-Schleife dient dazu, einen Programmabschnitt mit einer definierten Anzahl an Wiederholungen auszuführen. Im Blockdiagramm wird die FOR-Schleife als Rahmen dargestellt und weist zwei Terminals auf, ein mit „N“ bezeichnetes Zählterminal sowie

ein Iterationsterminal („ i “). Der Wert N wird aus dem übrigen Blockdiagramm übergeben und legt die Anzahl der Wiederholungen („Iterationsschritte“) fest, nach denen die Schleife verlassen werden soll. Der Wert i gibt den aktuellen Iterationsschritt wieder. Bei Vorgabe von $N=0$ wird die Schleife nicht ausgeführt. Für beide Terminals wird der Datentyp „Integer32“ verwendet, bei Übergabe anderer Datentypen erfolgt, sofern möglich, eine automatische Datentypkonvertierung. Das Beispiel in Abbildung 15 soll das Konzept der FOR-Schleife anhand eines einfachen Beispiels erläutern. Es wird ein Array des Typs „Integer32“ erstellt, wobei alle Iterationsindizes i von 0 bis $N-1$ als Zahlenfolge in das Array geschrieben werden. Man erkennt ferner eine Eigenschaft der FOR-Schleife: In den einzelnen Iterationsschritten werden „skalare“ Variablen verarbeitet, über den Tunnel mit Array-Indizierung (rote Markierung in Abbildung 15) wird allerdings ein Array aus „Integer32“-Elementen an das Ausgabefeld „Zahlenfolge“ übergeben.

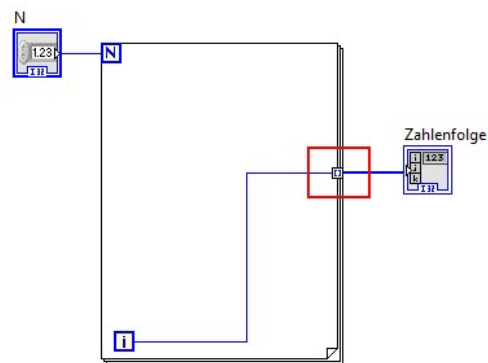


Abbildung 15: FOR-Schleife, Anwendungsbeispiel

5.1.3 WHILE-Schleife

Die WHILE-Schleife wird, ähnlich wie die FOR-Schleife, dazu verwendet, Anweisungen wiederholt auszuführen. Der Unterschied zur FOR-Schleife besteht darin, dass die Anzahl der Iterationsschritte nicht im Vorfeld definiert wird, es wird vielmehr eine logische Bedingung als Abbruchkriterium verwendet.

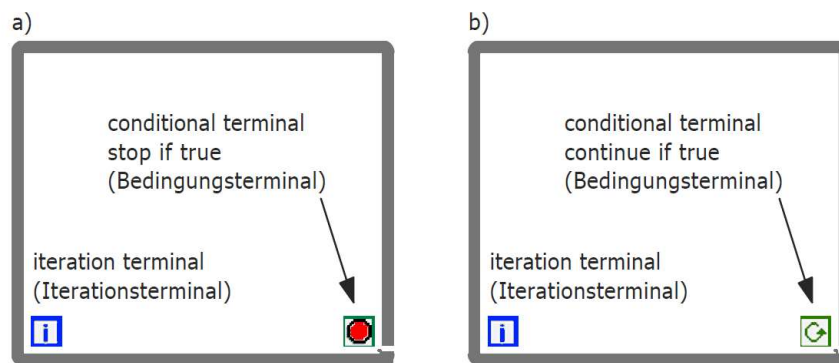


Abbildung 16: WHILE-Schleife in LabVIEW (vgl. [2])

Eine WHILE-Schleife besteht im Blockdiagramm aus einem Rahmen, der ein Iterationsterminal und ein Bedingungsterminal enthält. Das Bedingungsterminal ist ein Boole'sches Element, das mit dem Wert TRUE aktiviert wird. Es kann wahlweise als Abbruch- (engl. „Stop if True“, Abbildung 16a) oder als Laufbedingung („Run if True“, Abbildung 16b) dienen, wobei die Umschaltung des Terminals durch einfaches Anklicken desselben erfolgt.

5.1.4 Sequenz

Bei textbasierten Programmiersprachen ist die Reihenfolge der Bearbeitung durch die zeilenweise Anordnung des Quellcodes sichergestellt. In LabVIEW ist dies aufgrund des Datenflusskonzepts nicht sichergestellt, da Befehle prinzipiell immer dann ausgeführt werden, wenn Signale an den Eingangsterminals anliegen. Sofern Betriebssystem und Hardware dies zulassen, ist grundsätzlich auch ein paralleler Ablauf nicht ausgeschlossen. Hinsichtlich der Abarbeitung kann ein definierter Ablauf jedoch durch Einfügen einer Sequenzstruktur erreicht werden.

Im Blockdiagramm besteht eine Sequenz aus einem abschnittsweise unterteilten Rahmen, der vom Erscheinungsbild einem Filmstreifen nachempfunden sind. Es kann zwischen einer flachen und einer gestapelten Sequenz (engl. „flat/stacked sequence“) ausgewählt werden. Beiden Varianten sind gleichwertig und unterscheiden sich lediglich in der Anordnung der einzelnen Rahmen (engl. „Frames“).

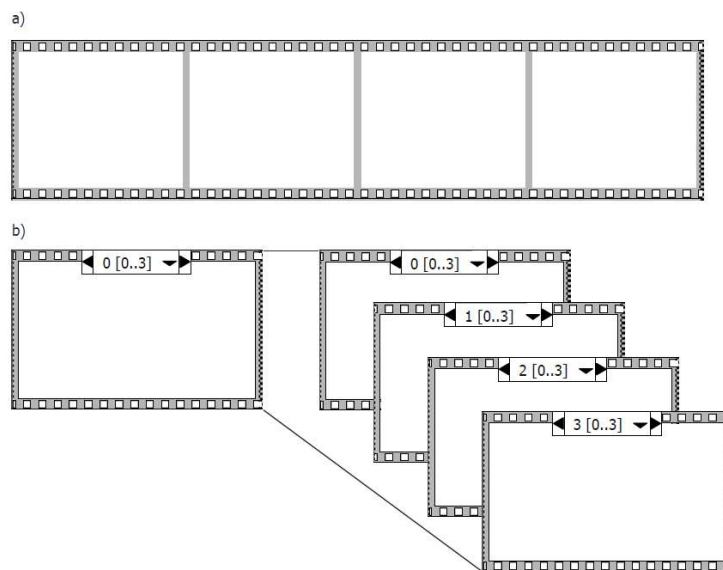


Abbildung 17: Darstellung einer Sequenz in LabVIEW; a) flache Sequenz b) gestapelte Sequenz [2]

Wird der Rahmen dem Blockdiagramm hinzugefügt, so erscheint zunächst ein einzelner Frame. Weitere Frames können durch Rechtsklick auf den Rahmen und Auswahl der entsprechenden Option hinzugefügt werden. Daten werden über Tunnel hinein- und herausgeführt, wobei anzumerken ist, dass herausgeführte Daten für das übrige Blockdiagramm erst in dem Moment zur Verfügung stehen, wenn sämtliche Frames der Sequenz abgearbeitet wurden.

5.2 SubVI

In den „klassischen“ Programmiersprachen ist der Prozeduraufruf ein Mittel, um komplexe Problemstellungen zu strukturieren. LabVIEW stellt eine derartige Funktionalität ebenfalls zur Verfügung, wobei Unterprogramme in diesem Zusammenhang als „SubVI“ bezeichnet werden. In LabVIEW kann aus jedem VI ein SubVI erstellt werden und beliebig aufgerufen werden. Die Vorgehensweise hierzu soll im Folgenden anhand eines Beispiels erläutert werden.

Zunächst einmal ist ein „gewöhnliches“ VI zu erstellen. Unser SubVI soll es ermöglichen, zwei beliebige Zahlen des Typs „Double Precision“ wahlweise zu addieren oder zu subtrahieren, wobei die auszuführende Operation mithilfe einer Bool'schen Variablen auszuwählen ist. Frontpanel und Blockdiagramm des erstellten VI sind in Abbildung 18 dargestellt.

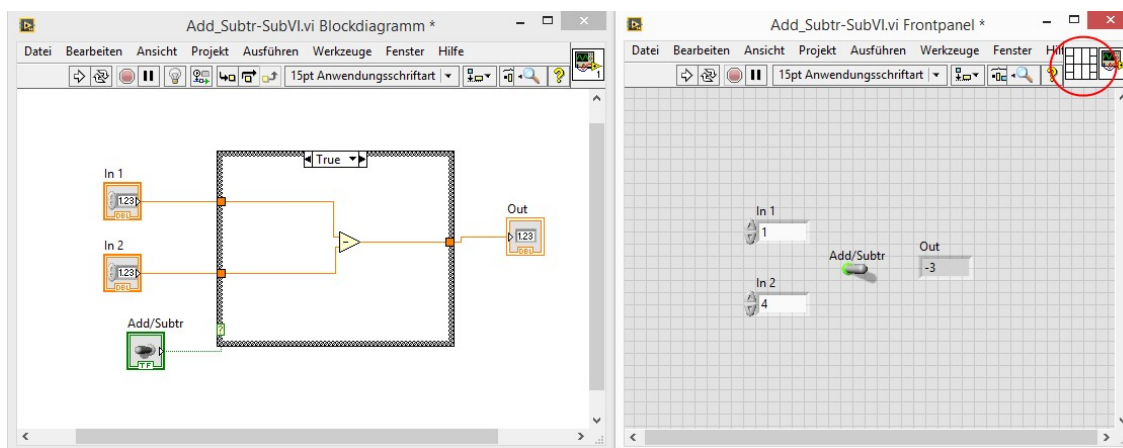


Abbildung 18: VI zur Addition/Subtraktion zweier Zahlen

Da ein erstelltes SubVI im Blockdiagramm als Knoten mit Ein- und Ausgangsterminals erscheint, muss zunächst festgelegt werden, an welcher Stelle die entsprechenden Terminals platziert werden sollen. Hierzu wird das Anschlussfeld „engl. Connector Pane“- (siehe rote Markierung in Abbildung 18) verwendet. LabVIEW ermöglicht prinzipiell bis zu 28 Terminals pro Block. Im Sinne eines übersichtlichen Blockdiagramms sollte die Anzahl jedoch auf ein Minimum begrenzt werden. Die Anzahl der Terminals richtet sich nach den benötigten Ein- und Ausgabeparameter. In unserem Fall werden drei Eingabe-Terminals (*In 1*, *In 2*, *Add/Subtr*) sowie ein Ausgabe-Terminal (*Out*) benötigt. Durch Klicken mit der rechten Maustaste (vgl. Abbildung 19) auf den „Connector Pane“ und Wahl des Menüpunkts „**Muster**“ kann eine für das zu erstellende SubVI geeignete Anzahl und Anordnung der Terminals festgelegt werden. Als Standard bietet LabVIEW ein Anschlussfeld mit zwölf Terminals an.

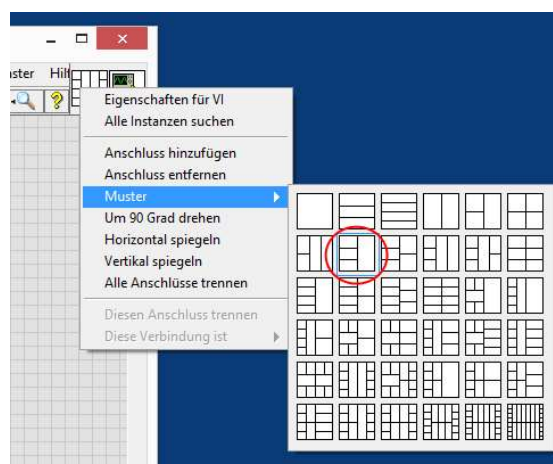


Abbildung 19: Konfiguration der Terminals eines SubVI

Das hier gewählte Anschlussmuster, das über insgesamt 4 Terminals verfügt, wird nun im Anschlussfeld angezeigt. Jedem der vier Terminals kann nun ein beliebiger Parameter

zugeordnet werden, wobei es sich aufgrund des Datenflusskonzepts empfiehlt, die Terminals auf der linken Seite (vgl. Terminal 1...3, Abbildung 20) als Eingangs- und die Terminals auf der rechten Seite (vgl. Terminal 4, Abbildung 20) als Ausgangsparameter des SubVI zu nutzen.

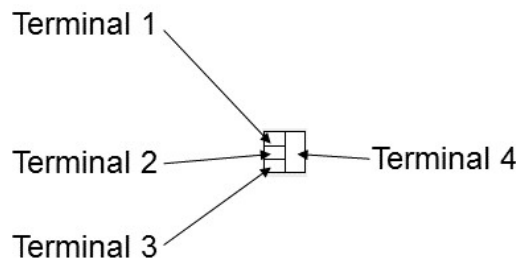


Abbildung 20: Darstellung der Terminals im "Connector Pane"

Im Folgenden sind die Terminals des Anschlussfeldes mit den entsprechenden Bedien- und Anzeigeelementen im Frontpanel zu verknüpfen. Die Zuordnung erfolgt dadurch, dass das gewünschte Terminal im Anschlussfeld und das jeweilige Anzeigeelement mit dem „Verbinden“-Werkzeug nacheinander angeklickt werden. Sofern die Zuordnung erfolgreich war, werden die zuvor weißen Segmente nun in der Farbe des jeweiligen Datentyps (siehe Abbildung 21, rechts oben) angezeigt. Eine nachträgliche Bearbeitung der Belegung ist ebenfalls möglich. Des Weiteren kann ein Symbol erstellt werden, mit dem das SubVI im Blockdiagramm gekennzeichnet wird.

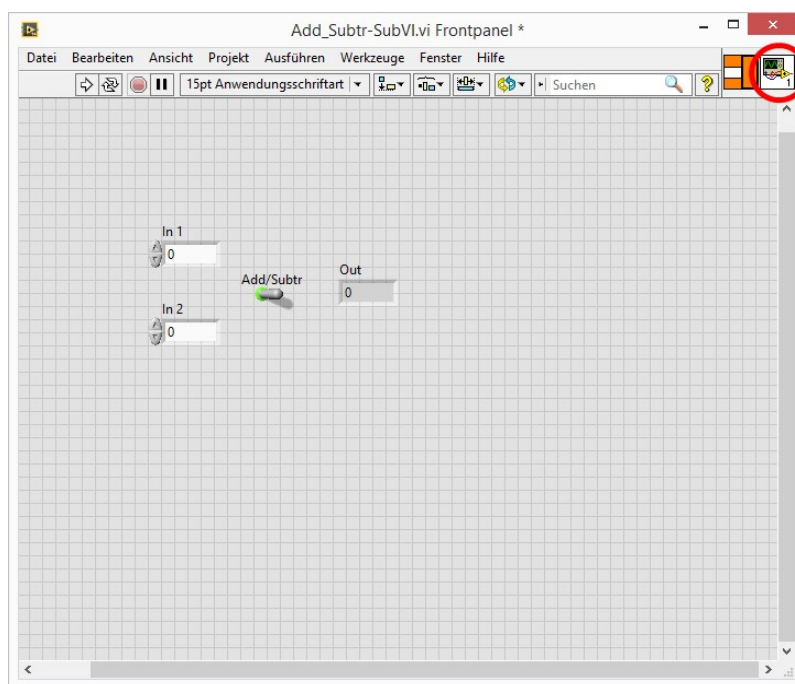


Abbildung 21: Schaltfläche zum Öffnen des Symbol-Editors

Das Symbol kann mithilfe des „Symbol-Editors“, der durch einen Doppelklick auf die entsprechende Schaltfläche (Abbildung 21, rote Markierung) bearbeitet werden. Die Benutzeroberfläche des Editors, auf die an dieser Stelle nicht näher eingegangen werden soll, entspricht einem rudimentären Bildbearbeitungsprogramm und ist in Abbildung 22 dargestellt.

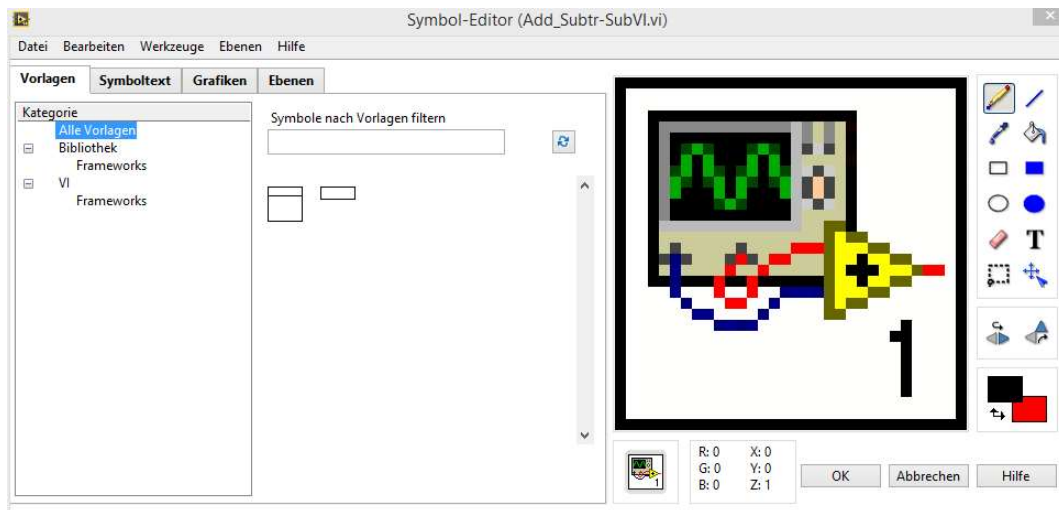









Abbildung 22: Symbol-Editor, Benutzeroberfläche






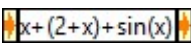

Es können vordefinierte Symbole eingefügt und bearbeitet werden, daneben besteht die Möglichkeit, ein benutzerdefiniertes Symbol neu zu erstellen.




Sofern die Erstellung abgeschlossen ist, kann das SubVI abgespeichert werden und kann über das Palettenmenü durch Auswahl des Menüpunkts „**VI Auswählen...**“ in das Blockdiagramm eingefügt werden.

6. Einige grundlegende LabVIEW-Funktionen

Im Folgenden sollen einige grundlegende LabVIEW-Funktionen vorgestellt werden. Aufgrund des sehr großen Funktionsumfang von LabVIEW kann an dieser Stelle nur eine kleine Auswahl an Funktionen ohne Anspruch auf Vollständigkeit besprochen werden. Die Belegung der Ein- und Ausgangsterminals, die jeweils verwendeten Datentypen sowie eine kurze Beschreibung der jeweiligen Funktion kann durch Aufrufen der Hilfefunktion in LabVIEW angezeigt werden. Hierzu ruft man das Kontextmenü durch Klicken mit der rechten Maustaste auf das Symbol der jeweiligen Funktion im Blockdiagramm auf und wählt dann den Menüpunkt „Hilfe“.

Symbol	Bezeichnung	Rubrik	Kurzbeschreibung
	Array erstellen	Array	Verbindet mehrere Arrays oder fügt Elemente an ein n -dimensionales Array an. Wenn Sie die Funktion ins Blockdiagramm einfügen, hat sie nur einen Eingang. Zum Hinzufügen von Eingängen klicken Sie den Eingang an und wählen Sie aus dem Kontextmenü die Option „Eingang hinzufügen“ aus oder ziehen Sie die Funktion mit der Maus auf.
	Array indizieren	Array	Gibt ein Element oder ein Teil-Array in Abhängigkeit des Eingangs „ n -dimensionales Array“ (Index) aus.
	aus Array entfernen	Array	Löscht, beginnend mit der Variablen „Index“, ein Element oder Teil-Array mit der angegebenen Länge. Das Ergebnis dieser Operation wird unter „Array (Ausgang)“ ausgegeben, der gelöschte Teil im Array „Gelöschter Teil“.
	Array-Größe	Array	Gibt die Anzahl der Elemente in jeder Dimension aus.
	in Array einfügen	Array	Fügt in ein n -dimensionales Array an der durch „Index“ angegebenen Stelle ein weiteres Element oder ein weiteres Teil-Array ein.
	Array initialisieren	Array	Erzeugt ein n -dimensionales Array, in dem jedes Element auf den Wert des Eingangs „Element“ initialisiert wird.
	Bündeln	Cluster	Verbindet einzelne Elemente unterschiedlichen Datentyps zu einem Cluster.

	Aufschlüsseln	Cluster	Schlüsselt einen Cluster in seine einzelnen Komponenten auf.
	Addieren	numerisch	Berechnet die Summe der Eingangswerte.
	Subtrahieren	numerisch	Berechnet die Differenz der Eingangswerte.
	Multiplizieren	numerisch	Gibt das Produkt der Eingangswerte aus.
	Dividieren	numerisch	Berechnet den Quotienten der Eingangswerte.
	Inkrementieren	numerisch	Addiert 1 zum Eingangswert.
	Dekrementieren	numerisch	Subtrahiert 1 vom Eingangswert.
	Absoluter Wert	numerisch	Gibt den Betrag des Eingangswerts aus.
	Ausdrucks-knoten	numerisch	Der Ausdrucks-knoten dient zur Berechnung von Ausdrücken mit einer Variablen. Hinweise auf mögliche Funktionen sind in der LabVIEW-Hilfe zu finden.
	UND	Boolesch	Verknüpft die Eingangsgrößen durch ein logisches UND. Die Eingangsgrößen müssen beide boolesche Werte, numerische Werte oder Fehler-Cluster sein.
	ODER	Boolesch	Wendet ein logisches ODER auf die Eingänge an. Die Eingangsgrößen müssen beide boolesche Werte, numerische Werte oder Fehler-Cluster sein.
	GLEICH?	Vergleich	Gibt TRUE aus, wenn Eingang „x“ gleich Eingang „y“ ist. Ansonsten wird FALSE ausgegeben.
	Sinus	Mathematik/ Funktionen/ Trigonometrie	Berechnet den Sinus von x im Bogenmaß.

	Cosinus	Mathematik/ Funktionen/ Trigonom.	Berechnet den Cosinus von x im Bogenmaß.
	1D-interpolieren	Mathematik/ interp.	Führt eine eindimensionale Interpolation mithilfe einer ausgewählten Methode basierend auf der durch die Eingänge „ X “ und „ Y “ definierten Zuordnungstabelle durch.
	Median	Mathematik/ Statistik	Bestimmt den Median der Eingangsfolge X durch Sortieren der Werte von X und Mittelwertbildung der mittleren Elemente des sortierten Arrays.

7. Vorbereitung und Aufgabenstellung

Zur Vorbereitung des Praktikums ist diese Kurzeinführung in die LabVIEW-Programmierung aufmerksam zu lesen.

Die Aufgabenstellungen werden im Rahmen des Praktikums mitgeteilt.

LITERATURVERZEICHNIS

- [1] W. Georgi und E. Metin: Einführung in LabVIEW mit 157 Aufgaben, München: Fachbuchverlag Leipzig im Carl-Hanser-Verlag, 2012.
- [2] B. Mütterlein: Handbuch für die Programmierung mit LabVIEW, Heidelberg: Spektrum Akademischer Verlag, 2007.
- [3] National Instruments Corp.: „Übersicht über die numerischen Datentypen in LabVIEW,“ 2015. [Online]. Available: http://zone.ni.com/reference/de-XX/help/371361H-0113/lvhowto/numeric_data_types_table/. [Zugriff am 20.08.2015].

Abkürzungsverzeichnis

LabVIEW „Laboratory Virtual Instrument Engineering Workbench“
VI „Virtual Instrument“











Abbildungsverzeichnis







Abbildung 1: Benutzeroberfläche der LabVIEW-Entwicklungsumgebung	4
Abbildung 2: LabVIEW-Symbol	5
Abbildung 3: LabVIEW, Startbildschirm	5
Abbildung 4: Auswahlmenü "Projekt erstellen"	6
Abbildung 5: Projekt-Explorer	6
Abbildung 6: Neues VI hinzufügen, Dialog	7
Abbildung 7: Dialog zum Speichern des VIs	7
Abbildung 8: Palettenmenü mit geöffneter Unterkategorie "Numerisch"	8
Abbildung 9: Frontpanel, Menüleiste	10
Abbildung 10: Palettenmenü mit geöffneter Unterkategorie "Strukturen"	12
Abbildung 11: Darstellung ausgewählter Datentypen in LabVIEW	14
Abbildung 12: Blockdiagramm, Menüleiste	15
Abbildung 13: Konzept der CASE-Anweisung in LabVIEW [2]	16
Abbildung 14: CASE-Anweisung, Anwendungsbeispiel	17
Abbildung 15: FOR-Schleife, Anwendungsbeispiel	18
Abbildung 16: WHILE-Schleife in LabVIEW (vgl. [2])	19
Abbildung 17: Darstellung einer Sequenz in LabVIEW; a) flache Sequenz b) gestapelte Sequenz [2]	20
Abbildung 18: VI zur Addition/Subtraktion zweier Zahlen	21
Abbildung 19: Konfiguration der Terminals eines SubVI	21
Abbildung 20: Darstellung der Terminals im "Connector Pane"	22
Abbildung 21: Schaltfläche zum Öffnen des Symbol-Editors	22
Abbildung 22: Symbol-Editor, Benutzeroberfläche	23

Tabellenverzeichnis

Tabelle 1: Bedienelemente Frontpanel, Auswahl	10
Tabelle 2: Frontpanel, Menüleiste, Erläuterungen zu den Symbolen	11
Tabelle 3: Blockdiagramm, Menüleiste, Erläuterungen zu den Symbolen.....	15

Übersicht über die numerischen Datentypen in LabVIEW [3]

Anschluss	Numerischer Datentyp	Belegter Speicherplatz in Bits	Ungefähre Anzahl an Dezimalstellen	Bereich (ungefähr)
	Fließkommazahl, einfache Genauigkeit	32	6	Kleinste positive Zahl: $1,40 \cdot 10^{45}$ Größte positive Zahl: $3,40 \cdot 10^{38}$ Kleinste negative Zahl: $-1,40 \cdot 10^{45}$ Größte negative Zahl: $-3,40 \cdot 10^{38}$
	Fließkommazahl, doppelte Genauigkeit	64	15	Kleinste positive Zahl: $4,94 \cdot 10^{324}$ Größte positive Zahl: $1,79 \cdot 10^{308}$ Kleinste negative Zahl: $-4,94 \cdot 10^{324}$ Größte negative Zahl: $-1,79 \cdot 10^{308}$
	Fließkommazahl, erweiterte Genauigkeit	128	variiert je nach Plattform von 15 bis 20	Kleinste positive Zahl: $6,48 \cdot 10^{-4966}$ Größte positive Zahl: $1,19 \cdot 10^{4932}$ Kleinste negative Zahl: $-6,48 \cdot 10^{4966}$ Größte negative Zahl: $-1,19 \cdot 10^{4932}$
	Komplexe Fließkommazahl, einfache Genauigkeit	64	6	Wie Fließkommazahl einfacher Genauigkeit für Real- und Imaginärteil
	Komplexe Fließkommazahl, doppelte Genauigkeit	128	15	Wie Fließkommazahl doppelter Genauigkeit für Real- und Imaginärteil
	Komplexe Fließkommazahl, erweiterte Genauigkeit	256	variiert je nach Plattform von 15 bis 20	Wie Fließkommazahl erweiterter Genauigkeit für Real- und Imaginärteil
	Festkomma	64 oder 72, wenn Sie einen Überlaufzustand einschließen	konfigurationsabhängig	konfigurationsabhängig
	Vorzeichenbehafteter Byte-Integer	8	2	-128 bis 127
	Vorzeichenbehafteter Word-Integer	16	4	-32.768 bis 32.767
	Vorzeichenbehafteter Long-Integer	32	9	-2.147.483.648 bis 2.147.483.647

	Vorzeichenbehafteter Quad-Integer	64	18	-1 10 ¹⁹ bis 1 10 ¹⁹
	Vorzeichenloser Byte-Integer	8	2	0 bis 255
	Vorzeichenloser Word-Integer	16	4	0 bis 65.535
	Vorzeichenloser Long-Integer	32	9	0 bis 4.294.967.295
	Vorzeichenloser Quad-Integer	64	19	0 bis 2e19
	128-Bit-Zeitstempel	128	19	Zeitbereichsanfang: 01/01/1600 00:00:00 Zeitbereichsende: 01/01/3001 00:00:00 (Weltzeit)