Entwicklung einer Software zur Strukturschwingungsanalyse von Hubschrauberwindkanalmodellen

Bachelorarbeit

im Studiengang Elektrotechnik – Automatisierung und Energiesysteme im Wintersemester 2011/2012

Verfasser:	René Pfeifer (20863159)
	Melanie Schulze (20863421)
Erstprüfer:	Prof. Dr. rer. nat. C. W. Turtur
Zweitprüfer:	Dipl.-Ing. Michael Przybilla

Ort:	Braunschweig
Abgabetermin:	09.01.2012
Bearbeitungszeit:	sechs Monate

Versicherung der selbstständigen Erarbeitung der Bachelorarbeit

Hiermit versichern wir, dass wir die Bachelorarbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und die Stellen der Bachelorarbeit, die im Wortlaut oder im wesentlichen Inhalt aus anderen Werken (auch aus dem Internet) entnommen wurden, mit genauer Quellenangabe kenntlich gemacht haben.

Braunschweig, 09. Januar 2012

.....
(Unterschrift des Studenten)

Braunschweig, 09. Januar 2012

.....
(Unterschrift der Studentin)

Da es sich bei dieser Bachelorarbeit um eine Gruppenarbeit handelt, sei an dieser Stelle kurz erläutert, welche Kapitel von René Pfeifer und welche von Melanie Schulze bearbeitet wurden.
Die Kapitel „Einleitung“, „Kurzzusammenfassung der Arbeit“, „Ablaufplan“, „Testaufbau“, „Das *TDMS*-Dateiformat“, „Messung an einem Hubschrauberwindkanalmodell“ sowie „Fazit und Ausblick“ wurden gemeinsam ausgearbeitet.
René Pfeifer befasste sich mit dem Kapitel „Das *LabVIEW*-Programm“ und Melanie Schulze mit den Kapiteln „Das *DIAdem*-Programm“ sowie „Umwandeln des Beschleunigungssignals in ein Wegsignal“.
Auch wenn die Themen für die Bachelorarbeit wie eben beschrieben aufgeteilt wurden, haben wir uns innerhalb des Praxisprojektes zusammen in das komplette Thema eingearbeitet. Entsprechend wurde auch das Kapitel „Dokumentation der Praxisphase“ gemeinsam erstellt.

Abkürzungsverzeichnis

BNC: „Bayonet Neill Concelman“

DAQ: „Data Acquisition“

DVI: „Digital Visual Interface“

FFT: „Fast Fourier Transformation“

FIR: „Finite Impulse Response“

FRF: „Frequency Response Function“

IEPE: „Integrated Electronics Piezo-Electric“

IIR: „Impulse Response Filter“

MAX: „Measurement and Automation Explorer“

NI: „National Instruments“

PCI: „Peripheral Component Interconnect“

PXI: „PCI eXtensions for Instrumentation“

RMS: „Root Mean Square“

TDMS: „Technical Data Management Streaming“

VGA: „Video Graphics Array“

VI: „Virtual Instrument“

Inhaltsverzeichnis

1. Einleitung

In diesem Kapitel soll erläutert werden, was die Aufgabe dieser Bachelorarbeit ist und wie der vorhandene Aufbau zu dieser Aufgabe aussieht.

Beim Lesen dieser Arbeit ist zu beachten, dass hochgestellte Zahlen hinter Wörtern auf das Fremdwortverzeichnis am Ende dieser Arbeit verweisen.

1.1. Aufgabenstellung

Es soll eine Software entwickelt werden, die es ermöglicht, Schwingungen auf ein Hubschrauberwindkanalmodell zu übertragen und Messdaten von Beschleunigungssensoren zu erfassen. Anschließend sollen mit Hilfe dieser Daten Resonanzfrequenzen ermittelt werden.
An Software stehen hierfür die Programme *LabVIEW* und *DIAdem* von *National Instruments* („NI“) zur Verfügung.
An Hardware sind einerseits sechs Beschleunigungssensoren sowie ein Kraftsensor, ein Shaker und verschiedene Verstärker, andererseits ein *PCI eXtensions for Instrumentation („PXI“)*-Messsystem vorhanden.

Es soll nun mittels *LabVIEW* über die Outputkarte des Messsystems ein Signal ausgegeben werden, das den Shaker anregt, der seine mechanische Bewegung wiederum auf das zu testende System überträgt.
Bei dem Signal soll es möglich sein, zwischen weißem Rauschen und einem periodischen Signal zu wählen. Bei der Resonanzfrequenzfindung wird in der Regel so vorgegangen, dass zunächst ein weißes Rauschen auf das System gegeben wird. Durch dieses lassen sich Bereiche erkennen, in denen die Resonanzen liegen. Diese Bereiche werden anschließend mit einem periodischen Signal, meist ein Sinussignal, näher untersucht.
Bei dem Rauschen soll das Root-Mean-Square-(„RMS“)-Level, der Offset sowie die maximale Frequenz im Rauschsignal einstellbar sein. Das periodische Signal soll in seiner Amplitude, Frequenz, Phase und seinem Offset veränderbar sein.
Die Ausgabe eines periodischen Signals soll automatisiert einen Sweep von einer vorher festgelegten Startfrequenz bis zu einer Endfrequenz mit einer gewünschten Schrittweite durchlaufen können.
Die Messdaten der auf dem Modell angebrachten Beschleunigungssensoren sollen über die Input-Karten des *PXI*-Messsystems eingelesen und mit Hilfe von *LabVIEW* verarbeitet und in eine Datei geschrieben werden, welche mit *DIAdem* ausgewertet wird. Die Auswertung der Daten soll schließlich noch anschaulich dargestellt und gespeichert werden.

1.2. Aufbau des Systems

Der Aufbau, der zur Bearbeitung der Aufgabe vorhanden ist, ist in Abbildung 1.1 schematisch dargestellt.

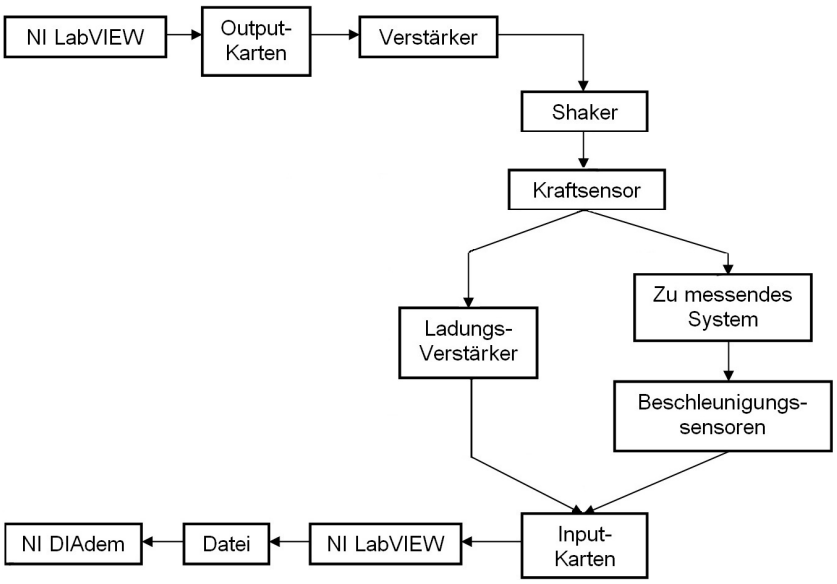


Abbildung 1.1.: Schematischer Aufbau des Systems, Autorin: Melanie Schulze

Genaueres zu den einzelnen Komponenten dieses Aufbaus wird in dem Kapitel „Dokumentation der Praxisphase“ (siehe Kapitel 3) erläutert.

2. Kurzzusammenfassung der Arbeit

Diese Bachelorarbeit hat zum Ziel, eine Software zu entwickeln, mit der Resonanzfrequenzen an Hubschrauberwindkanalmodellen gefunden werden können. Dieses hat folgende Motivation: Einerseits müssen die Resonanzfrequenzen gefunden werden, damit Bodenfrequenzen umgangen werden können, denn diese haben zur Folge, dass das System instabil werden kann (näheres ist hierzu in [6], Kapitel 7.6.4 zu finden). Gerät man andererseits mit der Rotordrehzahl in eine Resonanzfrequenz, wird das System zwar nicht instabil, aber es entstehen sehr hohe Materialbelastungen, wodurch im schlimmsten Fall das Modell zerstört werden kann. Aus diesem Grund muss auch dieser Fall vermieden werden.

Zur Bearbeitung dieser Aufgabe werden die Software *LabVIEW* von *NI* sowie die Software *DIAdem* von *NI* eingesetzt. Eine Einführung in diese beiden Programme ist in den Abschnitten 3.2.2 und 3.2.3 zu finden.
Mit *LabVIEW* wird der gesamte Messvorgang realisiert. Das heißt, dass mit *LabVIEW* einerseits der Shaker über die Output-Karte des *PXI*-Messsystems (siehe Abschnitt 3.1.2) angesteuert wird. Dabei müssen die Anforderungen aus Abschnitt 1.1 beachtet und umgesetzt werden. Andererseits werden die Messwerte der Sensoren mittels *LabVIEW* über die Input-Karten des *PXI*-Messsystems eingelesen. Die Erläuterung des *LabVIEW*-Programms ist in Kapitel 7 zu finden.
DIAdem wird nun dazu verwendet, die gesamte Auswertung auszuführen. Da Resonanzen gefunden werden sollen, werden mit *DIAdem* Fast Fourier Transformations („FFT“s) bzw. Frequency Response Functions („FRF“s) berechnet. Des Weiteren sollen die berechneten Ergebnisse mit Hilfe von *DIAdem* für den Anwender anschaulich dargestellt und übersichtlich gespeichert werden. Das *DIAdem*-Programm ist in Kapitel 8 erklärt.

Da die Messdaten in *LabVIEW* eingelesen und mit *DIAdem* ausgewertet werden, müssen die Daten von *LabVIEW* an *DIAdem* übermittelt werden. Die Schnittstelle hierfür ist eine sogenannte Technical Data Management Streaming („TDMS“)-Datei (siehe Kapitel 6). Es ist möglich, in *LabVIEW* in eine solche Datei zu schreiben und in *DIAdem* eine solche Datei zu öffnen und zu bearbeiten.

Die Programme werden zunächst an einem Testaufbau (siehe Kapitel 5) erprobt. Dabei handelt es sich hier um ein Feder-Masse-Pendel. Dieses wird gewählt, da es bei diesem einfachen Aufbau möglich ist, die Resonanzfrequenzen vorher zu berechnen und somit die Möglichkeit besteht, die Funktion des Programms zu überprüfen. Dieses wäre mit einem komplexen Hubschrauberwindkanalmodell nicht möglich.
Nach vielen Tests werden die Programme schlussendlich auch an einem realen Hubschrauberwindkanalmodell ausgeführt (siehe Kapitel 10).

3. Dokumentation der Praxisphase

Der Titel unserer Praxisphase lautet: „Einarbeitung in die Software „NI LabVIEW“ und „NI DIAdem“ inklusive der Programmiersprache „Visual Basic Script“ zur Nutzung eines angepassten *PXI*-Messsystems“.

Hier soll auf die Einarbeitung in die Hard- und Software sowie dabei gewonnene Erkenntnisse eingegangen werden.

3.1. Einarbeitung in die Hardware

In diesem Kapitel sollen zunächst Erläuterungen erfolgen, die sich auf die vorhandene Hardware beziehen.

3.1.1. Fertigung von Hardware-Komponenten

An dem in Abbildung 1.1 dargestellten Aufbau fehlen noch einige kleine Hardware-Komponenten, die im Rahmen dieses Praxisprojektes fertiggestellt werden.

Da der Aufbau mobil sein muss, befinden sich die Verstärker (Leistungsverstärker und Ladungsverstärker), das *PXI*-Messsystem sowie Peripheriegeräte in einem Rollcontainer (siehe Abbildung 3.1).

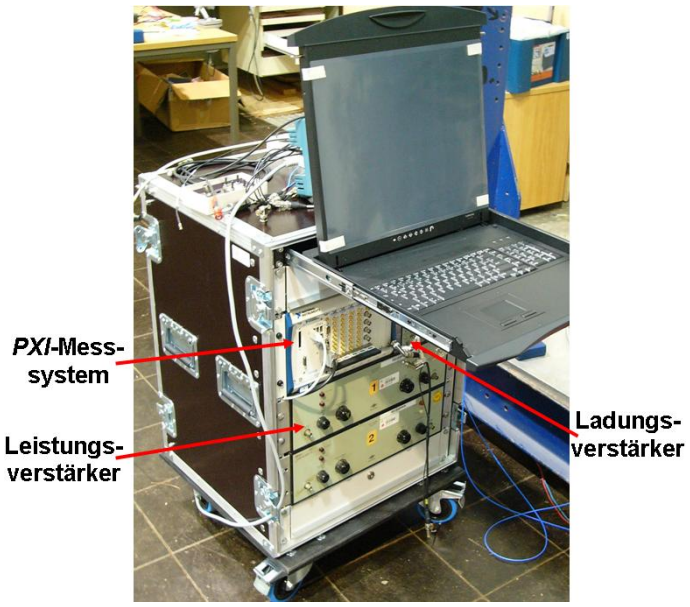


Abbildung 3.1.: Aufbau des Rollcontainers, Autor: René Pfeifer

Die Output-Kanäle der Messkarte „PXI-4461“ werden über *Bayonet Neill Concelman* („BNC“)-Kabel mit den Leistungsverstärkern verbunden. Genauso verhält es sich mit zwei Eingangskanälen der letzten Messkarte „PXI-4472B“ und den Ladungsverstärkern. Diese Kabel verlaufen momentan noch außerhalb des Rollcontainers. Da dieser aber zum Transport verschlossen werden muss, müssen die Kabel innerhalb des Containers verlegt werden.
Hierfür wird ein Loch in die vorhandene Metallplatte gebohrt (siehe Abbildung 3.2).

Abbildung 3.2.: Metallplatte mit Loch für die Kabel, Autor: René Pfeifer

Des Weiteren wird ein Adapter von *Video Graphics Array* („VGA“) auf *Digital Visual Interface* („DVI“) benötigt. Ein „VGA-Anschluss ist ein analoger Abbildungsübertragungsstandard für Stecker- und Kabelverbindungen zwischen Grafikkarten und Anzeigegeräten“ (aus [28]). Mit der elektronischen Schnittstelle DVI werden ebenfalls Videodaten übertragen. Außerdem wird es mit DVI ermöglicht, analoge und digitale Abbildungsdaten gleichzeitig zu übertragen. Der hier verwendete DVI-A-Stecker wird in der Praxis nur als Adapter-Kabel zu VGA eingesetzt (vgl. [24]). Die für die Herstellung des Adapters notwendigen Pinbelegungen sind in den Abbildungen 3.3 und 3.4 zu sehen.

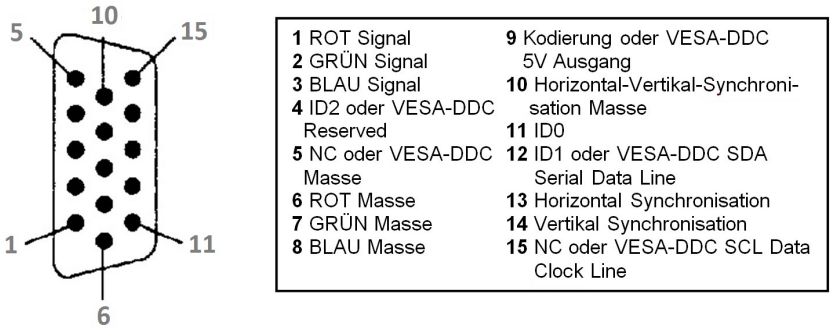
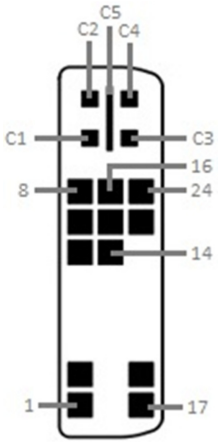


Abbildung 3.3.: Pinbelegung VGA, Autorin: Melanie Schulze



1 TDMS-Daten 2-	**18** TDMS-Daten 0+
2 TDMS-Daten 2+	**23** TDMS-Takt +
6 DDC Takt	**24** TDMS-Takt -
7 DDC Daten	**C1** Analog: Rot
8 Analog: V-Sync	**C2** Analog: Grün
14 +5 Volt	**C3** Analog: Blau
15 Masse für +5 Volt	**C4** Analog: Horizontal Synchronisation
16 Hotplug-Detect	
17 TDMS-Daten 0-	**C5** Analog: Masse

Abbildung 3.4.: Pinbelegung DVI-A, Autorin: Melanie Schulze

Mit Hilfe dieser Pinbelegung gelangt man zu der benötigten Verdrahtung. In Abbildung 3.5 ist aufgelistet, welche Pins des DVI-A-Steckers mit den jeweiligen Pins des VGA-Steckers verbunden werden müssen.
Um die Übersicht zu verbessern, wird dieses auch in Abbildung 3.6 grafisch verdeutlicht (rote Kreuze sind hierbei nicht verwendete Pins).

VGA		**DVI-A**
1 ROT Signal	→	**C1** Analog: Rot
2 GRÜN Signal	→	**C2** Analog: Grün
3 BLAU Signal	→	**C3** Analog: Blau
5 Masse	→	**15** Masse für +5V
6 ROT Masse	→	**C5** Analog: Masse
7 GRÜN Masse	→	**C5** Analog: Masse
8 BLAU Masse	→	**C5** Analog: Masse
9 5V Ausgang	→	**14** +5V
10 Hor.-Vert.-Sync. Masse	→	**15** Masse für +5V
12 Serial Data Line	→	**7** DDC Daten
13 Horizontal Sync.	→	**C4** Horizontal Sync.
14 Vertical Sync.	→	**8** Vertical Sync.
15 Clock	→	**6** Clock

Abbildung 3.5.: Anschlussübersicht von VGA auf DVI-A, Autorin: Melanie Schulze

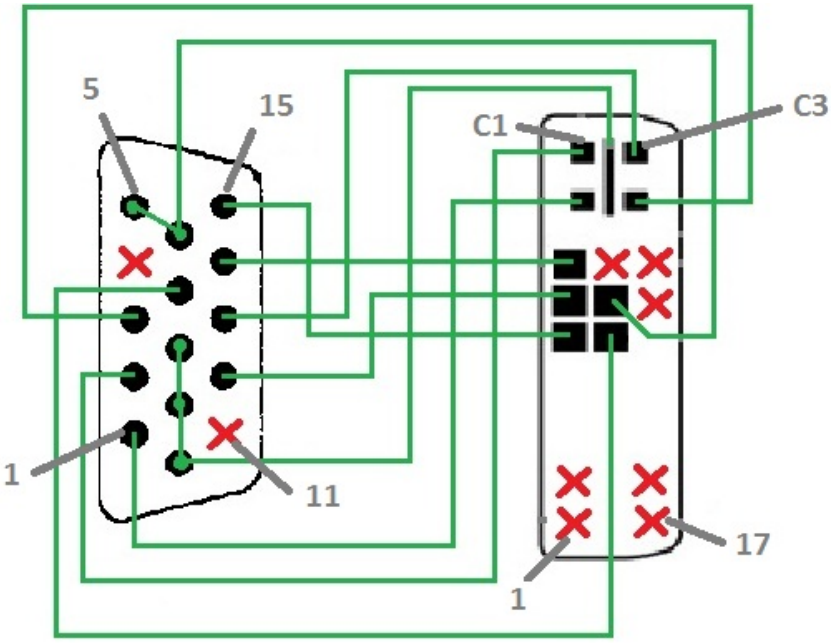


Abbildung 3.6.: Darstellung der Verdrahtung VGA auf DVI-A, Autorin: Melanie Schulze

Dieses Adapterkabel wird in dem Aufbau zur Verbindung des Controllers mit dem vorhandenen Bildschirm verwendet.

3.1.2. Das *PXI*-Messsystem

Vorhanden ist nun ein *PXI*-Messsystem (siehe dazu auch [18]), welches für die spätere Verwendung angepasst ist (siehe Abbildung 3.7).

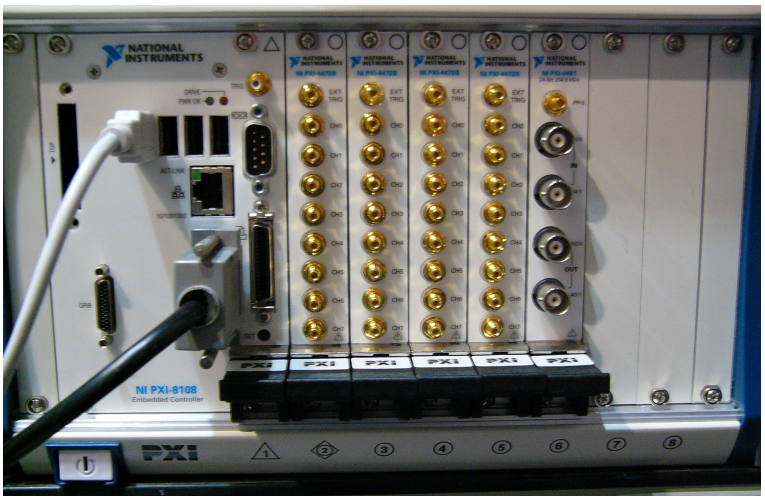


Abbildung 3.7.: *PXI*-Messsystem, Autor: René Pfeifer

Der Aufbau besteht aus einem Chassis vom Typ „NI PXI–1042Q“, welches acht Steckplätze („Slots“) besitzt. Der Controller vom Typ „NI PXI-8108“ bietet die Schnittstellen für den Monitor, Ethernet, die Tastatur, die Maus, USB-Anschlüsse sowie die gesamte Verarbeitungseinheit (CPU und Arbeitsspeicher etc.). Des Weiteren stehen vier Messkarten vom Typ „NI PXI–4472B“ zur dynamischen Erfassung von Signalen zur Verfügung, welche jeweils acht Eingänge und einen Anschluss für ein externes Triggersignal bieten. Schließlich ist eine Messkarte vom Typ „NI PXI-4461“ zur dynamischen Erfassung und Erzeugung von Signalen vorhanden, die je zwei simultan abgetastete Ein- sowie Ausgänge und ebenfalls einen Anschluss für ein externes Triggersignal besitzt.

Wichtige Spezifikationen:

NI PXI–1042Q (aus [13]):

- 8 Steckplätze

NI PXI–8108 (aus [9]):

- Intel Core 2 Duo T9400 CPU (2,53 GHz dual core)
- 2 GB Arbeitsspeicher
- 80 GB Festplatte SATA (5400 $\frac{U}{min}$)

NI PXI–4472B (aus [11]):

- acht Kanäle mit dynamischer Signalerfassung
- Auflösung: 24-Bit
- 102,4 $\frac{kS}{s}$ maximale Samplerate[2]
- Eingangsbereich: $\pm$ 10 V
- AC/DC Kopplung
- Integrated Electronics Piezo Electric („IEPE“; per Software einstellbar)
- Synchronisieren mehrerer Geräte möglich

NI PXI–4461 (aus [10]):

- zwei simultan einlesende analoge Eingänge

- zwei simultan ausgebende analoge Ausgänge
- Auflösung: 24-Bit
- 204,8 $\frac{kS}{s}$ maximale Samplerate[2]
- Eingangsbereich: $\pm$ 316 mV bis 42,4 V
- AC/DC Kopplung
- IEPE (per Software einstellbar)
- Synchronisieren mehrerer Geräte möglich

3.1.3. Die Beschleunigungssensoren

Es stehen zurzeit sechs Beschleunigungssensoren „Triaxial ICP Accelerometer - Model 356B18“ der Firma „PCB PIEZOTRONICS“ (siehe Abbildung 3.8) zur Verfügung. Mit diesen soll später die Messdatenerfassung am Hubschrauberwindkanalmodell stattfinden.

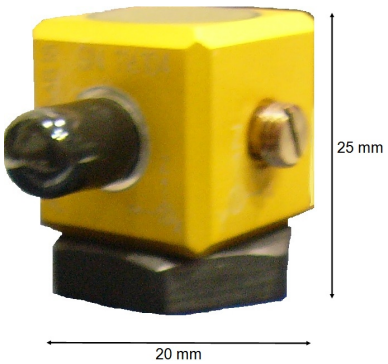


Abbildung 3.8.: Beschleunigungssensor, Autor: René Pfeifer

Diese Sensoren erfassen Beschleunigungen in drei Richtungen. Diese sind mit X, Y und Z auf dem Sensor gekennzeichnet. Die Sensoren benötigen einen konstanten Versorgungsstrom, welcher üblicherweise zwischen 2 mA und 20 mA liegt (die hier verwendeten Sensoren benötigen 4 mA). Diese Versorgungsart heißt IEPE und ist ein Industriestandard für piezoelektrische Sensoren mit eingebauter Impedanzwandler-Elektronik. Der Vorteil hiervon ist, dass das Sensorsignal zusammen mit dem Versorgungsstrom über ein Koaxialkabel übertragen werden kann (vgl. [25]).

Wichtige Spezifikationen (aus [20]):

- Sensitivität ($\pm$ 10 %): 102 $\frac{mV}{\frac{m}{s^2}}$
- Frequenzbereich ($\pm$ 5 %): 0,5 bis 3000 Hz

3.1.4. Der Kraftsensor

Vorhanden ist zudem ein Kraftsensor „Force Transducer – Type 8200“ der Firma „Brüel & Kjær“ (siehe Abbildung 3.9).

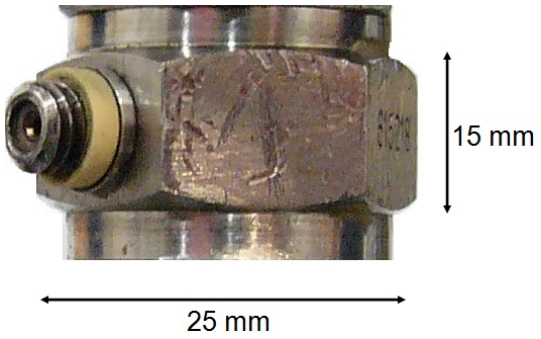


Abbildung 3.9.: Kraftsensor, Autorin: Melanie Schulze

Mit diesem Sensor wird die Kraft gemessen, die der Shaker auf das Modell überträgt. Hiermit soll es später möglich sein, eine Übertragungsfunktion (Ausgang / Eingang) zu berechnen.
Der Kraftsensor arbeitet mit dem piezoelektrischen Prinzip eines Quarzes. Ein Quarzkristall reagiert auf Drücken oder Ziehen mit einer elektrischen Polarisierung längs der Kraftrichtung, denn durch seine Verformung werden mikroskopische Dipole innerhalb der Elementarzellen ausgebildet (vgl. [26]). Diese Verschiebung der Ladungsschwerpunkte führt zu einer messbaren Spannung. Diese Spannung ist proportional zur drückenden bzw. ziehenden Kraft.

Wichtige Spezifikationen (aus [7]):

- Kraftbereich: -1000N bis +5000N
- Sensitivität: 4 $\frac{pC}{N}$

3.1.5. Die Verstärker

Wie in Abbildung 1.1 zu erkennen ist, sind zwei unterschiedliche Verstärker vorhanden. Bei dem einen handelt es sich um einen Ladungsverstärker „Charge Amplifier 2651“ der Firma „Brüel & Kjær“ (siehe Abbildung 3.10).

Wichtige Spezifikationen (aus [1]):

- Sensitivität: 0,1 $\frac{mV}{pC}$, 1 $\frac{mV}{pC}$ und 10 $\frac{mV}{pC}$
- Tiefpass-Filterung: 1 Hz für alle Sensitivitäten oder 0,003 Hz für 0,1 $\frac{mV}{pC}$, 0,03 Hz für 1 $\frac{mV}{pC}$ und 0,3 Hz für 10 $\frac{mV}{pC}$

Abbildung 3.10.: Ladungsverstärker, Quelle: [19]

Ein Ladungsverstärker gibt eine zu einer Ladung proportionale elektrische Spannung aus. Dies wird meistens benutzt, um die kleine Ladung eines piezoelektrischen Sensors in eine Spannung umzuwandeln, die zu dieser Ladung proportional ist (vgl. [27]). Auch hier wird der Ladungsverstärker zu genau diesem Zweck verwendet. An ihm ist der Kraftsensor (siehe Abschnitt 3.1.4) angeschlossen.
Im Folgenden soll kurz das Prinzip eines Ladungsverstärkers erläutert werden.
Ein Ladungsverstärker wird gewöhnlich mit einem Operationsverstärker ausgeführt (siehe Abbildung 3.11).
Es ist zu sehen, dass parallel zum Sensor zwei Kapazitäten liegen. Dieses sind zum einen Kabelkapazitäten (C_C) und zum anderen die Eingangskapazität des Verstärkers (C_{inp}).

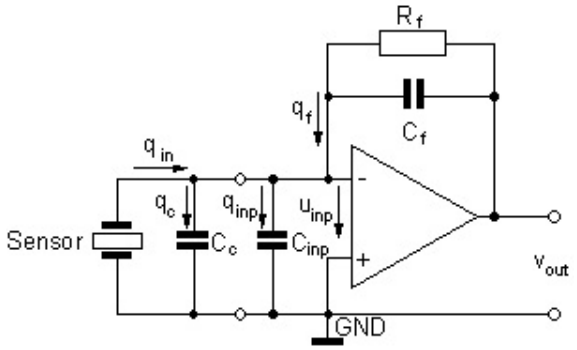


Abbildung 3.11.: Prinzipschaltung Ladungsverstärker, Quelle: [2]

Genau diese Kapazitäten sind das Problem bei Spannungsverstärkern mit hohem Eingangswiderstand, denn dort ist die Ausgangsspannung von diesen Kapazitäten abhängig.

Bei Ladungsverstärkern hängt die Ausgangsspannung jedoch nur von der eingespeisten Ladung (hier: q_{in}) und der Rückkoppelkapazität (hier: C_f) ab, was ein großer Vorteil ist. Dass die Ausgangsspannung nicht von den oben beschriebenen Kapazitäten abhängt, liegt daran, dass der nichtinvertierende Eingang des Operationsverstärkers (+) auf Massepotential liegt und der Operationsverstärker dafür sorgt, dass auch der invertierende Eingang (-) auf Massepotential liegt, wodurch die Spannung an seinem Eingang (hier u_{inp}) Null wird (vgl. [22]).

Der andere Verstärker ist ein Leistungsverstärker „Power Amplifier Type 2712“ der Firma „Brüel & Kjær“ (siehe Abbildung 3.12).

Abbildung 3.12.: Leistungsverstärker, Autorin: Melanie Schulze

Dieser wird verwendet, um das Signal, das von der Messkarte (siehe Abschnitt 3.1.2) ausgegeben wird, so zu verstärken, dass dem Shaker (siehe Abschnitt 3.1.6) die benötigte Leistung bereitgestellt wird.

3.1.6. Der Shaker

Ein Shaker ist ein Gerät, das der Anregung von Schwingungen in Bauteilen dient.
Der hier verwendete Shaker von Goodmans Industries (Type 790) ist in Abbildung 3.13 zu sehen. Mit diesem Shaker werden später auch die Hubschrauberwindkanalmodelle angeregt.

Abbildung 3.13.: Shaker, Autor: René Pfeifer

Im Inneren eines Shakers befindet sich ein permanentes Magnetfeld, in welchem sich ein stromdurchflossener elektrischer Leiter (eine Spule) befindet. Da sich in dem Magnetfeld bewegte Ladungen befinden, entsteht eine Lorentz-Kraft. An der Stirnseite des Shakers ist ein Stößel angebracht, welcher durch diese Kraft bewegt wird und mit dem die Schwingungen an das Testobjekt übertragen werden.

3.2. Einarbeitung in die Software

Im Folgenden wird auf die Software *Measurement and Automation Explorer* („MAX“), *LabVIEW* sowie *DIAdem* eingegangen.

3.2.1. Das Programm *Measurement and Automation Explorer* von *National Instruments*

Der *MAX* wird von *NI* angeboten, um verschiedene *NI*-Komponenten zu managen (siehe Abbildung 3.14).

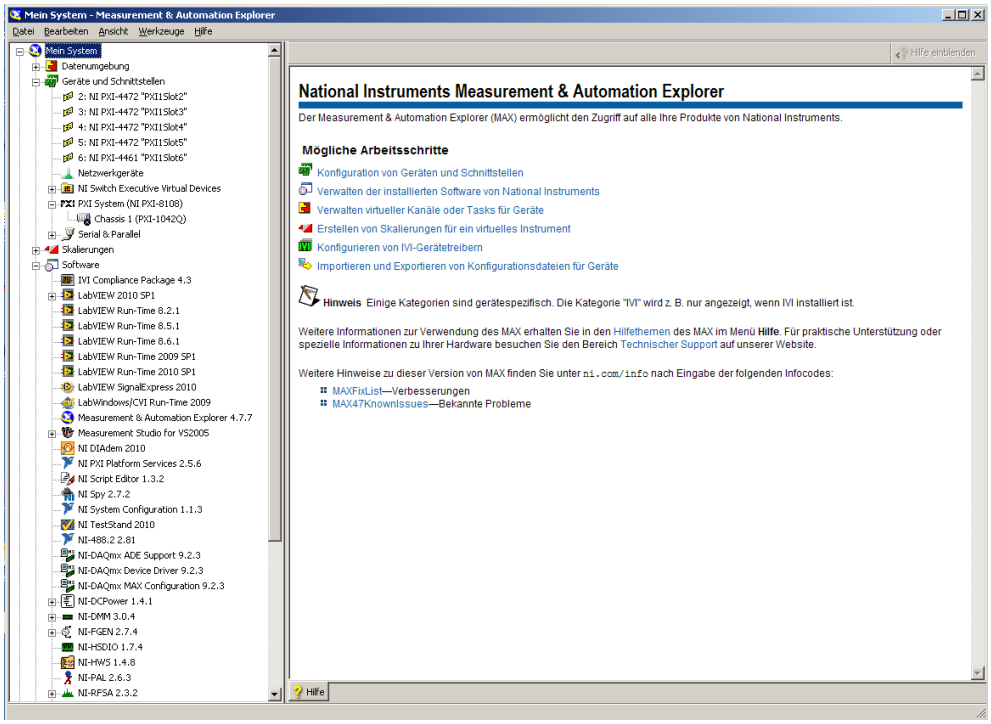


Abbildung 3.14.: Oberfläche des *MAX*, Autor: René Pfeifer

In dem *MAX* lässt sich zum Beispiel der komplette Aufbau des hier verwendeten Systems simulieren. Das hat den Vorteil, dass keine direkte Verbindung zum Messsystem bestehen muss, um ein Programm auszuführen.
Schlussendlich muss das Programm jedoch am realen System getestet werden, um zu überprüfen, ob es dort fehlerfrei funktioniert.

Es musste sich zunächst in den *MAX* eingearbeitet und das Messsystem als simulierte Hardware hinzugefügt werden. Dieses hat des Weiteren den Vorteil, dass alle im *MAX* eingetragenen Hardware-Elemente bei der Programmierung in *LabVIEW* berücksichtigt werden und im *LabVIEW*-Programm direkt zu Auswahl stehen (siehe Abbildung 3.15).

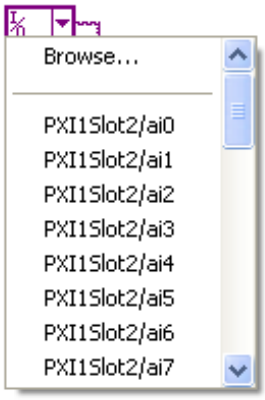


Abbildung 3.15.: Auswahl der Hardware in *LabVIEW*, Autor: René Pfeifer

Das Einbinden von simulierter Hardware im *MAX* wird im Folgenden erläutert.

Zuerst muss dem *MAX* ein Chassis hinzugefügt und dieses richtig identifiziert werden (vgl. Abbildung 3.16)

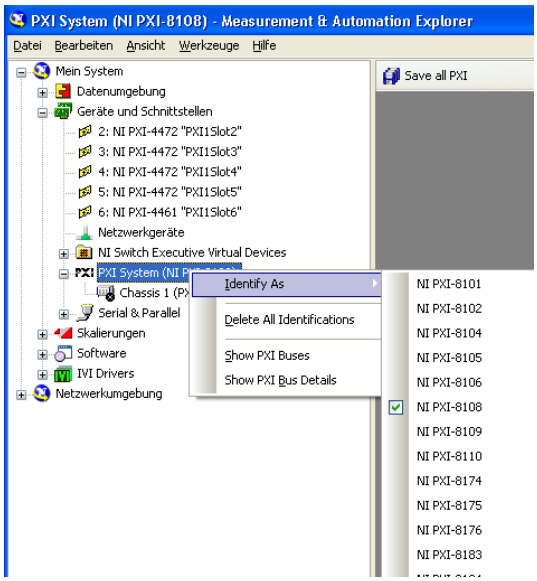


Abbildung 3.16.: Chassis Auswahl (*MAX*), Autor: René Pfeifer

Danach werden die simulierten Karten hinzugefügt.
Dazu muss als erstes unter dem Punkt „Geräte und Schnittstellen“ „Neu...“ ausgewählt werden (vgl. Abbildung 3.17).

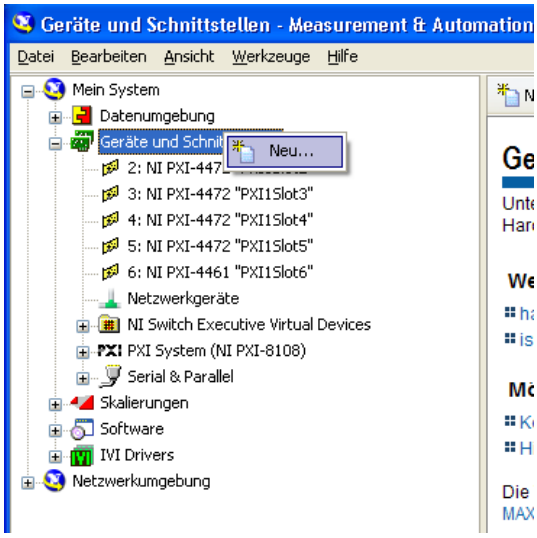


Abbildung 3.17.: Neue Karte einbinden (*MAX*), Autor: René Pfeifer

Anschließend muss „Simuliertes NI-DAQmx-Gerät oder modulares Instrument“ angeklickt werden (vgl. Abbildung 3.18).

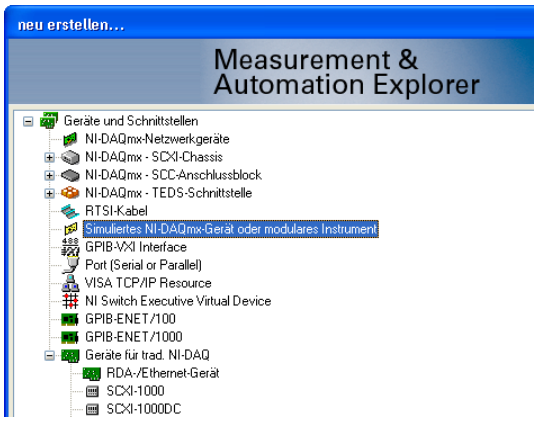


Abbildung 3.18.: Schnittstellen Auswahl (*MAX*), Autor: René Pfeifer

Daraufhin können die vorhanden Messkarten gesucht und in den *MAX* eingebunden werden (vgl. Abbildungen 3.19 und 3.20).

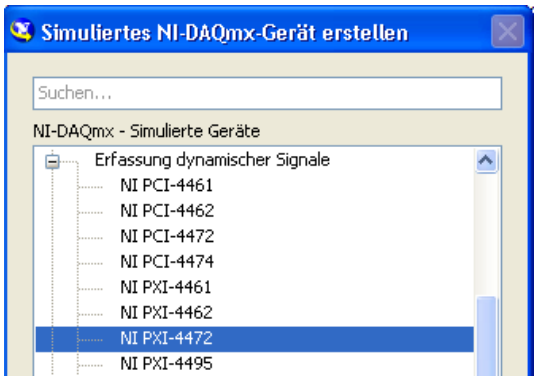


Abbildung 3.19.: Auswahl der Messkarte „PXI-4472“ (*MAX*), Autor: René Pfeifer

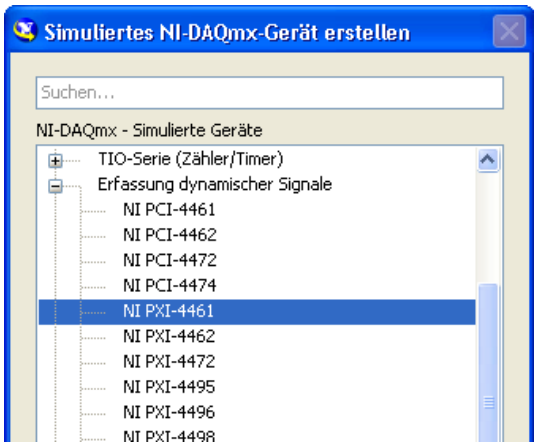


Abbildung 3.20.: Auswahl der Messkarte „PXI-4461“ (*MAX*), Autor: René Pfeifer

Zudem bietet der *MAX* eine Übersicht über die verfügbare Software, wie zum Beispiel *LabVIEW* und *DIAdem*. Die aufgelistete Software kann auch direkt über den *MAX* gestartet und verwaltet werden. Das schließt auch Updates der *NI*-Software mit ein.

In Abbildung 3.21 sind die simulierten Messkarten zu sehen. Den Karten wird durch das Einfügen in der richtigen Reihenfolge die jeweilige *PXI*-Steckplatz- sowie *PXI*-Chassis-Nummer zugewiesen. Damit ist es möglich, dass die simulierten Karten im *MAX* dem originalen Aufbau der Messkarten des realen Messystems entsprechen.

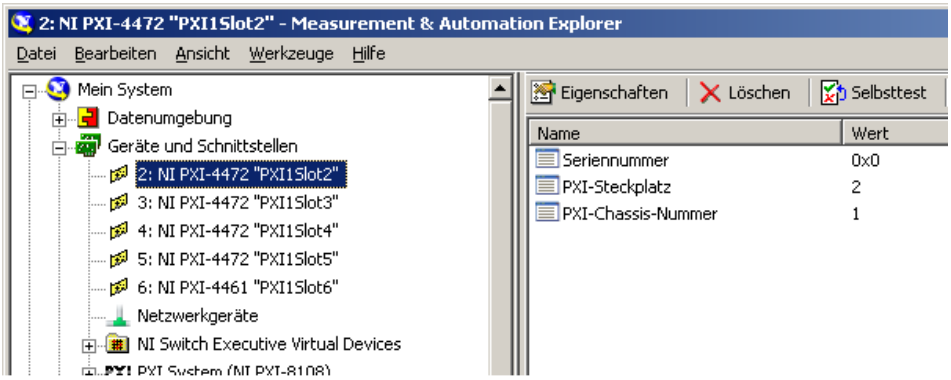


Abbildung 3.21.: Hardware Übersicht im *MAX*, Autor: René Pfeifer

3.2.2. Das Programm *LabVIEW* von *National Instruments*

Die Aufgabe ist es, mit *LabVIEW* eine benutzerfreundliche und sichere Bedienoberfläche zu erstellen, mit deren Hilfe eine Strukturschwingungsanalyse von Hubschrauberwindkanalmodellen durchgeführt werden kann. *LabVIEW* bietet eine programmierfreundliche Oberfläche und ist zudem stark am Markt verbreitet. Zur Verfügung steht die *LabVIEW 2010 DS1 Developer Suite* (Sprache: Englisch).

Bei *LabVIEW* gibt es zwei Fenster: Einerseits das Frontpanel (siehe Abbildung 3.22), auf dem die Bedienoberfläche für den späteren Bediener dargestellt wird, andererseits das Blockdiagramm (siehe Abbildung 3.23), in welchem alle Bausteine miteinander verknüpft werden und der Programmcode bzw. Programmfluss dargestellt wird.

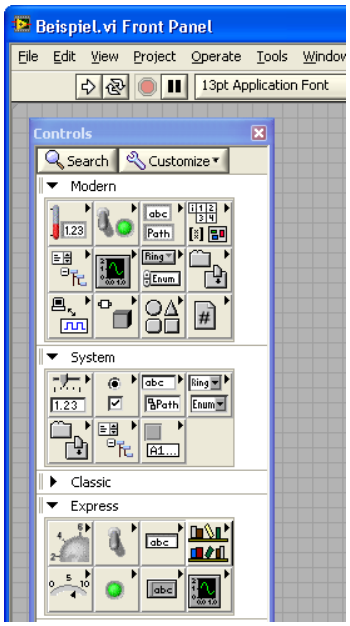


Abbildung 3.22.: Oberfläche mit der Elemente-Leiste des Frontpanels, Autor: René Pfeifer

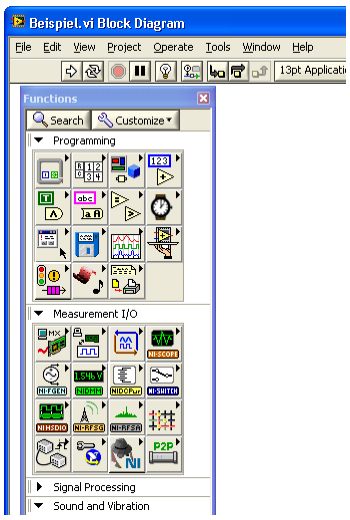


Abbildung 3.23.: Oberfläche mit der Funktionen-Leiste des Blockdiagramms, Autor: René Pfeifer

Als erstes muss sich in die Bedienoberfläche von *LabVIEW* eingearbeitet werden. Erste Informationen zu den Messkarten sind aus einer Webcast-Serie zu *PXI*-Systemen von *NI* (vgl. [15]) zu entnehmen.
Des Weiteren bietet *LabVIEW* in der „Hilfe“ (siehe [8]) eine „Indexsuche“. In dieser sind alle Bausteine sowie Begriffe erläutert. Dazu benötigt man jedoch die Bezeichnung des Bausteins oder Begriffs.
Zusätzlich zur Hilfe sind allerdings auch Beispielprogramme (siehe Abbildung 3.24) vorhanden, die nach Kategorien geordnet sind, sodass eine leichte Suche gewährleistet ist.

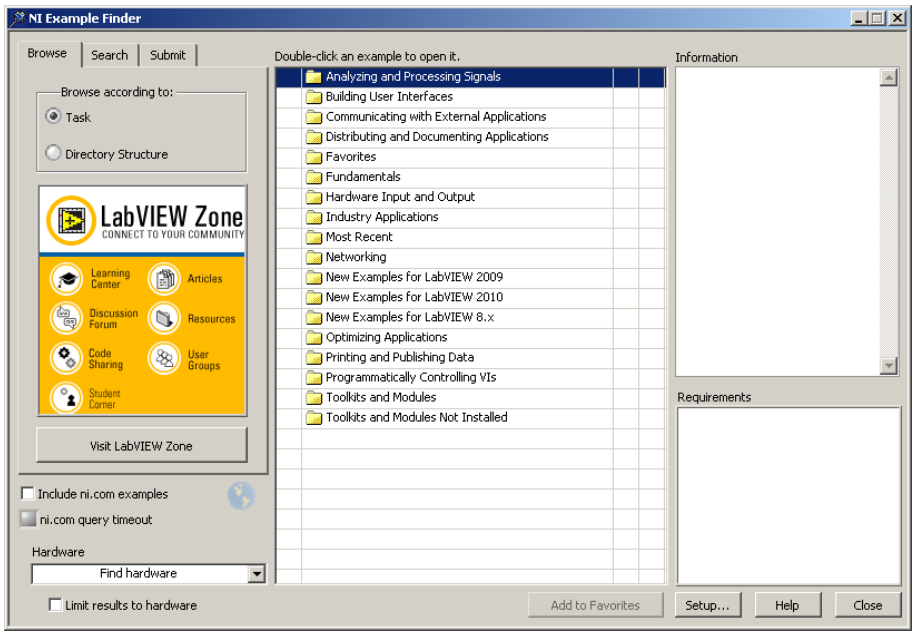


Abbildung 3.24.: Übersicht der Beispiele von *LabVIEW*, Autor: René Pfeifer

Im Weiteren werden wichtige Elemente und Informationen von *LabVIEW* kurz erläutert.

Dazu gehört unter anderem der Error-Cluster[4] (siehe Abbildung 3.25). Bei diesem Cluster[4] handelt es sich um eine Leitung, die gegebenenfalls aufgetretene Fehler an den nächsten Baustein weitergibt. Tritt kein Fehler auf, so wird auch diese Information übermittelt.

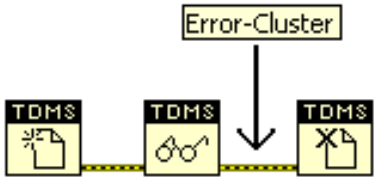


Abbildung 3.25.: Beispiel eines Error-Clusters[4] (Blockdiagramm), Autor: René Pfeifer

Die wichtigste Eigenschaft ist jedoch, dass durch diesen Cluster[4] gewährleistet wird, dass ein Baustein wirklich erst dann abgearbeitet wird, wenn der vorherige Baustein fertig ist.

Als nächstes soll auf die Bedienelemente hingewiesen werden. Bedienelemente gibt es nur an einer Stelle im Programm. Soll im späteren Programm auf dieses Bedienelement zugegriffen werden, kann von diesem eine lokale Variable oder eine Referenz erstellt werden. Damit kann der aktuelle Wert des Bedienelements in späteren Programmteilen ausgelesen bzw. überschrieben werden. Um den Wert einer Referenz zu ändern gibt es den Eigenschaftsknoten. An diesen wird die Referenz des Bedienelements angeschlossen und er hat den Vorteil, dass damit noch andere Eigenschaften des Bedienelements verändert werden können, wie z.B. ob das Bedienelement deaktiviert oder aktiviert werden soll (siehe Abbildung 3.26). Diese zusätzlichen Einstellungen sind bei einer einfachen lokalen Variable nicht möglich.

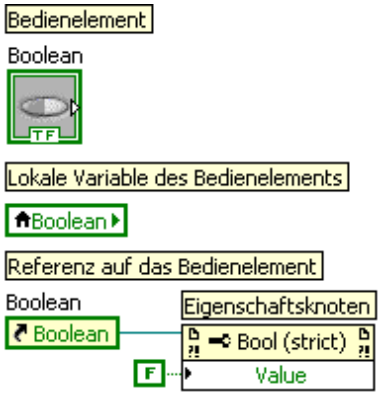


Abbildung 3.26.: Bedienelement, Lokale Variable und Referenz des Bedienelements (Blockdiagramm), Autor: René Pfeifer

Zum Schluss wird die Sequenzstruktur erklärt. Diese wird von links nach rechts abgearbeitet. Die nächste Sequenz erfolgt erst, wenn alle in diese Sequenz eingehenden Signal vorhanden sind (siehe Abbildung 3.27).

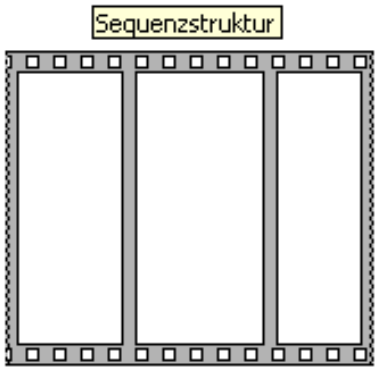


Abbildung 3.27.: Sequenzstruktur (Blockdiagramm), Autor: René Pfeifer

Nachdem einige wichtige *LabVIEW*-Elemente erläutert wurden, wird nun näher auf Bausteine eingegangen, welche für das Arbeiten mit dem *PXI*-System notwendig sind. Die spätere *LabVIEW*-Applikation wird direkt auf dem Controller ausgeführt. Dadurch ist eine Verbindung zu den Messkarten schon gewährleistet und braucht nicht hergestellt zu werden. Jedoch müssen die Messkarten richtig initialisiert und angesprochen werden. Die gezeigten Beispiele in den Abbildungen 3.30 bis 3.33 dienen als Grundgerüst für das weitere Programm. In den Beispielen werden unter anderem die *Data Acquisition*(„DAQ“)mx-Bausteine verwendet, welche dazu dienen, eine Verbindung mit den Karten („virtual channel“) herzustellen. Durch diese wird es auch ermöglicht, die einzelnen Messeingänge separat zu konfigurieren (vgl. Abbildung 3.28). Die einzelnen Bausteine dieser Abbildung werden weiter unten erläutert.

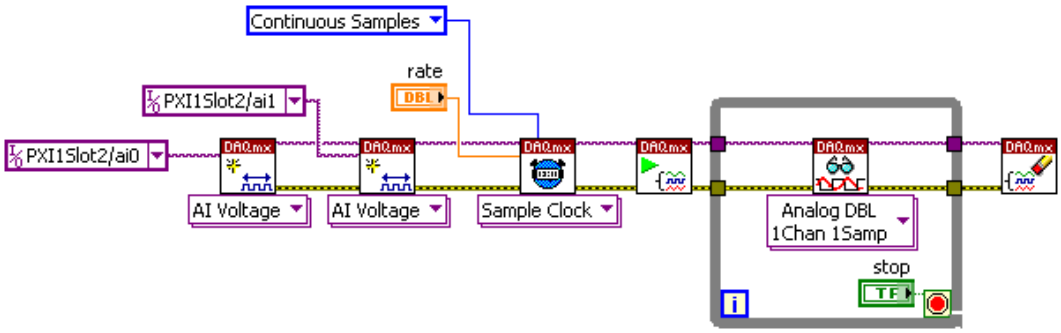


Abbildung 3.28.: Beispiel zur Konfiguration einzelner Kanäle einer Messkarte (Blockdiagramm), Autor: René Pfeifer

Zur Ausgabe über die Messkarte „PXI-4461“ sind Beispiele unter dem Pfad „Hardware Input and Output ⇒ DAQmx ⇒ Analog Generation ⇒ Voltage“ zu finden (siehe Abbildung 3.29).

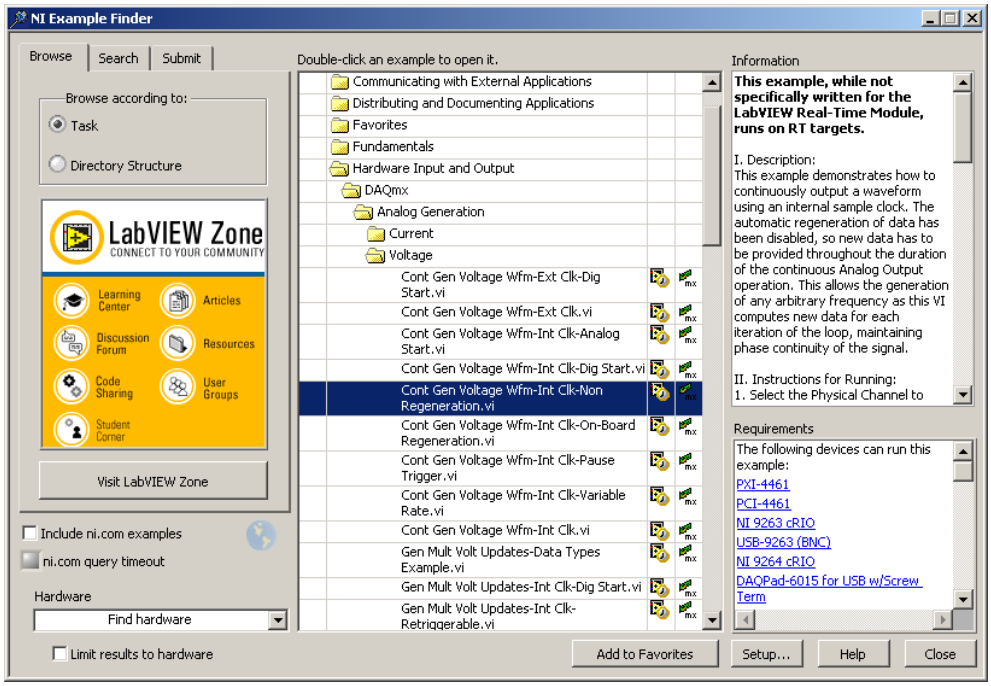


Abbildung 3.29.: *LabVIEW*-Beispiele: Ausgabe, Autor: René Pfeifer

In den Abbildungen 3.30 und 3.31 ist das gefundene Beispiel (Pfad: Hardware Input and Output ⇒ DAQmx ⇒ Analog Generation ⇒ Voltage ⇒ Cont Gen Voltage Wfm-Int Clk-Non Regeneration.vi) zu sehen, mit welchem die Signalerzeugung stattfindet.
In Abbildung 3.30 befinden sich links am Rand und rechts unten Einstellmöglichkeiten, die für die Ausgabe des Signals notwendig sind. Das sind unter anderem Auswahl der Hardware („Physical Channel“), der Signaltyp, die Frequenz und die Amplitude. Oben rechts im Bild ist ein Anzeigeelement zu erkennen, welches das generierte Signal darstellt.
In Abbildung 3.31 ist der zu Abbildung 3.30 gehörende Programmcode zu sehen. Links vor der While-Schleife werden die Parameter zur Ausgabe konfiguriert. Die Signalgenerierung und Datenausgabe finden in der While-Schleife statt. Nach Beenden dieser wird die Verbindung zur Messkarte geschlossen und reservierte Ressourcen wieder freigegeben.

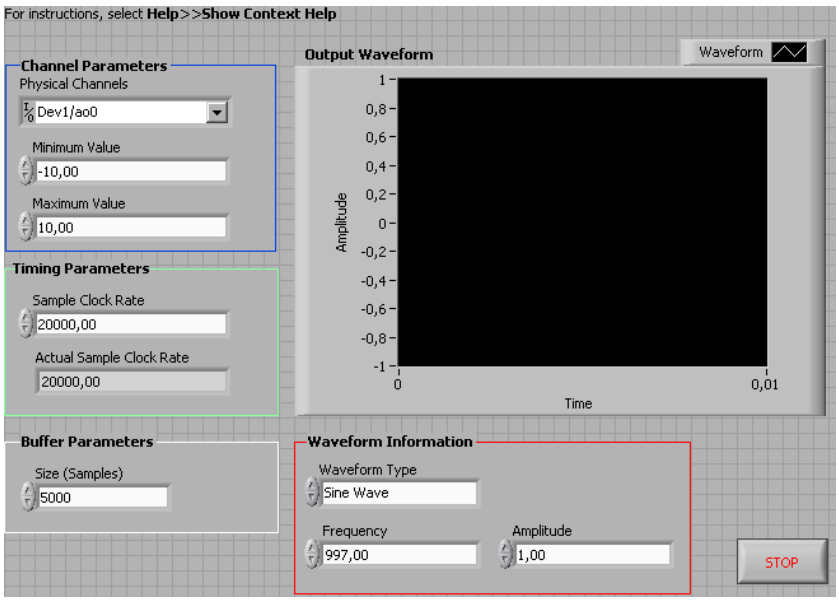


Abbildung 3.30.: *LabVIEW* Beispiel: Ausgabe über die „PXI-4461“ (Frontpanel), Autor: René Pfeifer

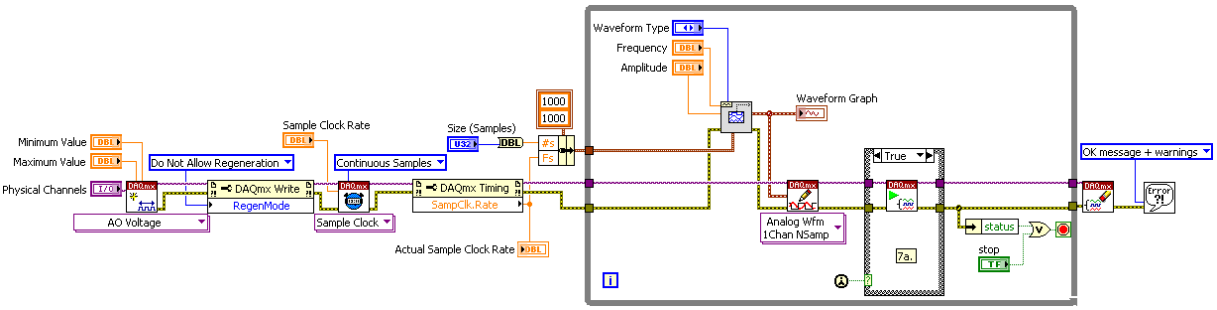


Abbildung 3.31.: *LabVIEW* Beispiel: Ausgabe über die „PXI-4461“ (Blockdiagramm), Autor: René Pfeifer

Im Folgenden wird ein Beispiel (Pfad: Hardware Input and Output ⇒ DAQmx ⇒ Analog Measurements ⇒ Acceleration ⇒ Cont Acq Accel Samples-Int Clk-Analog Start.vi) zum Einlesen mittels der Messkarten gezeigt (siehe Abbildung 3.32 und 3.33).
In Abbildung 3.32 sind ebenfalls verschiedene Parameter zum Einstellen vorhanden, welche unter anderem dazu dienen, Beschleunigungssensoren, welche eine konstante Stromversorgung benötigen, mit dieser zu versorgen. In Abbildung 3.33 ist ein ähnlicher Programmaufbau wie bei der Ausgabe zu sehen. Der wesentliche Unterschied ist, dass hier ein Einlesekanal geöffnet wird und das Erfassen vor der While-Schleife gestartet wird. Zudem wird hier auf ein Signal getriggert, welches zum Beispiel als Startsignal zur Erfassung der Daten dienen kann. Anschließend wird die Verbindung geschlossen und Ressourcen werden wieder freigegeben.

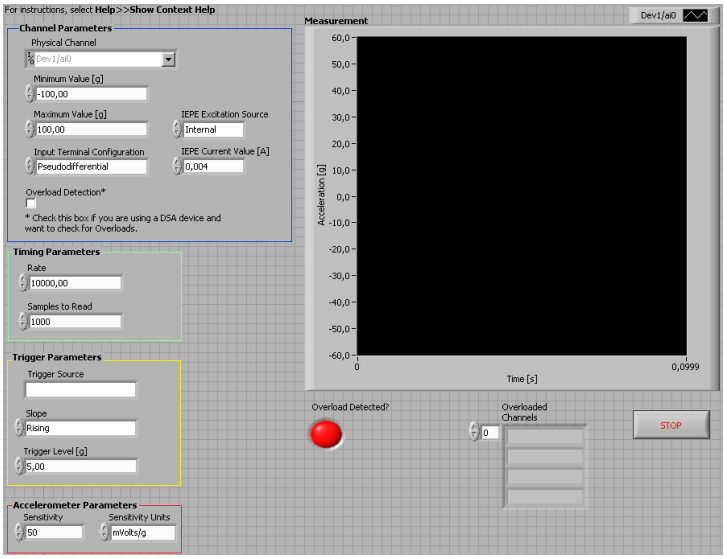


Abbildung 3.32.: *LabVIEW* Beispiel: zum Einlesen der Sensoren (Frontpanel), Autor: René Pfeifer

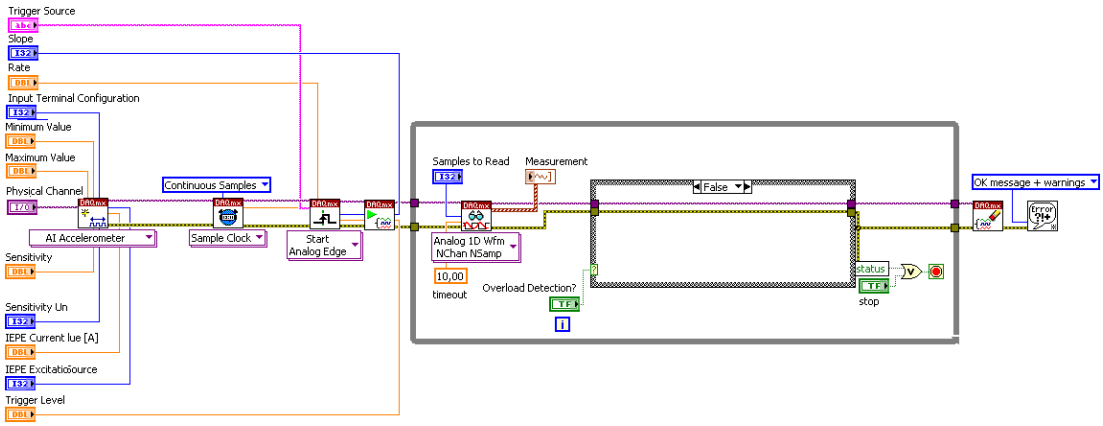


Abbildung 3.33.: *LabVIEW* Beispiel: Einlesen der Sensoren (Blockdiagramm), Autor: René Pfeifer

Die verwendeten *DAQmx*-Bausteine werden im Weiteren in folgender Reihenfolge genauer erläutert.

- „DAQmx Create Channel.vi“
- „DAQmx Timing.vi“
- „DAQmx Start Trigger.vi“
- „DAQmx Read.vi“
- „DAQmx Write.vi“

- „DAQmx Start Task.vi“
- „DAQmx Clear Task.vi“

Das „DAQmx Create Channel.vi“ (siehe Abbildung 3.34) dient dem Öffnen eines virtuellen Kanals zu der gewählten Hardware. Bei diesem virtual instrument („VI“; Dateiendung von *LabVIEW*-Dateien) werden erste Konfigurationen des Kanals vorgenommen, z.B., ob der Task[1] ein Aus- oder Eingang sein soll.

Tabelle 3.1.: Erklärung der Ein- und Ausgänge des „DAQmx Create Channel.vi“

Ein- /Ausgang	Erklärung
„input terminal configuration“	legt die Erfassungsart der Signale fest
„minimum/maximum value“	legt die untere/obere Grenze der zu erfassenden Größe für die Messkarte fest
„task in“	damit können verschiedene Tasks[1] verbunden werden
„physical channels“	hier werden die Karte, bzw. die Kanäle ausgewählt, zu denen der virtuelle Kanal geöffnet werden soll
„name to assign“	legt einen Namen für den virtuellen Kanal fest
„units“	legt die Einheit fest, mit der die Messdaten zurückgegeben werden sollen
„error in“	Eingang für Fehler vorheriger Tasks[1]
„custom scale name“	dient der benutzerdefinierten Skalierung der eingelesenen oder ausgegebenen Signale
„task out“	gibt die Referenz des Tasks[1] an den nächsten Baustein weiter
„error out“	gibt eventuelle Fehler an den nächsten Baustein weiter

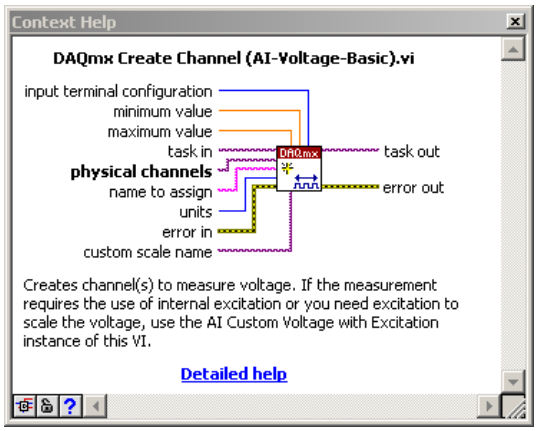


Abbildung 3.34.: *DAQmx* Create Channel, Autor: René Pfeifer

Das „DAQmx Timing.vi“ (siehe Abbildung 3.35) dient dem Festlegen der Samplerate[2] und der Anzahl an Samples[3] sowie der automatischen Bestimmung der Größe eines eventuell benötigten Puffers.

Tabelle 3.2.: Erklärung der Ein- und Ausgänge des „DAQmx Timing.vi“

Ein- /Ausgang	Erklärung
„samples per channel“	legt die Anzahl an Messpunkten fest, die erfasst oder erzeugt werden sollen
„sample mode“	hier kann zwischen kontinuierlicher und endlicher Erfassung gewählt werden
„task/channels in“	Anschluss der Taskreferenz
„rate“	Wahl der gewünschten Samplerate[2]
„source“	Auswahl der Taktquelle, Standard ist der Onboard-Clock
„active edge“	Auswahl, ob auf steigende oder fallende Flanke reagiert werden soll
„error in“	Eingang für Fehler vorheriger Tasks[1]
„task out“	gibt die Referenz des Tasks[1] an den nächsten Baustein weiter
„error out“	gibt eventuelle Fehler an den nächsten Baustein weiter

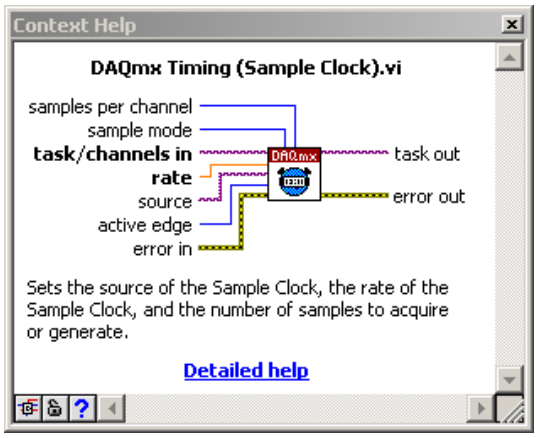


Abbildung 3.35.: *DAQmx* Timing, Autor: René Pfeifer

Das „DAQmx Start Trigger.vi“ (siehe Abbildung 3.36) dient dem Triggern eines Signals. Dafür können verschiedene Arten von Triggern gewählt werden, wie unter anderem der Starttrigger.

Tabelle 3.3.: Erklärung der Ein- und Ausgänge des „DAQmx Start Trigger.vi“

Ein- /Ausgang	Erklärung
„task/channels in“	Anschluss der Taskreferenz
„error in“	Eingang für Fehler vorheriger Tasks[1]
„task out“	gibt die Referenz des Tasks[1] an den nächsten Baustein weiter
„error out“	gibt eventuelle Fehler an den nächsten Baustein weiter

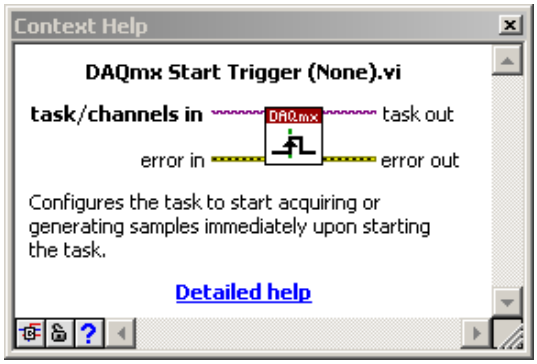


Abbildung 3.36.: *DAQmx* Start Trigger, Autor: René Pfeifer

Das „DAQmx Read.vi“ (siehe Abbildung 3.37) dient dem Einlesen von Daten.

Tabelle 3.4.: Erklärung der Ein- und Ausgänge des „DAQmx Read.vi“

Ein- /Ausgang	Erklärung
„task/channels in“	Anschluss der Taskreferenz
„error in“	Eingang für Fehler vorheriger Tasks[1]
„task out“	gibt die Referenz des Tasks[1] an den nächsten Baustein weiter
„data“	Ausgang für die eingelesenen Daten, diese können direkt über ein Anzeigeelement in *LabVIEW* dargestellt und/oder in einer Datei, wie z.B. in einer *TDMS*-Datei gespeichert werden
„error out“	gibt eventuelle Fehler an den nächsten Baustein weiter

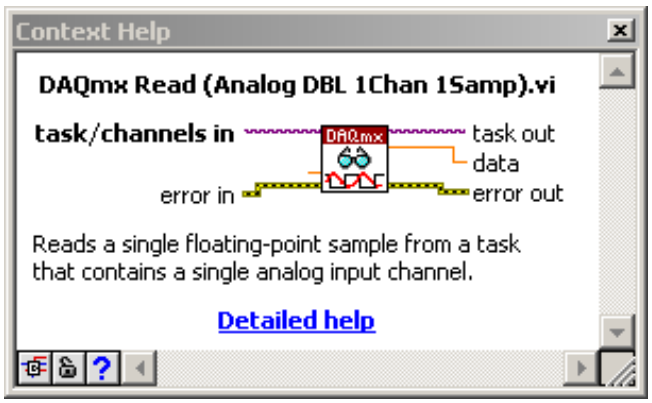


Abbildung 3.37.: *DAQmx* Read, Autor: René Pfeifer

Das „DAQmx Write.vi“ (siehe Abbildung 3.38) dient dem Schreiben von Daten auf den konfigurierten Kanal.

Tabelle 3.5.: Erklärung der Ein- und Ausgänge des „DAQmx Write.vi“

Ein- /Ausgang	Erklärung
„auto start“	dient dem automatischen Starten, wenn kein „DAQmx Start Task.vi“ verwendet wird
„task/channels in“	Anschluss der Taskreferenz
„data“	Eingang für die auszugebenden Daten, welche an dem konfigurierten Kanal ausgegeben werden
„error in“	Eingang für Fehler vorheriger Tasks[1]
„task out“	gibt die Referenz des Tasks[1] an den nächsten Baustein weiter
„number of samples written per channel“	gibt die Anzahl der geschriebenen Samples[3] pro Kanal an
„error out“	gibt eventuelle Fehler an den nächsten Baustein weiter

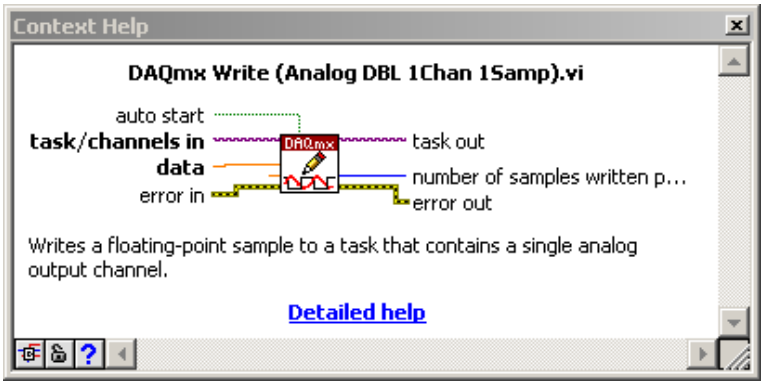


Abbildung 3.38.: *DAQmx* Write, Autor: René Pfeifer

Das „DAQmx Start Task.vi“ (siehe Abbildung 3.39) dient dem Starten des Tasks[1].

Tabelle 3.6.: Erklärung der Ein- und Ausgänge des „DAQmx Start Task.vi“

Ein- /Ausgang	Erklärung
„task/channels in“	Anschluss der Taskreferenz
„error in“	Eingang für Fehler vorheriger Tasks[1]
„task out“	gibt die Referenz des Tasks[1] an den nächsten Baustein weiter
„error out“	gibt eventuelle Fehler an den nächsten Baustein weiter

Abbildung 3.39.: *DAQmx* Start Task[1], Autor: René Pfeifer

Das „DAQmx Clear Task.vi“ (siehe Abbildung 3.40) dient dem Stoppen und Beenden des Tasks[1]. Zudem gibt dieser Baustein alle reservierten Ressourcen wieder frei.

Tabelle 3.7.: Erklärung der Ein- und Ausgänge des „DAQmx Clear Task.vi“

Ein- /Ausgang	Erklärung
„task/channels in“	Anschluss der Taskreferenz
„error in“	Eingang für Fehler vorheriger Tasks[1]
„error out“	gibt eventuelle Fehler an den nächsten Baustein weiter

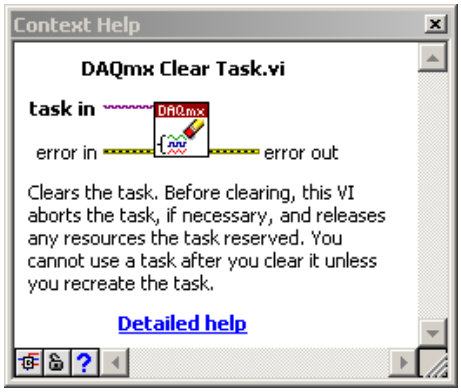


Abbildung 3.40.: *DAQmx* Clear Task[1], Autor: René Pfeifer

3.2.3. Das Programm *DIAdem* von *National Instruments*

Zur Einarbeitung in *DIAdem* wurde unter anderem ein Handbuch von *NI* (siehe [12]) verwendet.
Mit *DIAdem* sollen die Messdaten ausgewertet und für den Anwender übersichtlich dargestellt werden, denn *DIAdem* ist ein Programm von *NI*, welches speziell dafür konzipiert ist. Es sind viele Auswertefunktionen vorhanden und die Ergebnisse können zum Beispiel in Diagrammen oder Tabellen dargestellt werden. Außerdem können Berichte erstellt werden, in denen die Ergebnisse zusammengefasst werden können.
Zur Verfügung steht die Version *DIAdem 2010* (Sprache: Englisch).

In *DIAdem* kann mit verschiedenen Modulen gearbeitet werden.
Als erstes gibt es das Modul „NAVIGATOR“ (siehe Abbildung 3.41). Der Navigator dient dazu, externe Daten (also Daten, die außerhalb von *DIAdem* gespeichert sind) zu laden. Dazu wird im linken Fenster „External Data“ die jeweilige Datei ausgesucht und diese wird dann mit der Maus in das Fenster „Data Portal: Internal Data“ gezogen. Die Daten, die sich dort befinden, können nun mit *DIAdem* betrachtet, bearbeitet und anschließend gespeichert werden.

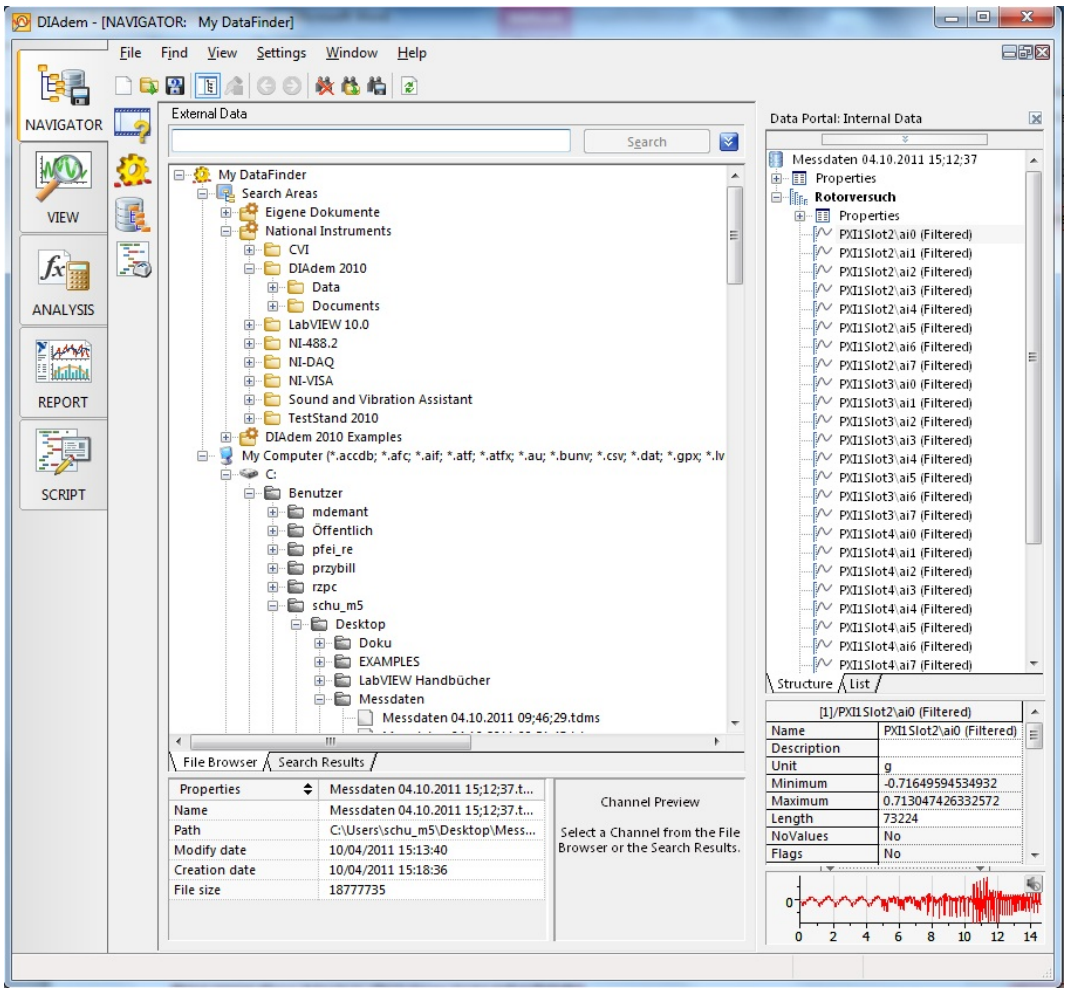


Abbildung 3.41.: Das Modul „DIAdem-NAVIGATOR“, Autorin: Melanie Schulze

Im Datenportal werden die Daten in Kanalgruppen („Groups“) und Datenkanäle („Channels“) sortiert. Die Kanalgruppe ist in diesem Fall „Rotorversuch“, die Datenkanäle sind die Einträge, die sich innerhalb dieser Gruppe befinden, wie zum Beispiel „PXI Slot2\ai0 (Filtered)“ (siehe Abbildung 3.41).
Unterhalb des Datenportals befindet sich noch ein weiteres Fenster. Dieses gibt Informationen zu dem jeweils ausgewählten Kanal, wie zum Beispiel den Namen, die Länge oder Maximum bzw. Minimum des Kanals. Darunter gibt es eine Vorschau, wie der Graph dieses Kanals aussieht.

Als nächstes gibt es das Modul „VIEW“ (siehe Abbildung 3.42). In dem Modul „VIEW“ ist es möglich, die Daten, die man in das Datenportal geladen hat, zu sichten. Wie in Abbildung 3.42 zu sehen ist, können die Daten entweder als Graph oder als Tabelle angezeigt werden.
Des Weiteren gibt es in diesem Modul viele Möglichkeiten, die Daten zu analysieren, wie zum Beispiel den Graphen zoomen, Extremwerte finden und kennzeichnen sowie Werte aus den Tabellen löschen, hinzufügen oder ändern.

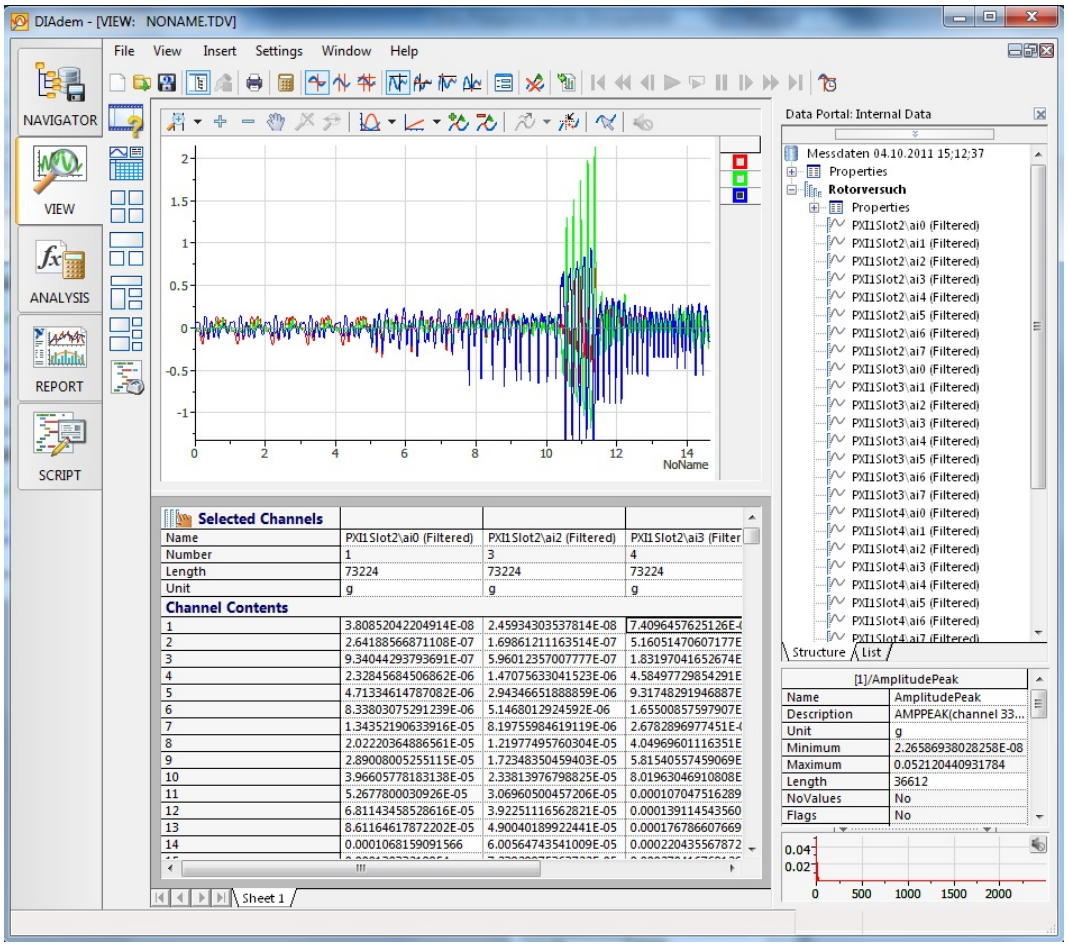


Abbildung 3.42.: Das Modul „DIAdem-VIEW“, Autorin: Melanie Schulze

In dem Modul „ANALYSIS“ (siehe Abbildung 3.43) ist es möglich, die Daten mathematisch zu analysieren. Für die Analyse stehen viele Funktionen zur Verfügung, welche in verschiedene Bereiche unterteilt sind, wie zum Beispiel „Basismathematik“, „Signalanalyse“ oder „Statistik“. Es ist beispielsweise möglich, den Mittelwert einer Funktion zu berechnen, eine Kurve neu zu skalieren, eine FFT zu berechnen oder Maximalwerte einer Funktion zu suchen.
Die Ergebnisse einer jeweiligen Berechnung werden dann in einem neuen Kanal gespeichert. So enthält der Kanal „AmplitudePeak“ das Ergebnis der Berechnung der FFT und der Kanal „LinearScaled“ die neu skalierte Kurve (siehe Abbildung 3.43).

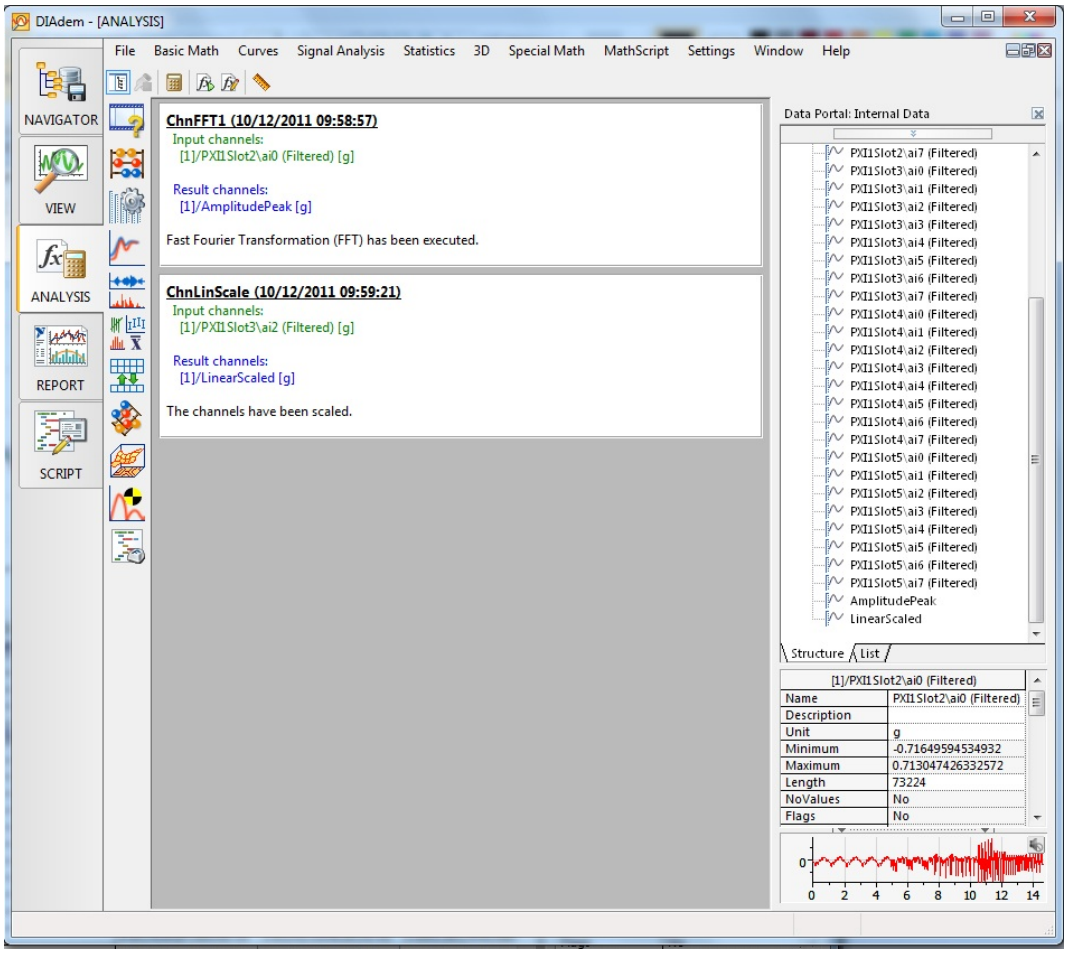


Abbildung 3.43.: Das Modul „DIAdem-ANALYSIS“, Autorin: Melanie Schulze

Des Weiteren gibt es das Modul „REPORT“ (siehe Abbildung 3.44). Dieses Modul ist dazu da, Berichte zu erstellen. In diesen Berichten können Ergebnisse zuvor gemachter Analysen dargestellt werden. Es besteht die Möglichkeit, zwei- oder dreidimensionale Koordinatensysteme, Texte, Grafiken oder Tabellen in einen Bericht einzubinden. Außerdem können Layouts erstellt werden. Diese werden unabhängig von den Daten gespeichert und können später auf andere Datensätze angewandt werden. Dieses ist vorteilhaft, wenn man ähnliche Daten immer auf die gleiche oder zumindest ähnliche Weise darstellen möchte. Genau dieses ist der wichtige Nutzen von *DIAdem*, der in dieser Bachelorarbeit angewendet wird, denn die spätere Auswertung der Sensorsignale soll immer auf die gleiche Weise erfolgen.

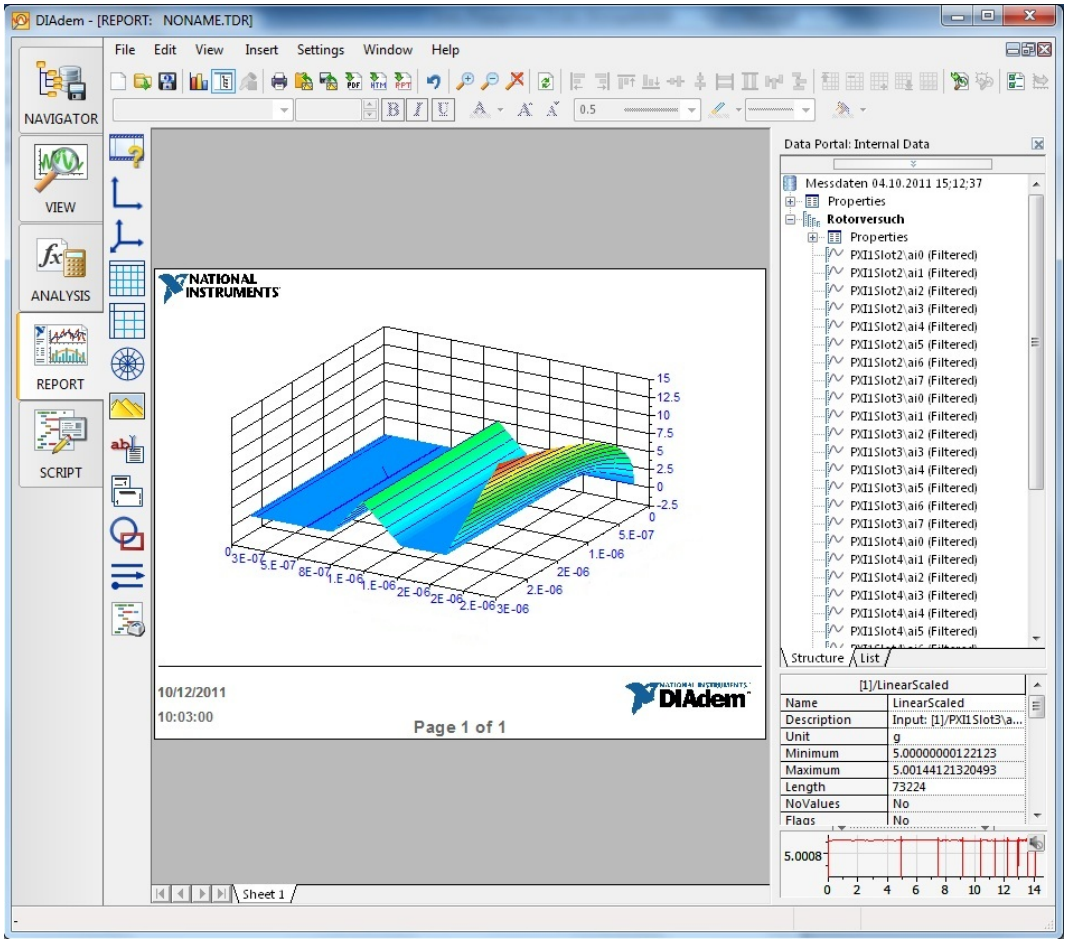


Abbildung 3.44.: Das Modul „DIAdem-REPORT“, Autorin: Melanie Schulze

Abschließend sei das Modul „SCRIPT“ erwähnt (siehe Abbildung 3.45). Hierbei handelt es sich um eine Programmieroberfläche. Es ist möglich, mit Hilfe der Programmiersprache „Visual Basic Script“ Abläufe zu automatisieren. Beispielsweise gibt es Befehle für alle Funktionen, die auch das Modul „ANALYSIS“ bietet oder dafür, Ergebnisse in einem Report darzustellen. Außerdem können unter anderem Kanäle gelöscht oder verschoben werden, neue Kanäle bzw. neue Gruppen erstellt oder Mitteilungen an den Benutzer ausgeben werden. Gestartet wird das Programm mit dem Button „Run Script“ (grünes Dreieck, siehe Abbildung 3.45) oder mit der Tastenkombination „Strg + F5“.

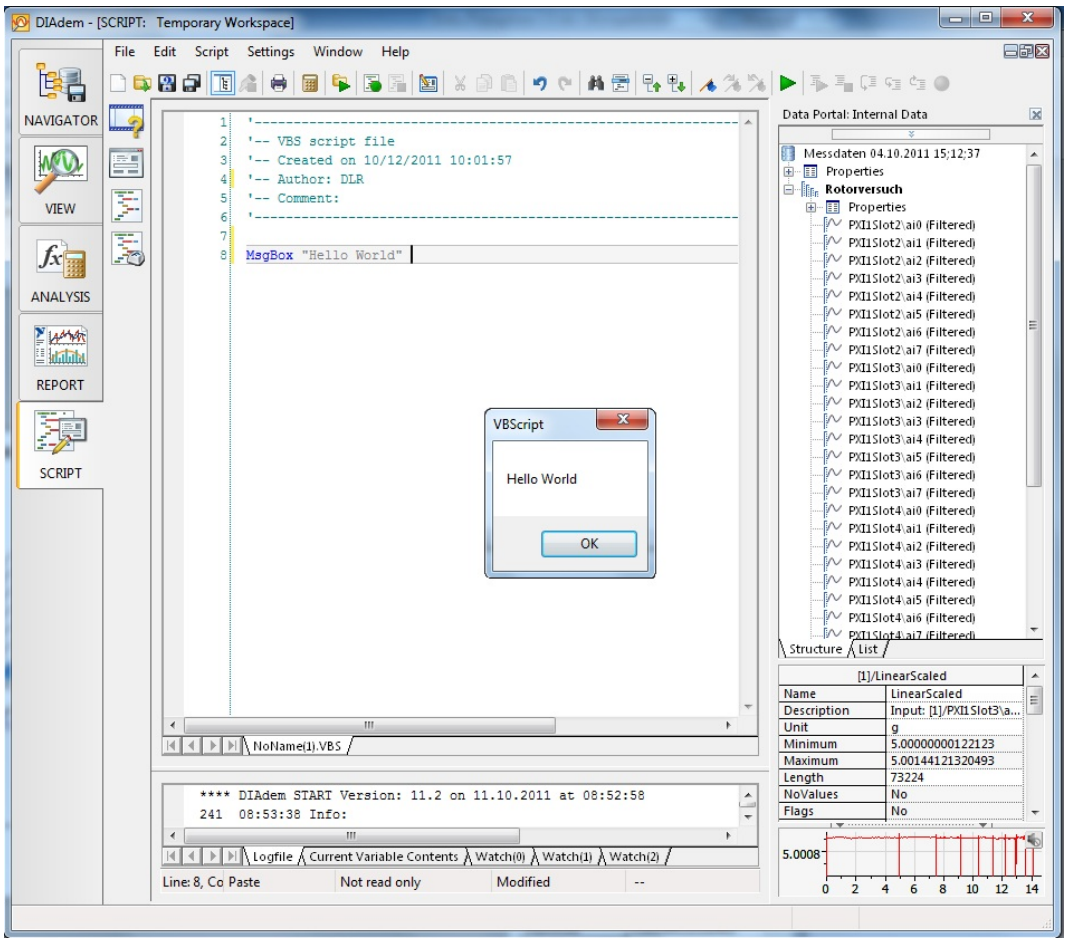


Abbildung 3.45.: Das Modul „DIAdem SCRIPT“, Autorin: Melanie Schulze

4. Ablaufplan

In diesem Kapitel wird der aufgestellte Ablaufplan erläutert.
Die blauen Balken stehen hierbei jeweils für die geplante Zeit und die roten Balken stellen dar, wie viel Zeit für die jeweilige Teilaufgabe tatsächlich gebraucht wurde.

Dieser Ablaufplan ist in Abbildung 4.1 dargestellt, dazu sollen im Folgenden zunächst die verwendeten Zahlen erklärt werden. Dabei werden als erstes die blauen Zahlen (geplante Zeit) und anschließend die roten Zahlen (tatsächliche Zeit) erläutert.

(1): Einrichten der Arbeitsplätze inklusive Installieren der Software
(2): Einarbeiten in das *PXI*-Messsystem
(3): Einarbeiten in die Beschleunigungs-Sensoren
(4): Einarbeiten in *LabVIEW*
(5): Einarbeiten in *DIAdem*
(6): Programmierung in *LabVIEW*
(7): Programmierung in *DIAdem*
(8): Programmtest an Hardware-Beispielaufbau
(9): Weitere Programmierung in *LabVIEW*
(10): Weitere Programmierung in *DIAdem*
(11): Testphase
(12): Schreiben der Dokumentation und Erstellen der Präsentation

(1): Einrichten der Arbeitsplätze inklusive Installieren der Software
(2): Fertigung von Hardware-Komponenten
(3): Einarbeiten in die Beschleunigungs-Sensoren und Funktionstest mit einem Rütteltisch
(4): Einarbeiten in das *PXI*-Messsystem
(5): Einarbeiten in *LabVIEW* sowie Verbinden von *LabVIEW* mit der Hardware
(6): Programmierung in *LabVIEW*
(7): Programmtest an Shaker mit Feder-Masse-Pendel
(8): Programmierung in *LabVIEW*
(9): Einarbeiten in *DIAdem* und Programmierung in *DIAdem*
(10): Programmtest und -erweiterung an Shaker mit Feder-Masse-Pendel
(11): Weitere Programmierung in *LabVIEW* und *DIAdem* inklusive Programmtest
(12): Weitere und abschließende Programmtests und -verbesserungen
(13): Schreiben der Dokumentation
(14): Erstellen der Präsentation

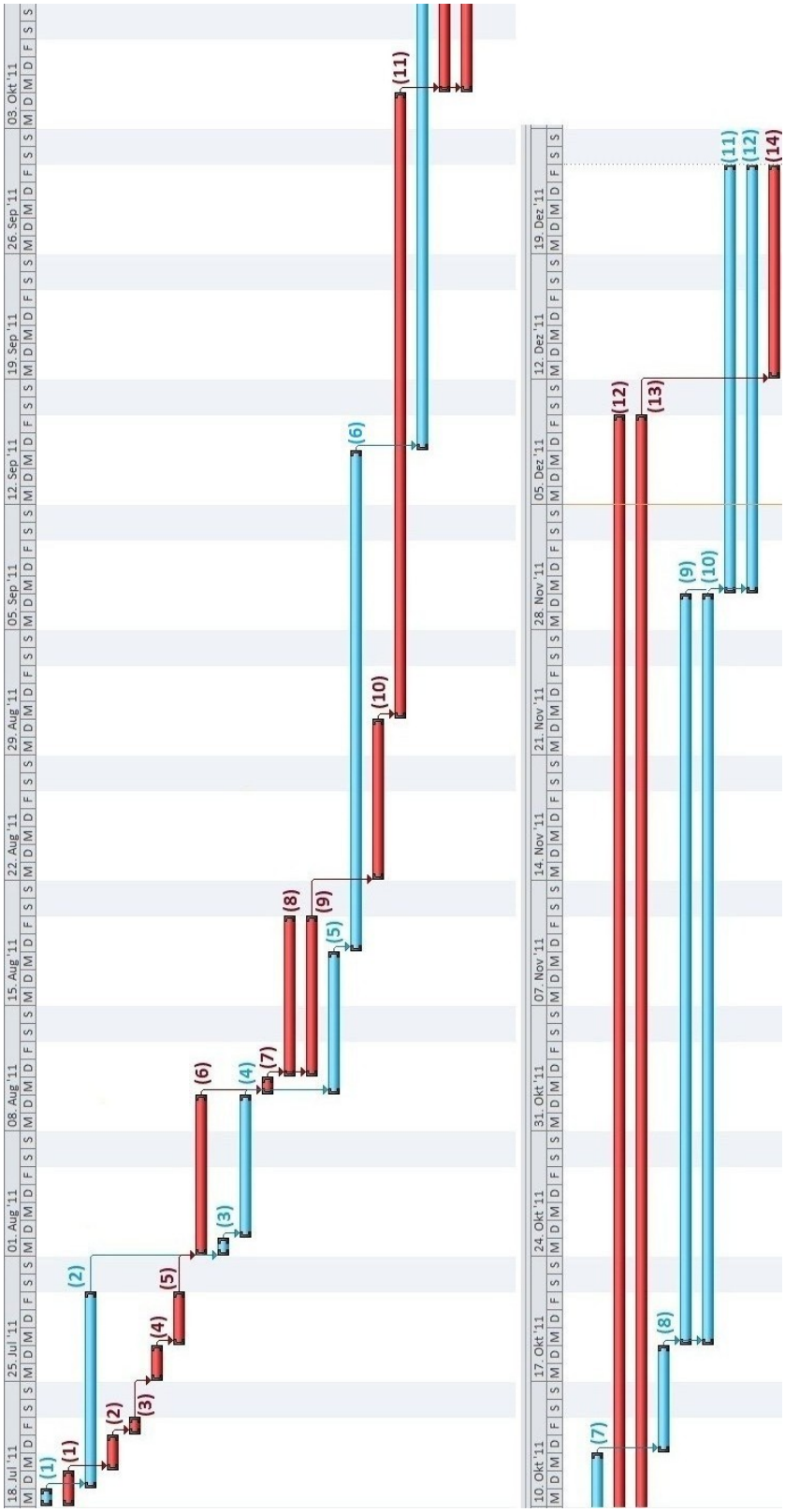


Abbildung 4.1.: Ablaufplan, Autorin: Melanie Schulze

An dem Ablaufplan ist gut zu erkennen, dass der geplante Zeitraum insgesamt eingehalten wurde. Die einzelnen Abschnitte sind allerdings anders als geplant verlaufen.
Beispielsweise war anfangs beabsichtigt, die Programme gemeinsam zu erstellen, weshalb die Punkte (6) und (7) nacheinander angeordnet sind. Es stellte sich jedoch schnell heraus, dass die beiden Programme einzeln und daher parallel erstellt werden können. Da es sich hier um eine Gruppenarbeit handelt, ist diese mögliche Aufteilung besonders vorteilhaft und erspart zusätzlich viel Zeit.
Des Weiteren fällt auf, dass die Testphase viel kürzer geplant war, dafür die Programmierung länger. Die Grundgerüste der Programme standen allerdings schon recht früh. Es musste jedoch Vieles immer neu angepasst werden. Dieses erforderte wiederum jeweils eine neue Testphase. Dadurch ergaben sich immer wieder Programmerweiterungen und Testphasen. Allerdings konnte nebenbei schon mit der Erarbeitung der Dokumentation begonnen werden.

5. Testaufbau

Um die Funktion des Programms sowie der Beschleunigungssensoren zu testen, wird folgender Versuchsaufbau verwendet (siehe Abbildung 5.1).

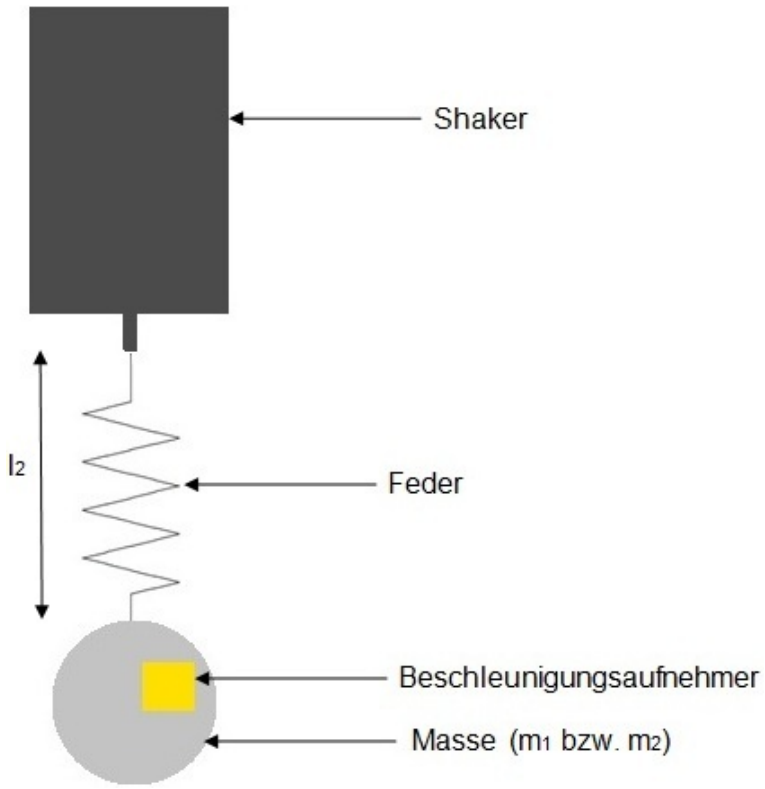


Abbildung 5.1.: Modell des Testaufbaus, Autorin: Melanie Schulze

Dieser Aufbau wird zunächst verwendet, um nicht gleich am „echten“ Modell zu testen. Dies hat den Vorteil, dass die Ergebnisse bekannt sind und nachvollzogen werden kann, ob das Programm richtig funktioniert. An dem „echten“ Modell wäre dieses Nachvollziehen nicht gegeben gewesen.

Bei dem Versuchsaufbau handelt es sich um ein Feder-Masse-Pendel, also um einen harmonischen Oszillator. Lenkt man das Massestück aus der Ruhelage aus, beginnt die harmonische Schwingung. Es wirkt eine Kraft entgegen der Auslenkungsrichtung (die elastische Kraft). Diese ist proportional zur Auslenkung der Feder (Hookesches Gesetz) und hängt des Weiteren von der Federkonstante D ab. Dadurch hat eine ausgelenkte Feder

immer das Bestreben, in ihre Ausgangslage zurückzukehren und wird zur Ruhelage hin beschleunigt. Beim Durchlaufen der Ruhelage bewirkt die Trägheit der Masse, dass das Pendel sich weiter über die Ruhelage hinaus bewegt. Da die elastische Kraft proportional zur Auslenkung ist, wird sie immer größer, je weiter sich die Masse von der Ruhelage entfernt. Ist sie so groß, dass die Trägheit überwunden wird, wird die Masse wieder in Richtung der Ruhelage beschleunigt. Indem sich dieser Vorgang periodisch wiederholt, entsteht die harmonische Schwingung.
Da das Feder-Masse-Pendel kontinuierlich von dem Shaker angeregt wird, entstehen erzwungene Schwingungen. Wenn nun der Shaker das Pendel mit seiner Eigenfrequenz anregt, wird die Amplitude der Schwingung theoretisch immer größer und es kommt im schlimmsten Fall zur Resonanzkatastrophe (vgl. [29]).

Um die Funktion des Programms zu testen, wird das Feder-Masse-Pendel mit einem Frequenz-Sweep angeregt. Dabei soll festgestellt werden, bei welcher Frequenz die reale Resonanz liegt, um diese mit der berechneten vergleichen zu können. Hierbei wird mit zwei verschiedenen Massen gearbeitet, um das Ergebnis noch einmal mit einer anderen Masse verifizieren zu können.

Gemessene Größen:

- Federlänge $l_1 = 3{,}3$ cm
- Federlänge nach Auslenkung mit m_1: $l_2 = 6{,}4$ cm
- Masse $m_1 = 5{,}16$ kg
- Masse $m_2 = 2{,}10$ kg

Die Eigenfrequenz des Feder-Masse-Pendels wird folgendermaßen berechnet (siehe (5.1) und (5.2)), wobei der Einfachheit halber von einer idealen, also masselosen Feder, ausgegangen wird.

$$F = m_1 \cdot g = D \cdot \Delta l \quad \Rightarrow \quad D = \frac{m_1 \cdot g}{l_1 - l_2} = \frac{5{,}16\ kg \cdot 9{,}81\ \frac{m}{s^2}}{(6{,}4 - 3{,}3) \cdot 10^{-2}\ m} \approx 1632{,}89\ \frac{N}{m} \tag{5.1}$$

$$\omega_1 = \sqrt{\frac{D}{m_1}} = \sqrt{\frac{1632{,}89\ \frac{N}{m}}{5{,}16\ kg}} \approx 17{,}79\ \frac{1}{s} \quad \Rightarrow \quad f_1 = \frac{\omega_1}{2\pi} \approx \underline{\underline{2{,}38\ Hz}} \tag{5.2}$$

$$\omega_2 = \sqrt{\frac{D}{m_2}} = \sqrt{\frac{1632{,}89\ \frac{N}{m}}{2{,}1\ kg}} \approx 27{,}88\ \frac{1}{s} \quad \Rightarrow \quad f_2 = \frac{\omega_2}{2\pi} \approx \underline{\underline{4{,}44\ Hz}}$$

In den Versuchen ermittelte Resonanzfrequenzen:

$$f_1 \approx 3,1\ Hz$$

$$f_2 \approx 5,3\ Hz$$

Die gemessenen Resonanzfrequenzen weisen Abweichungen von etwa 30 % bzw. 20 % von den berechneten Werten auf. Die Abweichungen lassen sich dadurch erklären, dass in der Berechnung von einer komplett starren Aufhängung ausgegangen wurde. In der Realität schwingt diese allerdings auch, was Einfluss auf die Resonanzfrequenz hat, denn die zusätzliche Weichheit ändert die Federkonstante.
Einen kleinen Einfluss auf die Abweichungen haben auch Messungenauigkeiten sowie Rundungsfehler in der Rechnung.

6. Das *TDMS*-Dateiformat

In dem *LabVIEW*-Programm werden die Messdaten in eine Datei geschrieben (siehe Kapitel 7). Das Dateiformat dieser Datei heißt *TDMS*. Dieses Dateiformat wurde von *NI* unter anderem dafür entwickelt, Messdaten und simulierte Daten mit Zusatzinformationen zu versehen. Dieses soll dafür sorgen, dass Messdaten effizient und organisiert gespeichert werden.
Das Wichtigste der *TDMS*-Datei ist ihr Aufbau: Datei, Gruppe, Kanal (siehe Abbildung 6.1). Eine Datei kann dabei mehrere Gruppen enthalten und eine Gruppe wiederum mehrere Kanäle. Durch diese Aufteilung ist es möglich, Daten verständlich zu sortieren.

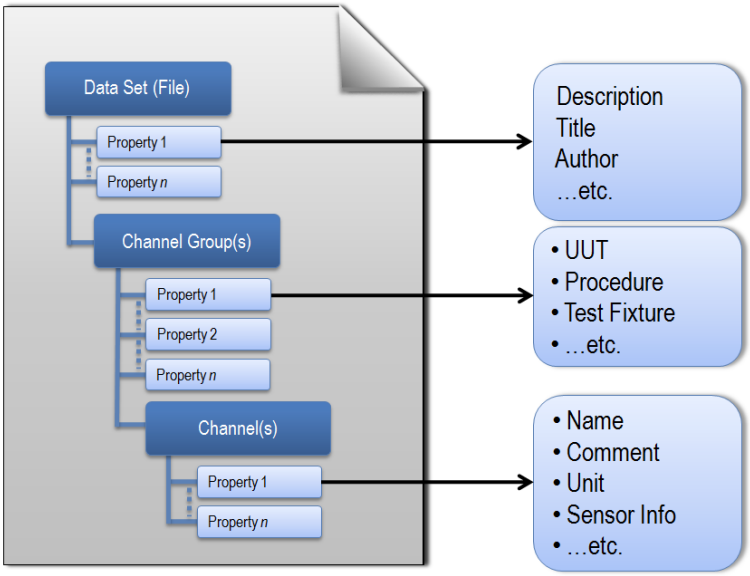


Abbildung 6.1.: Aufbau des *TDMS-Dateiformats*, Quelle: [17]

In Abbildung 6.1 ist weiterhin zu erkennen, dass in jeder Ebene Zusatzinformationen („Properties“), wie zum Beispiel ein Titel oder der Autor, gespeichert werden können.
Ein weiterer Vorteil der *TDMS*-Datei ist, dass diese automatisch eine „*.tdms_index“-Datei erzeugt, welche „Informationen zu allen Attributen und Pointern der Massendatendatei“ (aus [16]) enthält. Dadurch wird der Zugriff auf die Daten großer Datensätze beschleunigt.

7. Das *LabVIEW*-Programm

Einen kurzen Einstieg in die Software *LabVIEW* sowie erste Schritte zur Implementierung der Messkarten bietet das Kapitel 3.2.2. Im Folgenden wird auf das *LabVIEW*-Programm eingegangen.

7.1. Hauptprogramm

Als erstes wird das gesamte Programm im Zusammenhang erläutert, um später detailliert auf die Unterprogramme einzugehen. Es wird zuerst die Bedienoberfläche (Frontpanel) und danach der Programmcode (Blockdiagramm) erläutert.

7.1.1. Erklärung der Bedienoberfläche

Die Bedienoberfläche ist in Abbildung 7.1 zu sehen. Diese wird zuerst erklärt, da das den Einstieg in den Programmcode erleichtert.

Im oberen Teil befindet sich eine Signalleuchte „Ende der gesamten Messung“. Diese dient dazu, dem Benutzer anzuzeigen, dass die Messung beendet ist. Neben dieser Signalleuchte befindet sich ein „Stop“-Button. Dieser dient dazu, die gesamte Messung vorzeitig zu beenden. Wird die Messung vorzeitig beendet, kann *DIAdem* die Daten jedoch nicht korrekt auswerten.
Darunter ist auf der linken Seite die Steuerungseinheit, die aus drei Seiten besteht, zu sehen (siehe Abbildungen 7.2 bis 7.4).

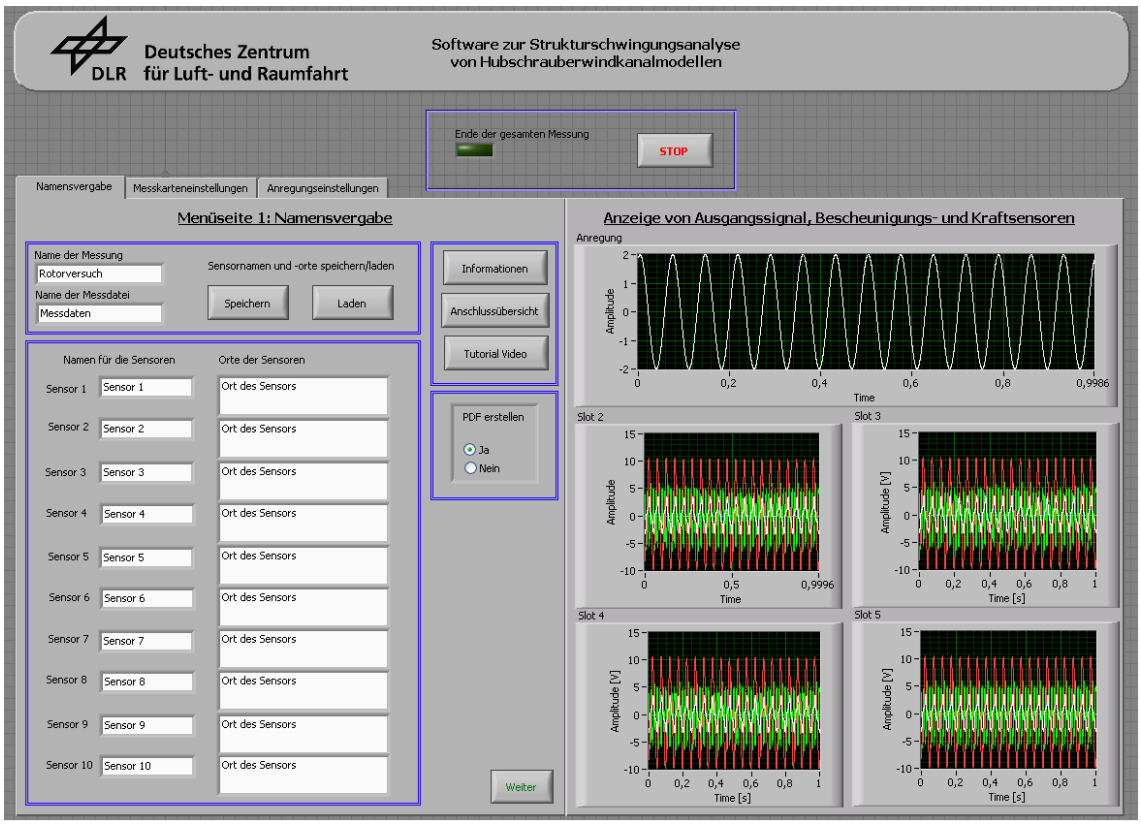


Abbildung 7.1.: Bedienoberfläche, Autor: René Pfeifer

Auf der ersten Menüseite (siehe Abbildung 7.2) kann der *TDMS*-Datei ein Name gegeben und ein Titel für das spätere Deckblatt der *DIAdem*-Auswertung angegeben werden. Des Weiteren kann für jeden Sensor ein Name sowie eine Ortsbeschreibung eingegeben werden.

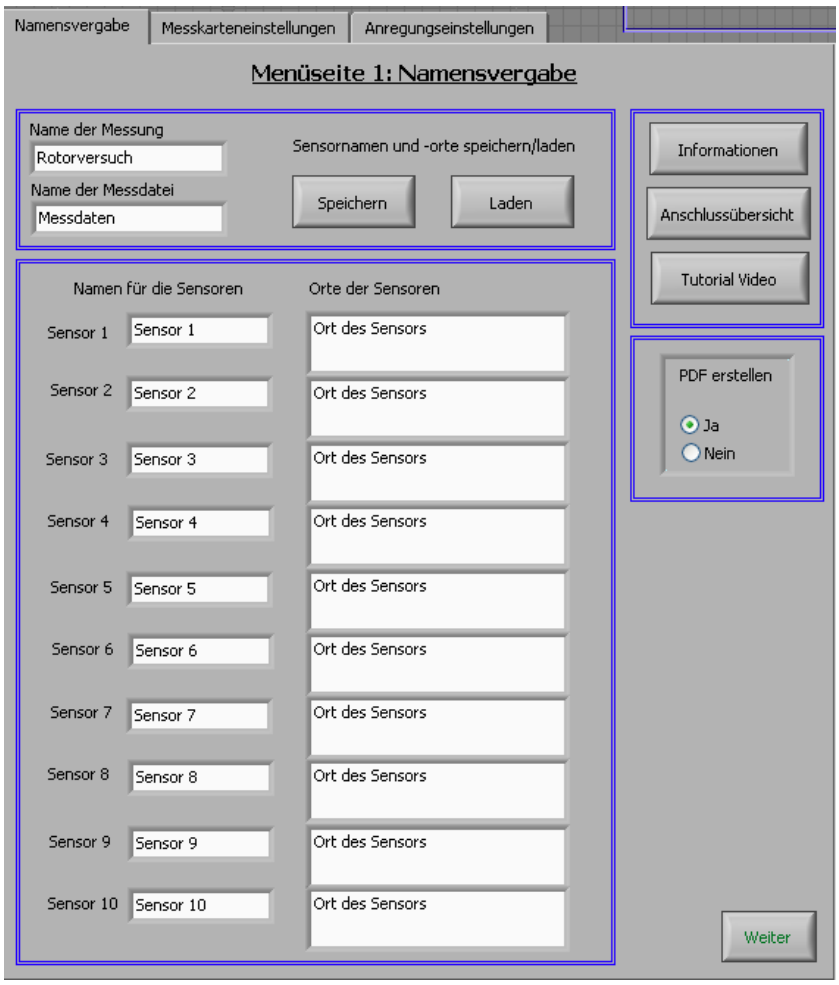


Abbildung 7.2.: Menüseite 1, Autor: René Pfeifer

Die Menüseite 2 (siehe Abbildung 7.3) dient der Festlegung der Anzahl der Sensoren und der Einstellung der Kartenparameter. So können hier mit der Aktivierung der Buttons „Slot 2“ bis „Slot 6“ Einstellungen der jeweiligen Messkarte vorgenommen werden.

Einzustellen sind:

Minimal-/Maximalwert: Hier wird der minimale und maximale Wert eingetragen, den die Messkarte erfassen soll/kann. Die „PXI-4461“ hat einen Bereich von $\pm$ 316 mV bis 42.4 V und die „PXI-4472B“ von $\pm$ 10 V.

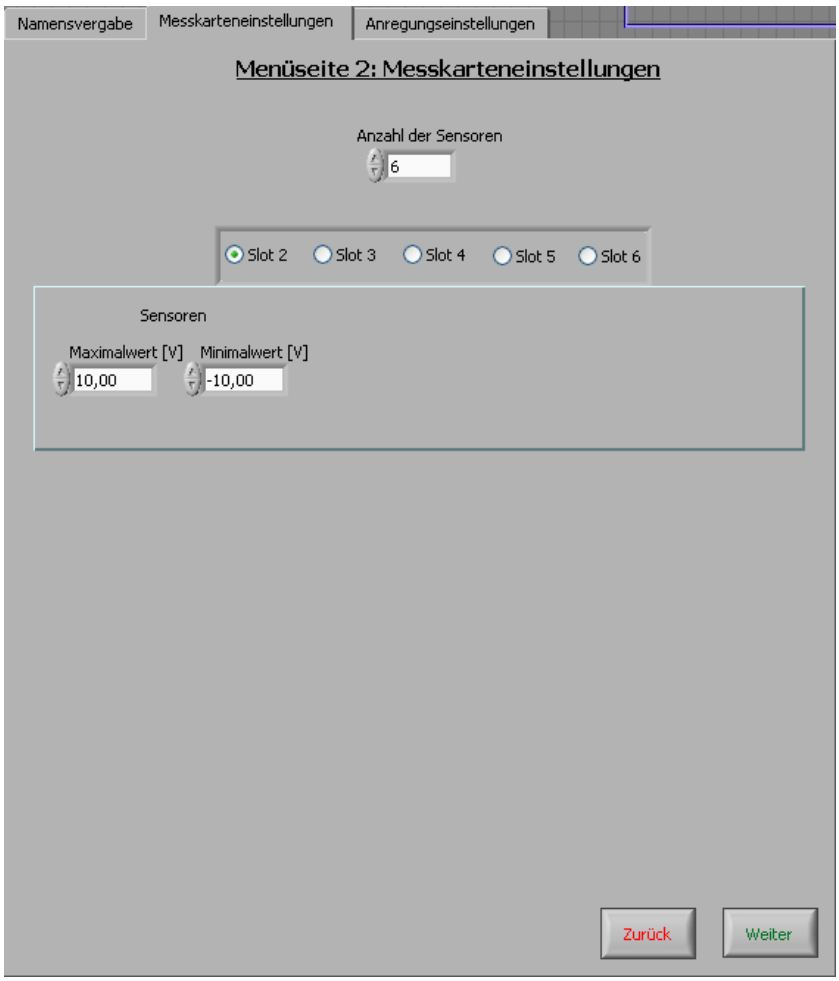


Abbildung 7.3.: Menüseite 2, Autor: René Pfeifer

Auf der letzten Menüseite (siehe Abbildung 7.4) werden die Samplerate[2], die Rampenansteigszeit für das periodische Signal und die „Charge Amplifier“-Einstellungen für die Messung festgelegt.
Danach kann das anregende Signal gewählt werden. Zur Auswahl stehen „Periodisches Signal“ und „Rauschen“.
Bei dem periodischen Signal stehen die Signaltypen Sinus, Dreieck, Rechteck und Sägezahn zur Auswahl. Dort können die Amplitude, der Offset, die Schrittweite, die Start- und Endfrequenz sowie die „Messzeit“ oder „Periodenanzahl“ eingestellt werden. Bei der „Messzeit“ werden bei jeder Frequenz die Sensordaten für diese Zeit eingelesen. Durch die Wahl „Periodenanzahl“ wird für jede Frequenz die Aufnahmezeit berechnet (siehe Erklärung unten).
Das „Rauschen“ ist ein weißes Rauschen, bei dem das RMS-Level, der Offset, die Grenzfrequenz und die maximal im Rauschsignal enthaltene Frequenz eingestellt werden kann.

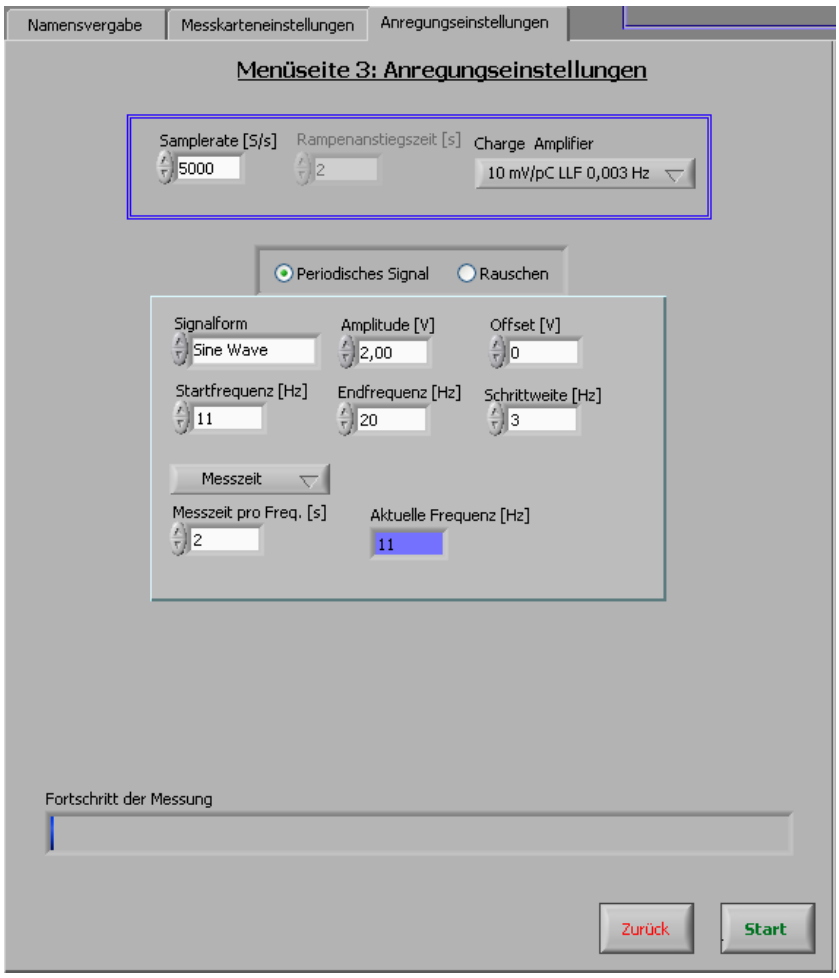


Abbildung 7.4.: Menüseite 3, Autor: René Pfeifer

Erklärung der Periodenanzahl:

Die Periodenanzahl bezieht sich auf die Frequenz des anregenden Signals. Das bedeutet, dass wenn das anregende Signal mit 1 Hz schwingt, von diesem fünf Perioden aufgenommen werden sollen und die Samplerate[2] auf 5000 $\frac{S}{s}$ eingestellt ist, werden die Sensordaten fünf Sekunden lang eingelesen. Bei der nächsten Frequenz, z.B. 2 Hz, dauert die Aufnahme nur noch 2,5 Sekunden. Dies lässt sich durch folgende Rechnung erklären.

Zuerst wird die Anzahl an Samples[3], die für die gewünschte Anzahl an Perioden aufgenommen werden muss, berechnet (siehe (7.1)).

$$Samples\ [S] = \frac{Samplerate\left[\frac{S}{s}\right]}{Frequenz\ [Hz]} \cdot Periodenanzahl \tag{7.1}$$

Mit dieser Anzahl an Samples[3] kann die benötigte Erfassungszeit der Daten berechnet werden (siehe (7.2)).

$$Zeit\ [s] = \frac{Samples\ [S]}{Samplerate\left[\frac{S}{s}\right]} \tag{7.2}$$

Rechts von der Steuerungseinheit befinden sich die Anzeigen der ausgegebenen und erfassten Signale.
In Abbildung 7.5 ist im oberen Teil das ausgegebene Signal, welches aus der Messkarte „PXI-4461“ kommt, zu sehen. Darunter sind die Anzeigen „Slot 2“, „Slot 3“, „Slot 4“ und „Slot 5“. In diesen Slots im *PXI*-Chassis befinden sich die Messkarten vom Typ „PXI-4472B“. Die Karten erfassen alle Sensordaten der Beschleunigungssensoren sowie die des Kraftsensors.

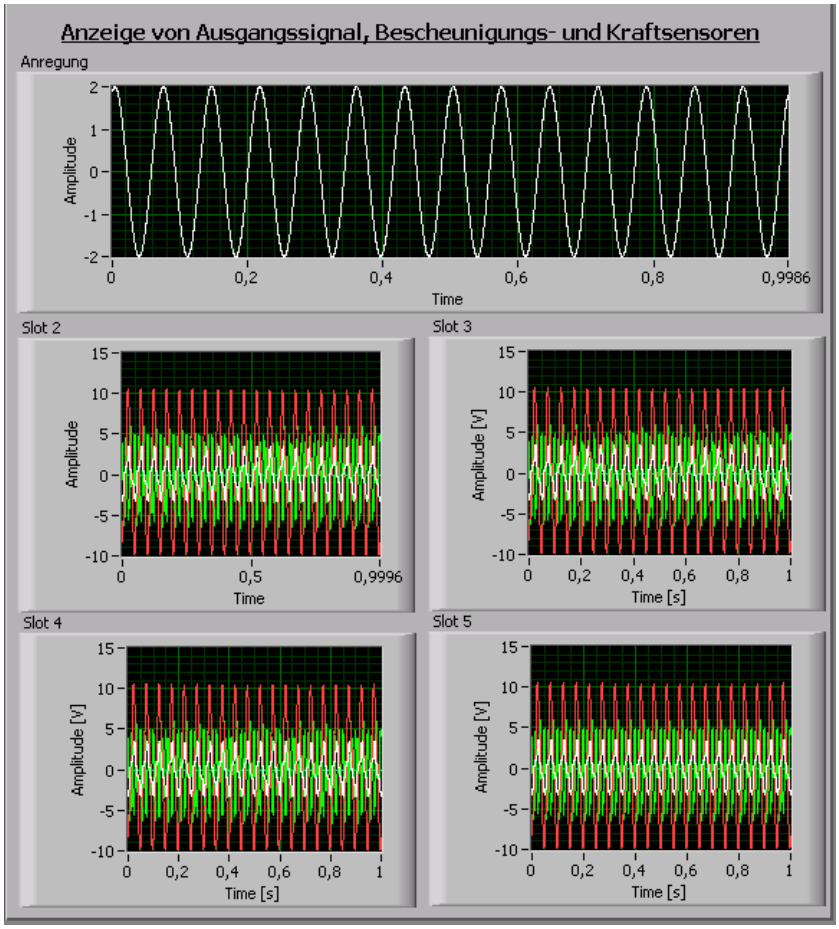


Abbildung 7.5.: Anzeige der Signale, Autor: René Pfeifer

7.1.2. Schematischer Aufbau des Programmcodes (Blockdiagramm)

Im Folgenden wird das gesamte Programm (siehe Anhang G) zusammenhängend erläutert. Dazu ist es auch hier notwendig, es in Teilabschnitten zu zeigen und zu erläutern, um die Verständlichkeit und Nachvollziehbarkeit des großen Programms zu gewährleisten.
Das Grundgerüst des gesamten Programms ist in Abbildung 7.6 zu sehen. Es ist zu erkennen, dass sich um das gesamte Programm eine While-Schleife befindet, die für ein wiederholtes Durchlaufen des Programms sorgt. In dieser While-Schleife befindet sich eine Sequenzstruktur. Diese sorgt dafür, dass der Programmcode von links nach rechts verarbeitet wird. Zudem wird mit Hilfe dieser Struktur die Lesbarkeit gesteigert.

Erklärung der Programmabschnitte in Abbildung 7.6:

1.: Hier werden die Startinitialisierungen für das Programm verarbeitet.
2.: In diesem Abschnitt werden die Initialisierungen durchlaufen, welche bei jeder neuen Messung initialisiert werden müssen.
3.: Die gesamten Menüfunktionen und Einstellungen der zukünftigen Messung werden hier verarbeitet und getroffen.
4.: In diesem Punkt werden alle nötigen Einstellungen parametriert und gespeichert, die vor dem eigentlichem Start der Messung abgearbeitet werden können. Dies bezweckt, dass nur das Nötigste für das Einlesen und Ausgeben von Signalen in dem nächsten Abschnitt verarbeitet wird.
5.: In diesem Abschnitt findet das gesamte Einlesen und Ausgeben von Signalen statt.
6.: Hier werden noch laufende Programmteile beendet.
7.: Dieser Teil dient der Kommunikation zwischen *LabVIEW* und *DIAdem*.
8.: Hier besteht die Möglichkeit, eine neue Messung zu starten oder das Programm zu beenden.

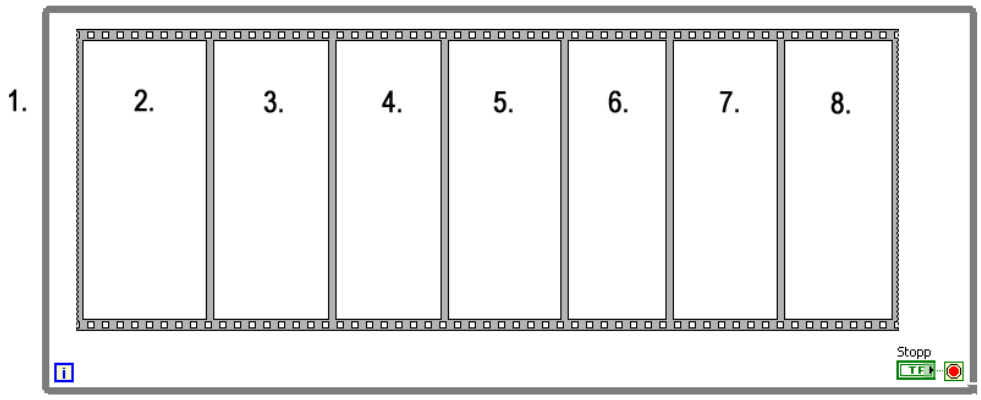


Abbildung 7.6.: Grobes Schema des Grundgerüstes (Blockdiagramm), Autor: René Pfeifer

Die einzelnen Programmabschnitte werden im Folgenden genauer erklärt.

7.1.3. Programmabschnitt 1

Dieser Programmabschnitt dient dazu, Voreinstellungen, die einmalig vorgenommen werden müssen, zu initialisieren (siehe Abbildung 7.7). Aus diesem Grund befinden sich diese Blöcke außerhalb der While-Schleife.

Erklärung des Programmabschnitts in Abbildung 7.7:

Zuerst wird bei der Reiterkarte „Auswahlmenü“ die Startseite „Namensvergabe“, die Radio Buttons auf den „Slot 2“ und „Rauschen aktivieren“ auf „Wellenform“ eingestellt. Darunter sind die Reiterkarten zu sehen. Diese sind für das Wechseln der Menüseiten oder der Slotparametereinstellungen nötig. Zudem wird die Anzeige „Aktuelle Frequenz“ deaktiviert („disabled“), damit der Bediener dort keine Eingabe tätigen kann.

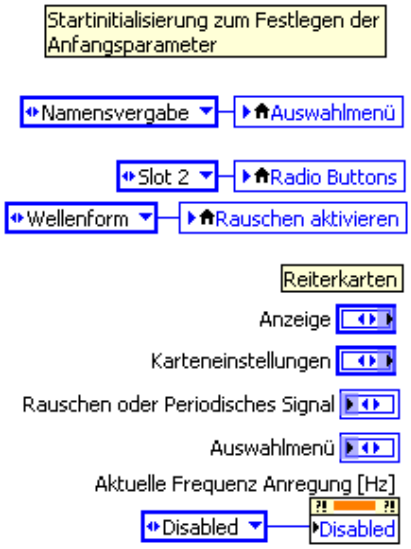


Abbildung 7.7.: Abschnitt 1 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.1.4. Programmabschnitt 2

In der ersten Sequenz des Sequenzdiagramms werden Initialisierungen ausgeführt, welche zu Beginn jeder Messung gesetzt werden müssen (siehe Abbildung 7.8).

Erklärung des Programmabschnitts in Abbildung 7.8:

Als erstes werden die „Initialisierung“ und „Rampenzeit mit einbeziehen“ auf „True“ gesetzt. „Initialisierung“ wird auf „True“ gesetzt, damit bestimmte Programmstellen später nur beim ersten Schleifendurchlauf der inneren While-Schleife im Programmabschnitt 5 (siehe Abbildung 7.11) berücksichtigt werden. „Rampenzeit mit einbeziehen“ dient dazu, dass die vom Benutzer eingegebenen Rampenzeit zur Aufnahmezeit der ersten anregenden Frequenz („Startfrequenz“) addiert wird. Dies ist notwendig, damit die vom Bediener

eingegebene Aufnahmezeit eingehalten wird, weil die Rampe eine bestimmte Zeit benötigt und danach erst das Anregungssignal ausgegeben wird.
Zudem wird „Frequenz erhöht“ und „Ende der gesamten Messung“ auf „False“ gesetzt. Diese zwei Größen werden immer zu Anfang auf „False“ gesetzt, damit gewährleistet ist, dass immer die richtigen Startbedingungen vorliegen.
„Frequenz erhöht“ ist eine intern benötigte Variable. Diese signalisiert, dass bei Anregung mit „Periodisches Signal“ die Messdaten einer Frequenz vollständig erfasst wurden. Hierdurch wird die Frequenz um die Schrittweite erhöht. „Ende der gesamten Messung“ signalisiert dem Benutzer, dass die Messung beendet ist. Die Fortschrittsanzeige wird zudem wieder auf null gesetzt, da eine neue Messung beginnt. Darunter befindet sich das Sub-VI „Bedienelemente aktivieren.vi“ (siehe Abschnitt 7.2.7).

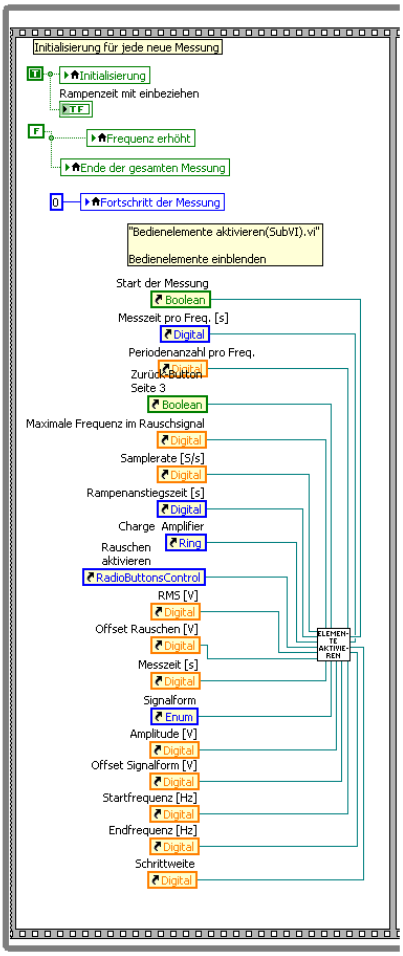


Abbildung 7.8.: Abschnitt 2 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.1.5. Programmabschnitt 3

In diesem Abschnitt werden alle Einstellungen vom Bediener getroffen (siehe Abbildung 7.9). Die gesamten Einstellungen befinden sich in einer While-Schleife, welche erst verlassen

wird, wenn der Bediener den „Start“-Button betätigt hat und die Eingaben korrekt sind. Durch das Verlassen der While-Schleife wird der nächste Programmabschnitt (siehe 7.1.6) automatisch durchlaufen.

Erklärung der Programmabschnitte in Abbildung 7.9:

1.: Im oberen linken Teil der While-Schleife befindet sich die gesamte Menüsteuerung (vgl. Abschnitt 7.2.3).
2.: Hier wird das Bedienelement „positiver Anteil in % (Rechtecksignal)“ nur eingeblendet, wenn auch die Signalform „Rechtecksignal“ ausgewählt ist, ansonsten ist es ausgegraut (vgl. Abschnitt 7.2.12).
3.: Dieser Bereich ist dafür zuständig, bei Betätigung des Buttons „Tutorial Video“, das Tutorial Video im Windows Media Player zu öffnen. Das setzt voraus, dass dieser auf dem Zielrechner installiert ist (vgl. Abschnitt 7.2.8).
4.: In diesem Teil befindet sich die Auswahl, ob mit einem periodischen Signal oder einem weißen Rauschen angeregt werden soll. Zudem werden nicht benötigte Bedienelemente ausgeblendet oder benötigte wieder eingeblendet (vgl. Abschnitt 7.2.9).
5.: Dieser Abschnitt graut das Bedienelement „Schrittweite“ aus, wenn die Start- und Endfrequenz gleich sind (vgl. Abschnitt 7.2.13).
6.: Hier wird bei Betätigung des „Start“-Buttons geprüft, ob die Einstellungen zueinander passen (vgl. Abschnitt 7.2.10).
7.: Dieser Abschnitt dient dem Speichern der Sensornamen und -orte (vgl. Abschnitt 7.2.5).
8.: Dieser Abschnitt dient dem Laden der Sensornamen und -orte (vgl. Abschnitt 7.2.4).
9.: Dieser Teil dient dazu, dass wenn „Messzeit“ (Zeit der Erfassung der Sensordaten pro Frequenz) ausgewählt ist, das Bedienelement zur Eingabe für die Messzeit angezeigt wird. Bei der Auswahl von „Periodenanzahl“ wird das andere Bedienelement ausgeblendet und das zur Eingabe der Periodenanzahl gehörige angezeigt.
10.: Zum Schluss wird die *TDMS*-Datei für die benötigten Werte für *DIAdem* erstellt.

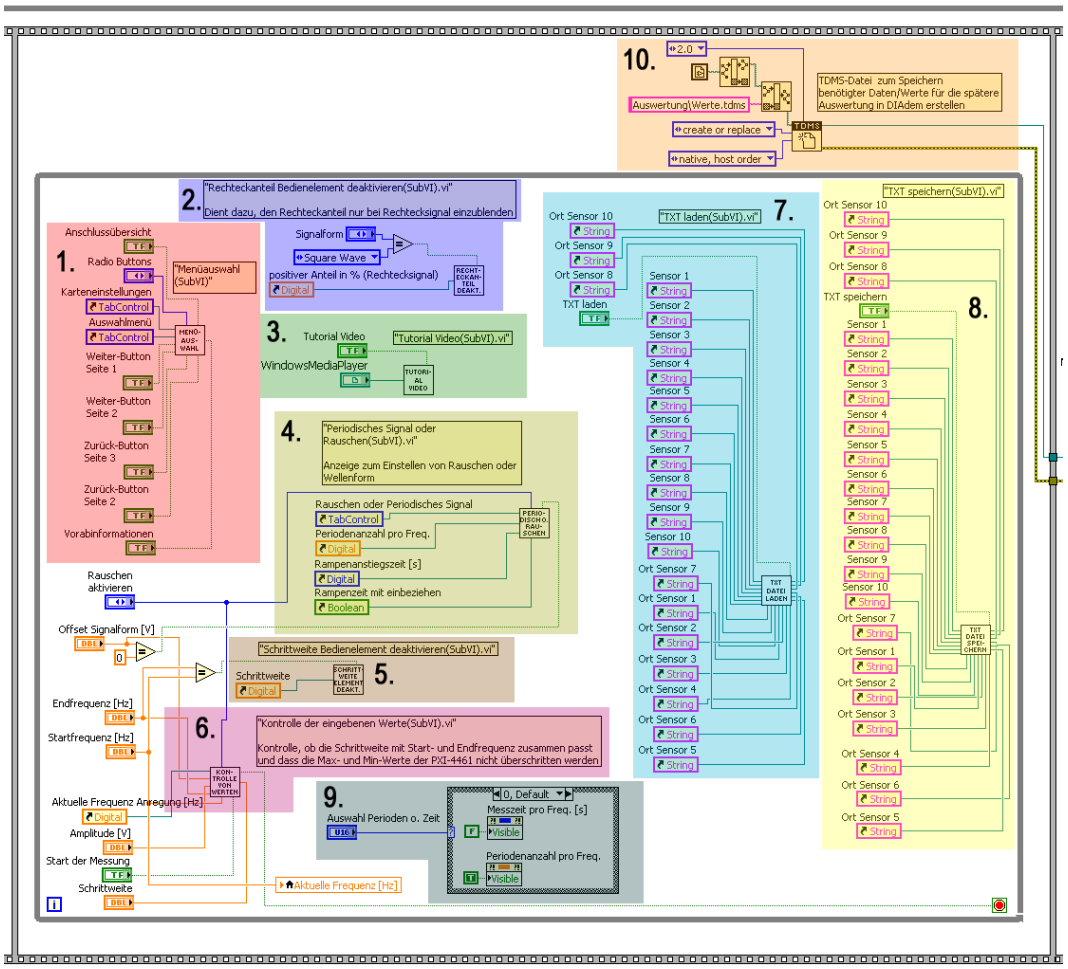


Abbildung 7.9.: Abschnitt 3 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.1.6. Programmabschnitt 4

In diesem Teil stehen alle Einstellungen vom Bediener fest und werden für die spätere Auswertung in *DIAdem* gespeichert. Außerdem werden alle weiteren Parametrierungen und Einstellungen ausgeführt (siehe Abbildung 7.10). Wenn diese abgeschlossen sind, wird die Messung automatisch gestartet.

Erklärung der Programmabschnitte in Abbildung 7.10:

1.: Hier werden die Werte für die „Werte.tdms“-Datei zusammengefasst (vgl. Abschnitt 7.2.21) und in diese geschrieben.

Werte der „Werte.tdms“-Datei:

- „Anzahl der Sensoren“
- „Charge Amplifier“ (beinhaltet eine Zahl von 0 bis 5, welche jeweils eine Einstellung am „Charge Amplifier“ darstellt, siehe Abbildung 8.6)

- „Endfrequenz“
- „Amplitude“
- „Startfrequenz“
- „Schrittweite“
- „Rauschen aktivieren“
- „PDF erstellen“ (*DIAdem*-Auswertung mit oder ohne .pdf-Datei)
- „Maximale Frequenz im Rauschsignal“
- „Datum und Zeit“

2.: Alle Bedienelemente, die während der Messung verändert werden könnten, aber nicht verändert werden dürfen, werden hier deaktiviert (vgl. Abschnitt 7.2.6).
3.: In diesem Teil wird der Name für die *TDMS*-Datei „Name vom Bediener.tdms“ (vgl. Abschnitt 7.2.15) festgelegt. Außerdem wird hier die *TDMS*-Datei erstellt.
4.: Erstellung der Namen der einzelnen Kanäle für die *TDMS*-Datei (vgl. Abschnitt 7.2.19).
5.: Berechnung der Aufteilung des Fortschrittsbalkens. Hier wird der prozentuale Anteil für einen Schritt ermittelt und zudem der Zähler für den Fortschrittsbalken auf null gesetzt. Diese Berechnung wird nur bei Auswahl „Periodisches Signal“ benötigt.
6.: Hier wird die aktuelle Zeit genommen, um sie als Bezugszeit (Startzeit) im weiteren Programm zu nutzen.
7.: In diesem Teil wird die Rampe initialisiert und generiert.
8.: Dieser Block ist zuständig dafür, dass die richtige Grenzfrequenz an das Tiefpassfilter in den Programmabschnitt 5 (siehe Abschnitt 7.1.7) weiter gegeben wird.
9.: Parametrierung des Ausgangskanals für die Ausgabe des Signals (vgl. Abschnitt 7.2.22).
10.: Parametrierung und Start des Eingangskanals für das Einlesen des anregenden Signals zur Triggerung (vgl. Abschnitt 7.2.23).
11.: Hier werden die benötigten Werte zur Signalerzeugung übergeben.
12.: Parametrierung und Start der Eingangskanäle für die Aufnahme der Sensordaten (vgl. auch die Abschnitte 7.2.24, 7.2.25 und 7.2.26). Die Sequenzstruktur dahinter dient dazu, dass zuerst die Slavekarten in den Slots 3, 4 und 5 gestartet werden und dann die Master-Messkarte in Slot 2. Das ist für die Synchronisierung der Karten notwendig. Wenn alle Messkarten, die die Sensordaten erfassen, gestartet sind und die Parametrierung des Eingangskanals für das anregende Signal abgeschlossen ist, wird dieser gestartet. Das hat den Grund, dass so sichergestellt ist, dass die Messkarten für die Sensordaten bereit sind, das Starttriggersignal zu empfangen. Ansonsten kann es passieren, dass die Messkarten für die Sensordaten dieses Signal nicht empfangen.

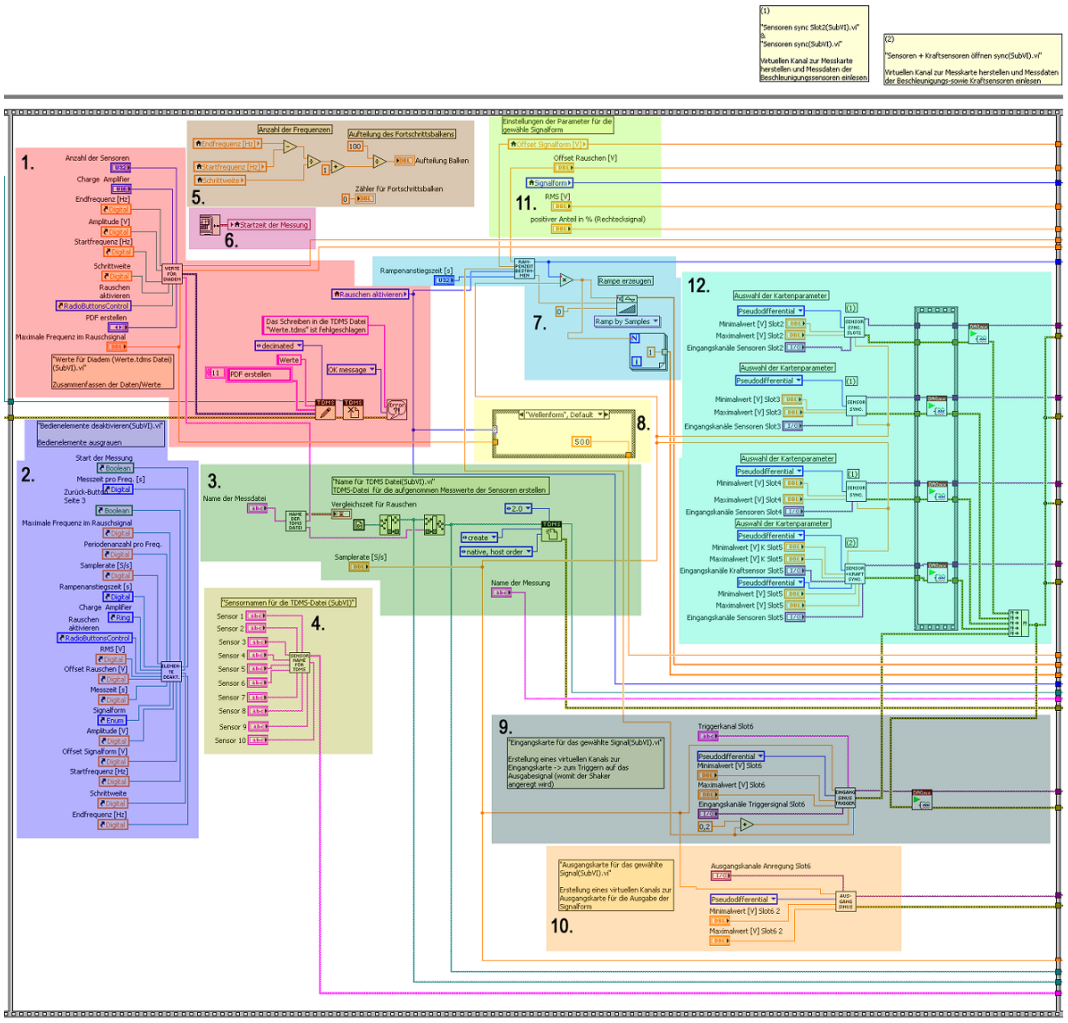


Abbildung 7.10.: Abschnitt 4 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.1.7. Programmabschnitt 5

Hier findet die gesamte Erfassung und Ausgabe der Signale statt. Zudem wird die zeitliche Berechnung für die Aufnahme der Sensordaten pro Frequenz bestimmt und ggf. die Frequenz des Ausgabesignals erhöht.

Erklärung der Programmabschnitte in Abbildung 7.11:

1.: In diesem Block wird bei Auswahl „Periodisches Signal“ die Zeit bestimmt, wie lange die Sensordaten pro Frequenz aufgenommen werden sollen. Bei Auswahl „Rauschen“ wird hier die Rampenzeit abgewartet, bis die eigentliche Erfassungszeit der Sensordaten beginnt. Die Zeit kann nur in ganzen Sekunden verarbeitet werden, da der Programmdurchlauf eine feinere Bestimmung der Zeit nicht zulässt.

2.: Wenn die Sensordaten (bei Auswahl „Periodisches Signal“) zur aktuellen Frequenz für die berechnete Zeit erfasst wurden, wird hier die Frequenz um die Schrittweite und der Fortschrittsbalken um die in Abschnitt 7.1.6 Punkt 5 bestimmte Aufteilung erhöht. Bei „Rauschen“ wird hier die eingestellte Messzeit abgewartet und der Fortschrittsbalken fortlaufend erhöht.
3.: Hier werden die Sensordaten eingelesen.
4.: Die aufgenommenen Messdaten werden hier mit der Grenzfrequenz (vgl. 7.1.6 Punkt 8) tiefpassgefiltert und zusammengeführt.
5.: Dieser Block schreibt die zusammengeführten Messdaten für die spätere Auswertung in *DIAdem* in die *TDMS*-Datei (siehe Abschnitt 8.1).
6.: Einlesen des anregenden Signals, auf welches getriggert wird.
7.: Hier wird das Signal für die Ausgabe erzeugt (vgl. Abschnitt 7.2.14).
8.: Hier findet die Ausgabe des Anregungssignals statt. Zudem wird die Ausgabe hier einmalig mit der Case-Struktur gestartet (siehe auch Punkt 9).
9.: Dieser Teil dient der Bestimmung, ob die While-Shleife ihren ersten Durchlauf hat. Beim ersten Durchlauf der Schleife muss die Ausgangskarte einmalig gestartet werden.
10.: Hier wird beim automatischen Beenden der Messung die Endfrequenz der aktuellen Frequenz zugewiesen, da es sonst passieren kann, dass die Endfrequenz + Schrittweite auf der Bedienoberfläche angezeigt wird. Dies soll vermieden werden.
Das Problem tritt auf, da der Programmdurchlauf etwas länger braucht, um die While-Schleife zu beenden und die Endfrequenz + Schrittweite noch in das Anzeigeelement geschrieben wird. Das ausgegebene Signal wird davon jedoch nicht beeinflusst.
Beim manuellen Beenden der Messung durch Betätigung des Buttons „STOP“ wird nur die While-Schleife beendet.

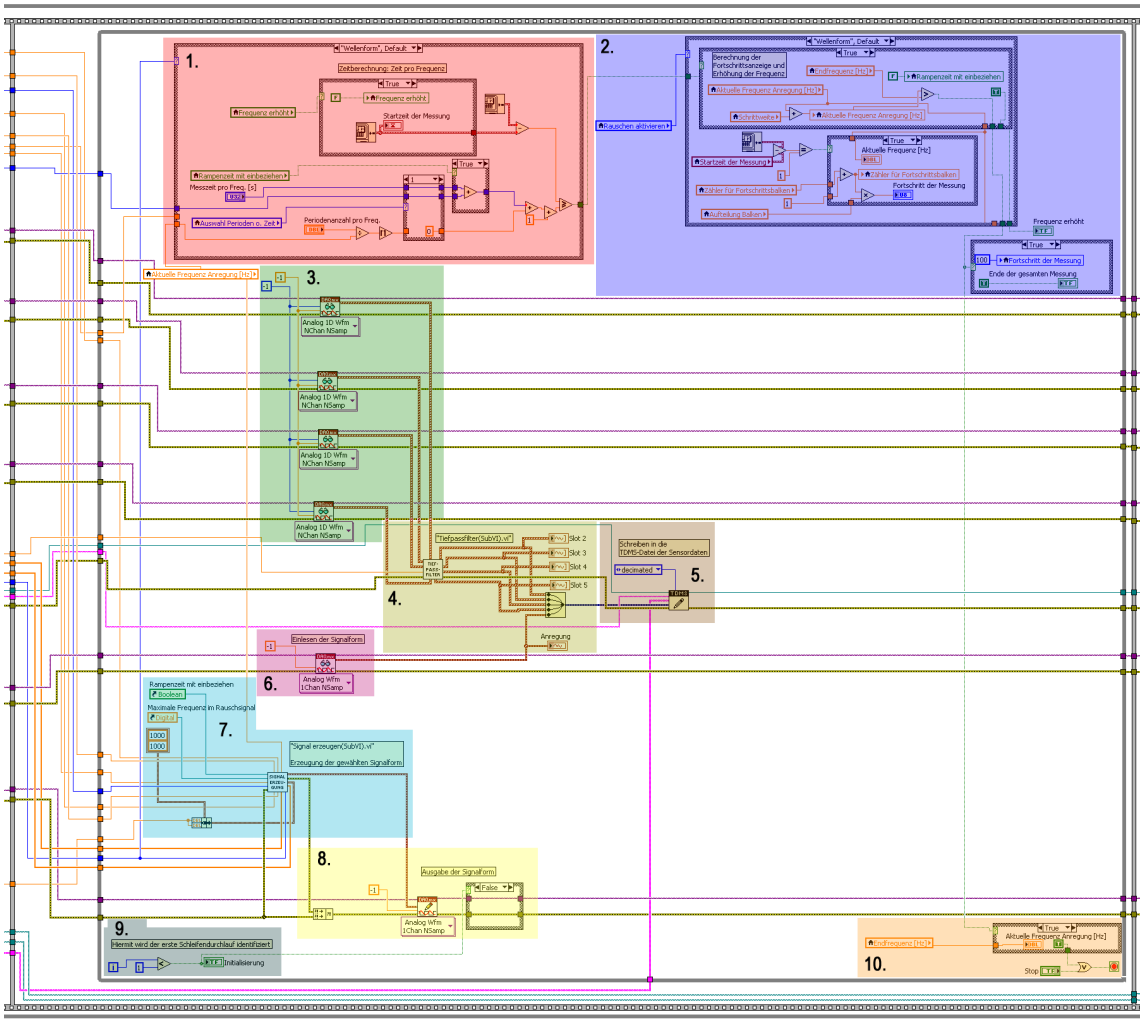


Abbildung 7.11.: Abschnitt 5 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.1.8. Programmabschnitt 6

Nachdem die Messung beendet ist, werden die *TDMS*-Datei „Name vom Bediener.tdms“ und der Ausgabetask sowie die Eingabetasks geschlossen und alle Ressourcen wieder frei gegeben (siehe Abbildung 7.12). Sollte während der Messung ein Fehler auftreten, wird dem Bediener durch die Bausteine „Simple Error Handler VI“ eine Fehlermeldung angezeigt. Diese enthält die Information, ob bei der *TDMS*-Datei oder bei einer Messkarte ein Fehler stattgefunden hat. Sollte ein Fehler bei einer Messkarte aufgetreten sein, so wird der Bediener weiterhin darüber informiert, um welche Messkarte es sich handelt.

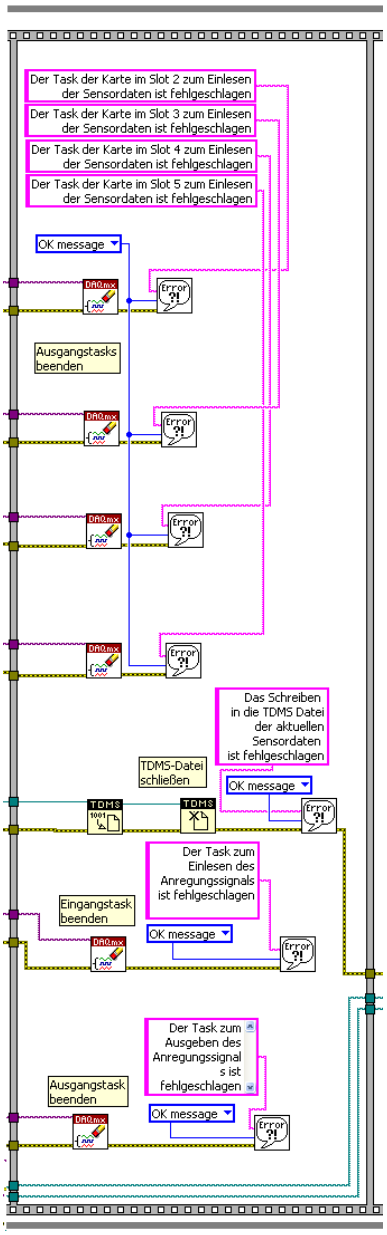


Abbildung 7.12.: Abschnitt 6 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.1.9. Programmabschnitt 7

Hier werden dem Sub-VI „*DIAdem* initialisieren und starten(SubVI).vi“ (vgl. 7.2.17) alle benötigten Informationen übergeben. Das sind der Name der Scriptdatei zur Auswertung der Daten sowie die Sensornamen und -orte (siehe Abbildung 7.13).

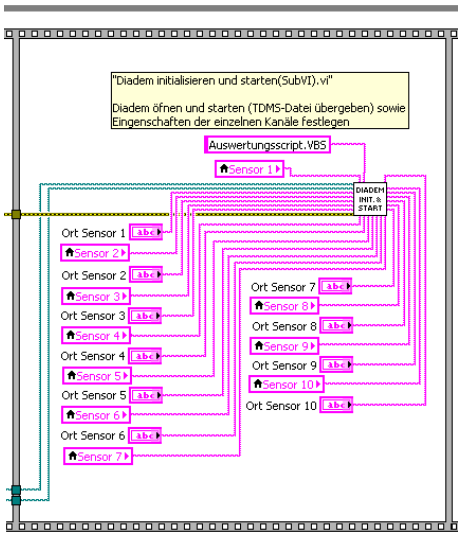


Abbildung 7.13.: Abschnitt 7 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.1.10. Programmabschnitt 8

Der letzte Teil der Sequenzstruktur erlaubt es dem Benutzer, zu wählen, ob er eine „Neue Messung starten“, „Neue Einstellungen vornehmen“ oder „*LabVIEW* beenden“ möchte (siehe Abbildung 7.14). Bei der Wahl „Neue Messung starten“ gelangt man direkt zur Menüseite 3 und muss so nicht erneut das gesamte Menü durchblättern. Jedoch ist es möglich, durch den „Zurück“-Button die Einstellungen auf den Menüseiten 1 oder 2 zu erreichen. Wird „Neue Einstellungen vornehmen“ gewählt, so fängt das Menü auf Menüseite 1 an. Beim Betätigen von „*LabVIEW* beenden“ wird das Programm beendet und geschlossen.

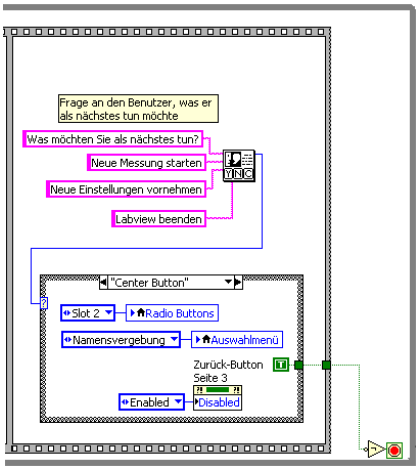


Abbildung 7.14.: Abschnitt 8 des gesamten Programms (Blockdiagramm), Autor: René Pfeifer

7.2. Unterprogramme (Sub-VIs)

Nachdem das gesamte *LabVIEW*-Programm erläutert wurde, soll nun auf die Unterprogramme (Sub-VIs) eingegangen werden.
Ein Sub-VI ist ein eigenes Programm (VI), das erst durch das Einbinden in einem anderen VI Sub-VI genannt wird. Sub-VIs werden im Blockdiagramm als kleine Blöcke dargestellt, an denen Anschlüsse für Ein- und Ausgangspunkte vorhanden sind (siehe Abbildung 7.15). Sie dienen der besseren Übersicht im Programm.

Abbildung 7.15.: Beispiel eines Sub-VIs, Autor: René Pfeifer

Das Aussehen der Sub-VIs kann auch selbst gestaltet werden. Somit wird noch einmal die Verständlichkeit und Übersichtlichkeit verbessert. Ein Beispiel dafür sieht man in Abbildung 7.16.

Abbildung 7.16.: Beispiel zum Aussehen eines Sub-VIs, Autor: René Pfeifer

Um ein Sub-VI zu erstellen, müssen die gewünschten Bausteine markiert werden. Anschließend wird durch den Menüpunkt „Edit“ ⇒ „Create a SubVI“ das gewünschte Sub-VI konfiguriert.
Die Anzahl von Sub-VIs in einem Programm ist nicht begrenzt und diese sollten großzügig benutzt werden. Manche Programmteile lassen sich aber nicht so einfach in ein Sub-VI umwandeln. Das tritt bei Programmabschnitten auf, die in sich verschachtelt sind und Anzeigeelemente enthalten. Dies ist z.B. in Abschnitt 7.1.7 unter Punkt 1 der Fall. In so einem Fall wird der Programmierer aber von *LabVIEW* darauf hingewiesen, dass es hier zu einer Änderung der Funktion kommen kann. Zudem ist es möglich, dass ein Sub-VI selbst ein Sub-VI beinhaltet.

Beispiel zum Erstellen eines Sub-VIs:

Zunächst liegt der normale Programmcode vor, in dem Ein- und Ausgabeelemente vorhanden sein müssen. Diese bilden später die Ein- und Ausgangspunkte des Sub-VIs (vgl. Abbildung 7.15).
Im Folgenden ist zunächst das Frontpanel (siehe Abbildung 7.17) und das Blockdiagramm (siehe Abbildung 7.18) des normalen Programmcodes zu sehen. Dieser wird nun in ein Sub-VI umgewandelt. Die benötigten Daten werden über die Ein- und Ausgangspunkte angeschlossen (siehe Abbildungen 7.19 und 7.20).

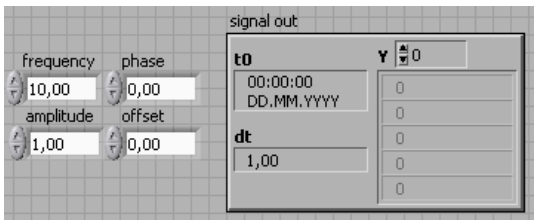


Abbildung 7.17.: Normaler Programmcode (Frontpanel), Autor: René Pfeifer

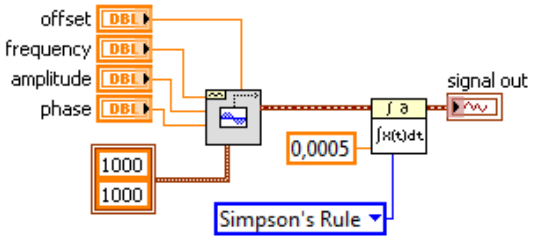


Abbildung 7.18.: Normaler Programmcode (Blockdiagramm), Autor: René Pfeifer

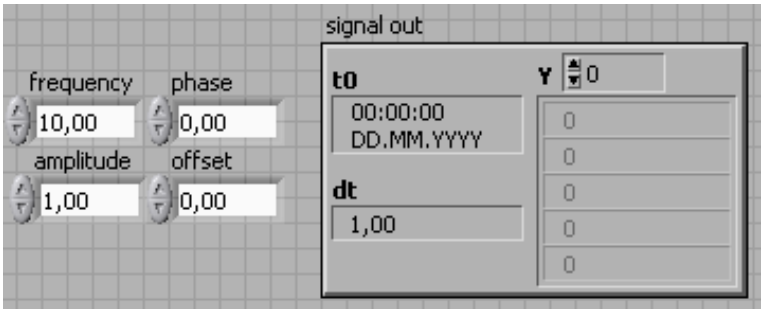


Abbildung 7.19.: Sub-VI Programmcode (Frontpanel), Autor: René Pfeifer

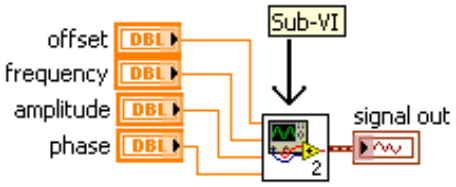


Abbildung 7.20.: Sub-VI Programmcode (Blockdiagramm), Autor: René Pfeifer

Dieses Sub-VI kann nun in beliebiger Anzahl in einem anderen VI eingesetzt werden. Dies ist vor allem bei Programmteilen, die gleich sind, von Vorteil. Damit kann der Programmieraufwand reduziert und zugleich die Übersichtlichkeit gesteigert werden.

7.2.1. Bild(SubVI).vi

Das erste Sub-VI dient lediglich dazu, anzuzeigen, wie die Anschlüsse der Messkarten zu nutzen sind. Zudem stehen unter den Messkarten die zugehörigen Nummern der Slots, in denen sich die Karten im Messsystem befinden. Obwohl die Blöcke des dafür benötigten Programmcodes nicht viel Platz einnehmen (siehe Abbildung 7.21 und 7.22), muss dieses in ein Sub-VI gespeichert werden, denn durch das Sub-VI hat man die Möglichkeit, diese Anschlussübersicht als Pop-Up-Fenster anzeigen zu lassen.

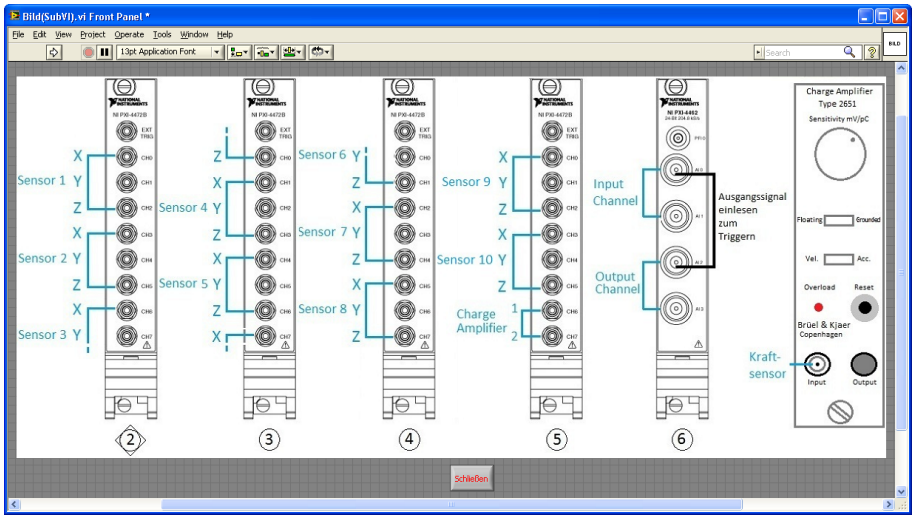


Abbildung 7.21.: Sub-VI zur Übersicht der Anschlüsse der Messkarten (Frontpanel), Autor: René Pfeifer

Abbildung 7.22.: Sub-VI zur Übersicht der Anschlüsse der Messkarten (Blockdiagramm), Autor: René Pfeifer

7.2.2. Informationen(SubVI).vi

In diesem Sub-VI (siehe Abbildung 7.23 und 7.24) stehen Informationen (siehe Anhang A), die vor Benutzung des Programms gelesen werden sollten. Darin befindet sich eine kurze Anleitung zur Bedienung des Programms sowie wichtige Hinweise, die bei der Bedienung beachtet werden sollen.

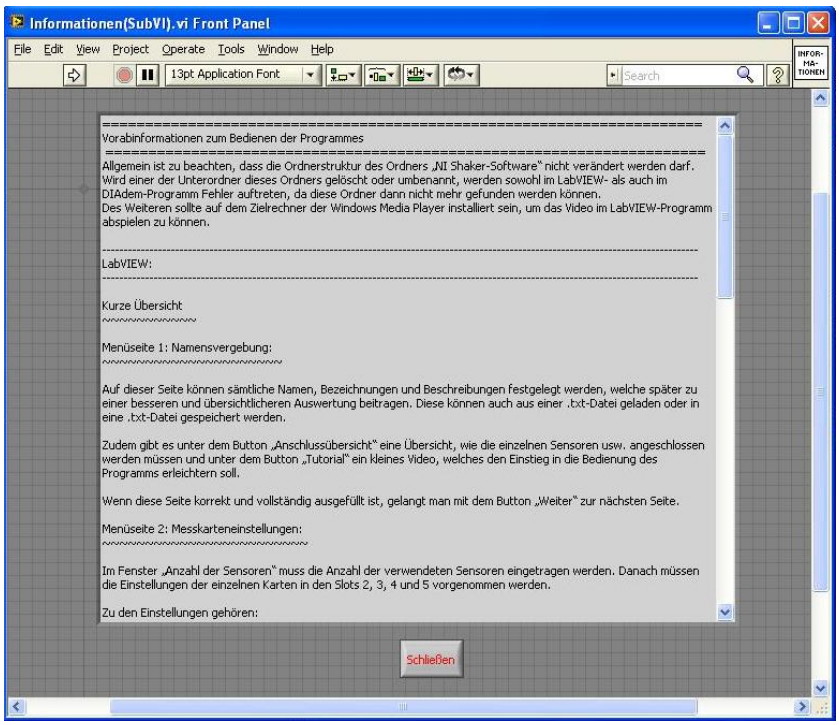


Abbildung 7.23.: Sub-VI für die Informationen (Frontpanel), Autor: René Pfeifer

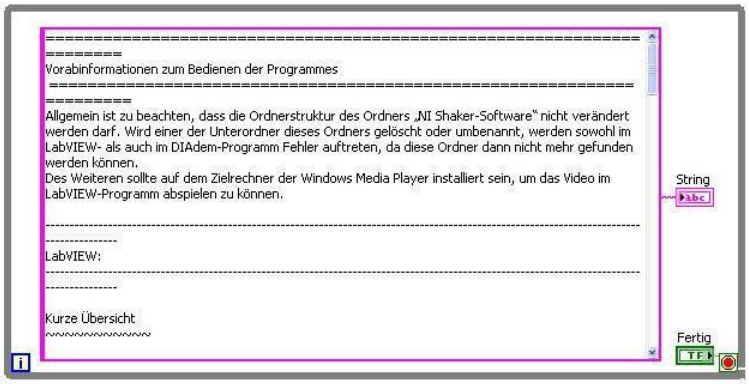


Abbildung 7.24.: Sub-VI für die Informationen (Blockdiagramm), Autor: René Pfeifer

7.2.3. Menüauswahl(SubVI).vi

In diesem Sub-VI ist die Menüsteuerung enthalten (siehe Abbildung 7.25 und 7.26). Es beinhaltet die Buttons „Weiter" und „Zurück" für das Menü sowie die beiden Sub-VIs „Bild(SubVI).vi" (vgl. Abschnitt 7.2.1) und „Informationen(SubVI).vi" (vgl. Abschnitt 7.2.2). Zudem wird damit auch die Anzeige der einzelnen Parameter der Karten, welche sich in den jeweiligen Slots befinden, gesteuert (siehe Abbildung 7.27).

Erklärung der Programmabschnitte in Abbildung 7.26:

1.: Hier werden die „Weiter“- und „Zurück“-Buttons mit den dazugehörigen Seiten verknüpft, sodass man z.B. bei Betätigung des Buttons „Weiter“ von Seite 1 zur Seite 2 des Menüs gelangt.
2.: In diesem Abschnitt werden die einzelnen Parameter der Karten ausgewählt und dementsprechend angezeigt. Alle Parameter einer Karten befinden sich auf einer extra Seite, um so die Übersicht zu verbessern.
3.: Dient dazu, bei Betätigung des Buttons „Anschlussübersicht“ die Anschlussübersicht der Messkarten als Pop-Up-Fenster aufzurufen (vgl. Abschnitt 7.2.1).
4.: Ruft bei Betätigung des Buttons „Information“ das Pop-Up-Fenster mit den Vorabinformationen auf (vgl. Abschnitt 7.2.2).

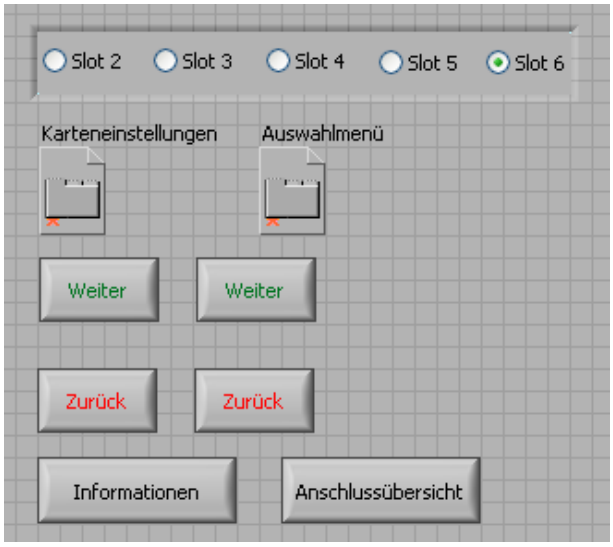


Abbildung 7.25.: Sub-VI für die Steuerung des Menüs (Frontpanel), Autor: René Pfeifer

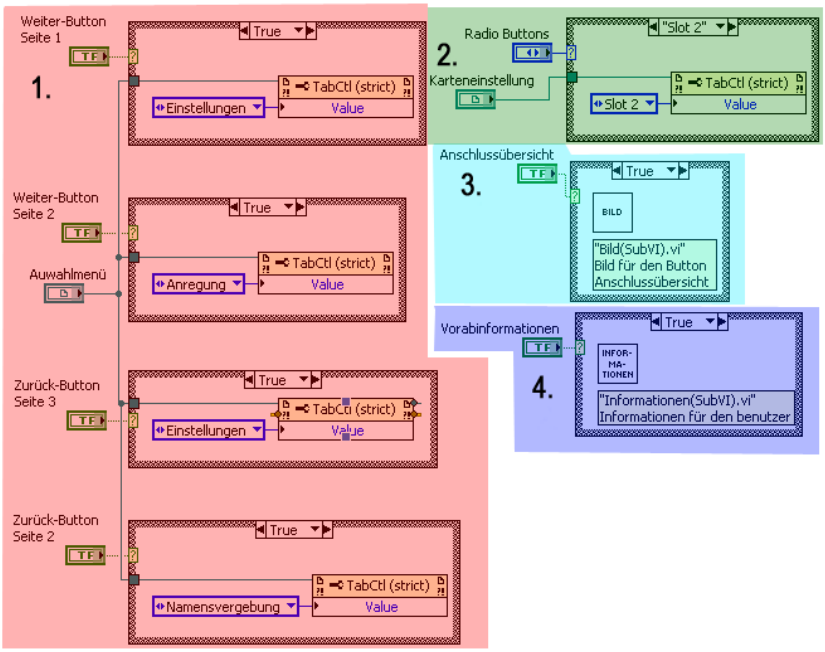


Abbildung 7.26.: Sub-VI für die Steuerung des Menüs (Blockdiagramm), Autor: René Pfeifer

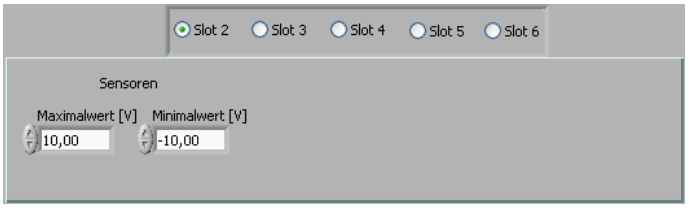


Abbildung 7.27.: Sloteinstellungen auf der Menüseite 2 (Frontpanel), Autor: René Pfeifer

7.2.4. TXT laden(SubVI).vi

Mit diesem Sub-VI (siehe Abbildungen 7.28 und 7.29) kann eine .txt-Datei durch Betätigung des Buttons „Laden“ aus einem beliebigen Pfad geladen werden. Das Format, in dem die Sensornamen und -orte in dieser Datei vorhanden sein müssen, ist fest vorgegeben. Zuerst müssen alle zehn Sensornamen hintereinander mit dem Trennzeichen „|“, danach die zehn Sensororte, auch jeweils mit dem Trennzeichen „|“, in der .txt-Datei enthalten sein. Diese dürfen nicht durch einen Zeilenumbruch getrennt sein.

Beispiel zur .txt-Datei:

Sensor 1|Sensor 2|Sensor 3|Sensor 4|Sensor 5|Sensor 6|Sensor 7|Sensor 8|Sensor 9|Sensor 10|Ort 1|Ort 2|Ort 3|Ort 4|Ort 5|Ort 6|Ort 7|Ort 8|Ort 9|Ort 10

Erklärung des Programmabschnitts in Abbildung 7.29:

Durch die Referenzen „Sensor 1“ bis „Sensor 10“ und „Ort 1“ bis „Ort 10“ werden die Stringinhalte direkt in die Bedienelemente „Sensor 1“ bis „Sensor 10“ und „Ort des Sensors 1“ bis „Ort des Sensors 10“ geschrieben. Dazu wird in diesem Sub-VI der eingelesene String aus der .txt-Datei aufgeteilt, indem im String nach dem Trennzeichen „|“ gesucht wird. Alles was vor dem ersten Trennzeichen steht, bildet den Sensornamen für den ersten Sensor. Das Trennzeichen „|“ wird abgeschnitten und der restliche String wird nach dem selben Verfahren weiter aufgeteilt.

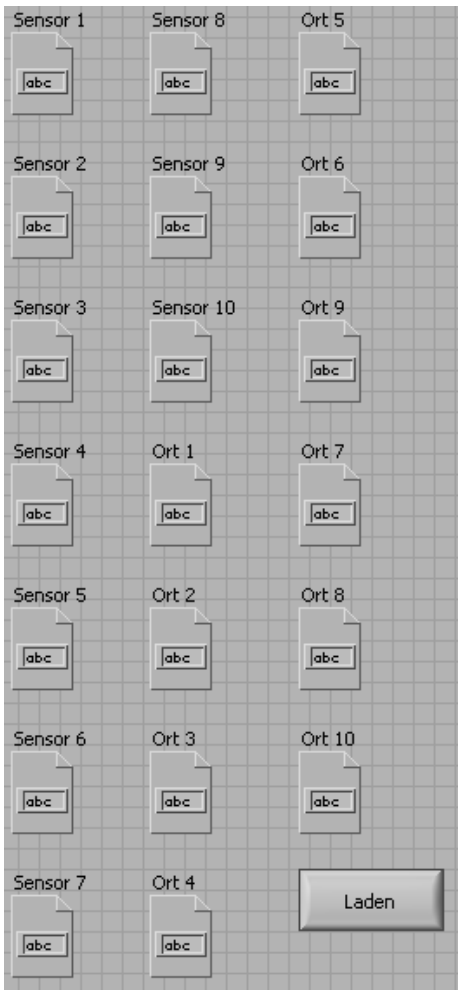


Abbildung 7.28.: Sub-VI zum Laden einer .txt-Datei (Frontpanel), Autor: René Pfeifer

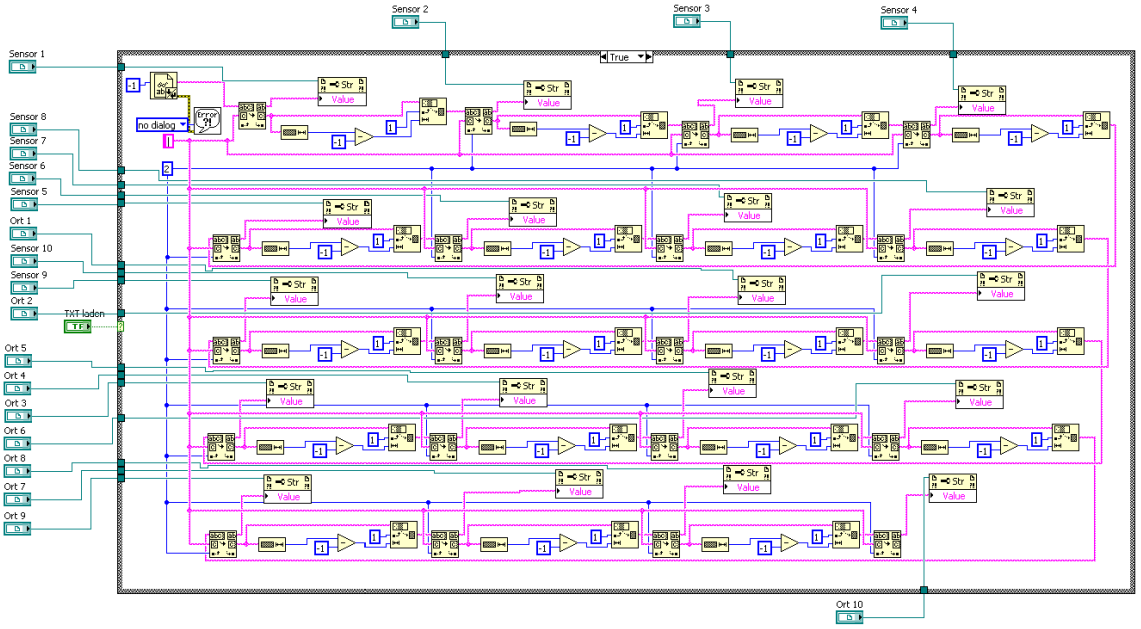


Abbildung 7.29.: Sub-VI zum Laden einer .txt-Datei (Blockdiagramm), Autor: René Pfeifer

7.2.5. TXT speichern(SubVI).vi

In diesem Sub-VI werden bei Betätigung des Buttons „Speichern“ die vom Bediener auf der Bedienoberfläche eingegebenen Sensornamen und -orte in einer .txt-Datei gespeichert. Der Pfad und Dateiname ist vom Bediener frei wählbar. Hier wird dieselbe feste Vorgabe wie in Abschnitt 7.2.4 verwendet. In diesem Fall wird aus den einzelnen Strings eine Stringkette mit dem Trennzeichen „|“ erstellt (siehe Abbildungen 7.30 und 7.31).

Erklärung des Programmabschnitts in Abbildung 7.31:

Mit Hilfe der Referenz werden die Stringinhalte der Stringelemente „Sensor 1“ bis „Sensor 10“ und „Ort des Sensors 1“ bis „Ort des Sensors 10“ ausgelesen und mit dem Trennzeichen „|“ zu einem String zusammengefügt und anschließend in einer .txt-Datei gespeichert.

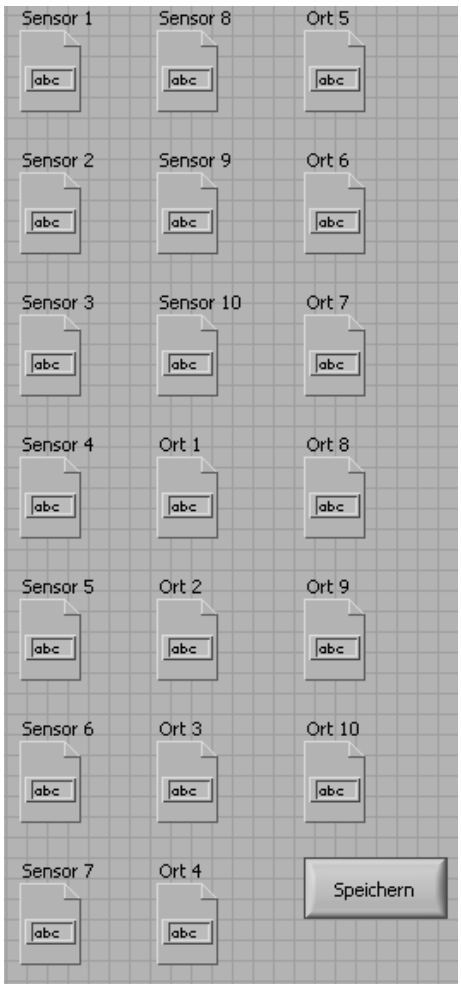


Abbildung 7.30.: Sub-VI zum Speichern einer.txt-Datei (Frontpanel), Autor: René Pfeifer

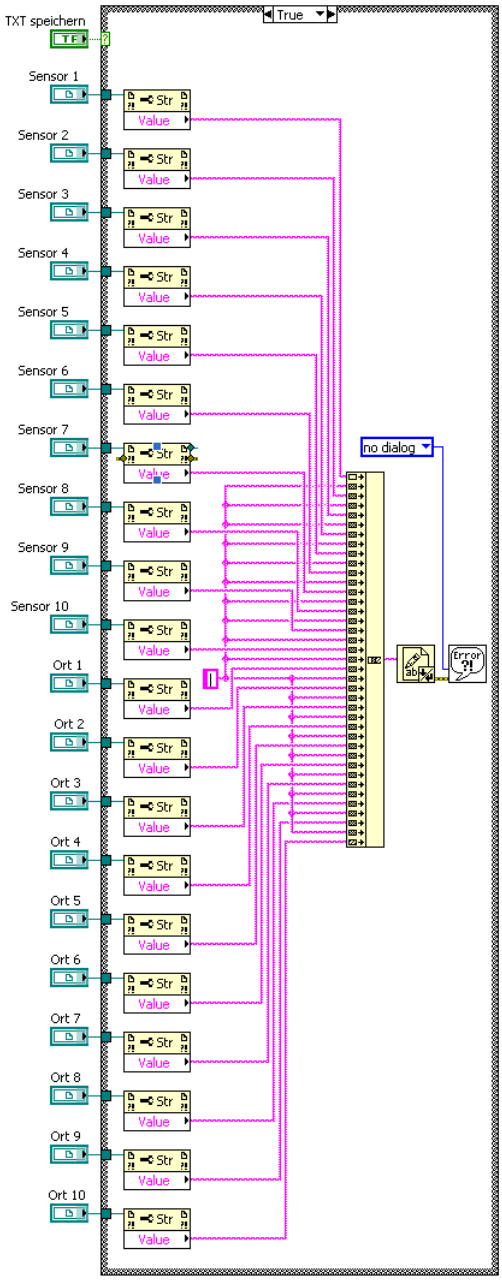


Abbildung 7.31.: Sub-VI zum Speichern einer .txt-Datei (Blockdiagramm), Autor: René Pfeifer

7.2.6. Bedienelemente deaktivieren(SubVI).vi

Dieses Sub-VI deaktiviert während der Aufnahme der Messdaten Bedienelemente auf der Bedienoberfläche, um die Eingabe vom Bediener zu verhindern (siehe Abbildungen 7.32 und 7.33).

Erklärung des Programmabschnitts in Abbildung 7.33:

Mit Hilfe der Referenzen auf die Bedienelemente werden diese durch die Knotenpunkteigenschaft „Disabled“ mit dem Eingangssignal „Disabled and Grayed Out“ deakiviert.

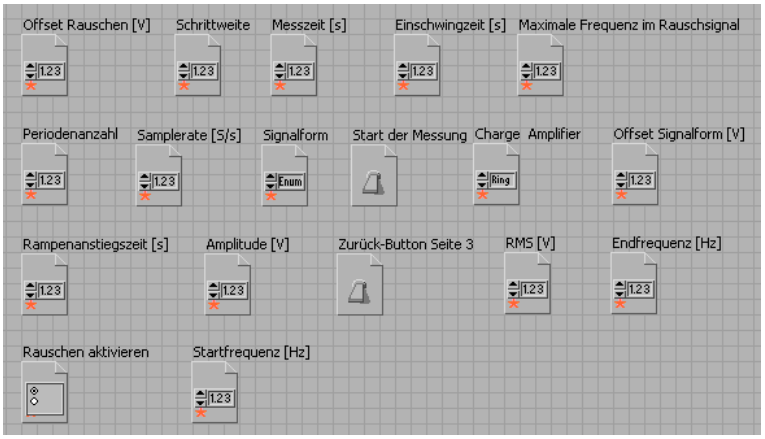


Abbildung 7.32.: Sub-VI zum Deaktivieren von Bedienelementen (Frontpanel), Autor: René Pfeifer

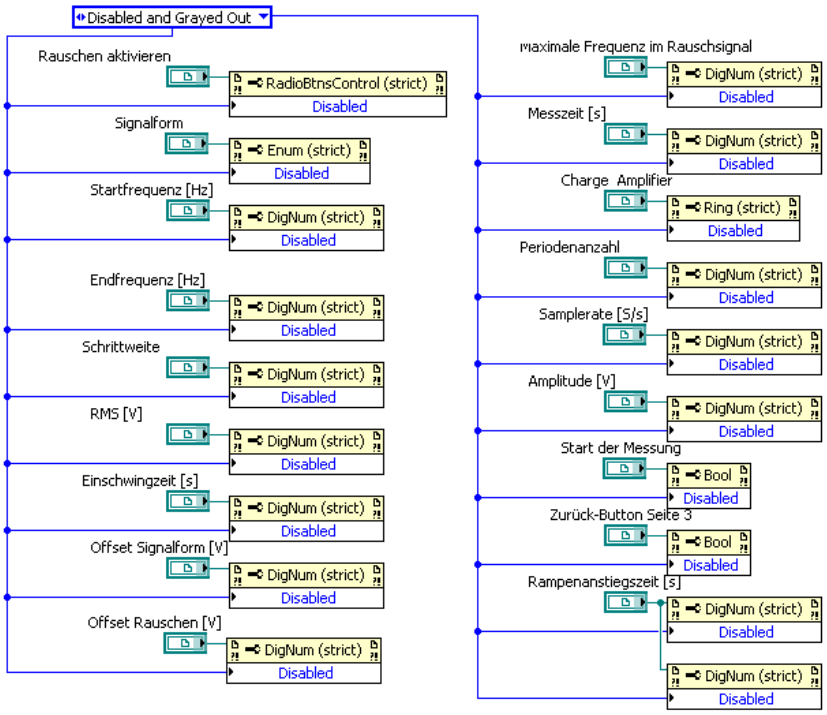


Abbildung 7.33.: Sub-VI zum Deaktivieren von Bedienelementen (Blockdiagramm), Autor: René Pfeifer

7.2.7. Bedienelemente aktivieren(SubVI).vi

Hier werden die Bedienelemente, die vom „Bedienelemente deaktivieren(SubVI).vi“ (vgl. Abschnitt 7.2.6) deaktiviert werden, wieder aktiviert, um eine Eingabe durch den Bediener wieder zuzulassen (siehe Abbildungen 7.34 und 7.35).

Erklärung des Programmabschnitts in Abbildung 7.35:

Mit Hilfe der Referenzen auf die Bedienelemente werden diese durch die Knotenpunkteigenschaft „Disabled“ mit dem Eingangssignal „Enabled“ akiviert.

Abbildung 7.34.: Sub-VI zum Aktivieren von Bedienelementen (Frontpanel), Autor: René Pfeifer

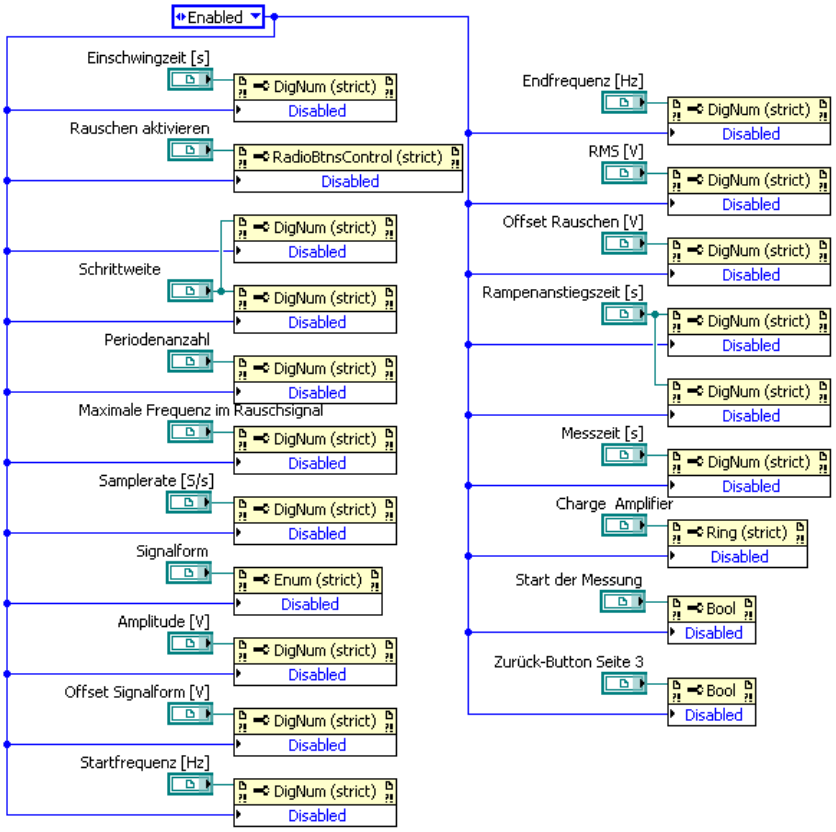


Abbildung 7.35.: Sub-VI zum Aktivieren von Bedienelementen (Blockdiagramm), Autor: René Pfeifer

7.2.8. Tutorial Video(SubVI).vi

Dieses Sub-VI dient dem Öffnen und Abspielen des Tutorial Videos. Dies soll die Einführung in das Programm erleichtern (siehe Abbildungen 7.36 und 7.37).

Erklärung des Programmabschnitts in Abbildung 7.37:

Durch Betätigung des Buttons „Tutorial Video“ wird der Pfad des Videos an den Knotenpunkt, der das Video aus dem Pfad öffnet, weitergeleitet.

Abbildung 7.36.: Sub-VI zum Öffnen des Tutorial Videos (Frontpanel), Autor: René Pfeifer

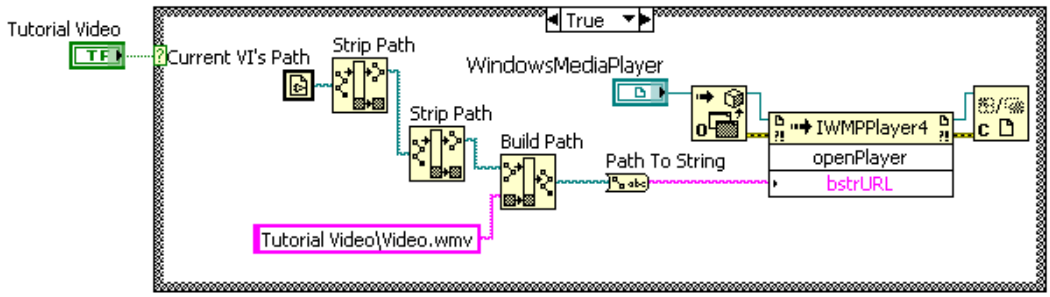


Abbildung 7.37.: Sub-VI zum Öffnen des Tutorial Videos (Blockdiagramm), Autor: René Pfeifer

7.2.9. Periodisches Signal oder Rauschen(SubVI).vi

In diesem Sub-VI findet die Auswahl zwischen „Periodisches Signal“ und „Rauschen“ statt (siehe Abbildungen 7.38 und 7.39).

Erklärung des Programmabschnitts in Abbildung 7.39:

Hier wird das Bedienelement „Rampenanstiegszeit“ deaktiviert, wenn der Signaltyp Rauschen gewählt ist und umgekehrt. Die Rampenanstiegszeit bestimmt die Steigung der Rampe (da der Endpunkt auf der Ordinate durch den eingegebenen Offset feststeht). Weiterhin befindet sich die Reitersteuerung, welche die Parametereinstellungen nur für den gewählten Signaltyp anzeigt, in diesem Sub-VI.

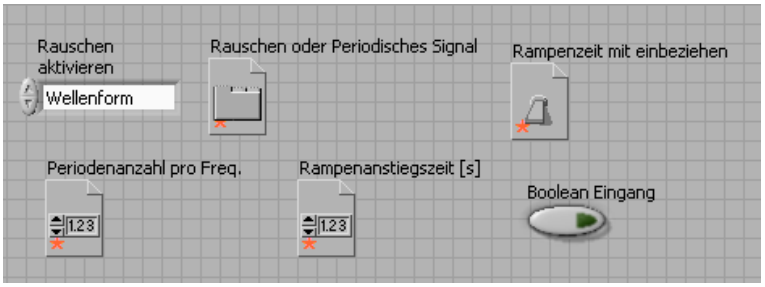


Abbildung 7.38.: Sub-VI für Einstellungen für das periodische Signal oder Rauschen (Frontpanel), Autor: René Pfeifer

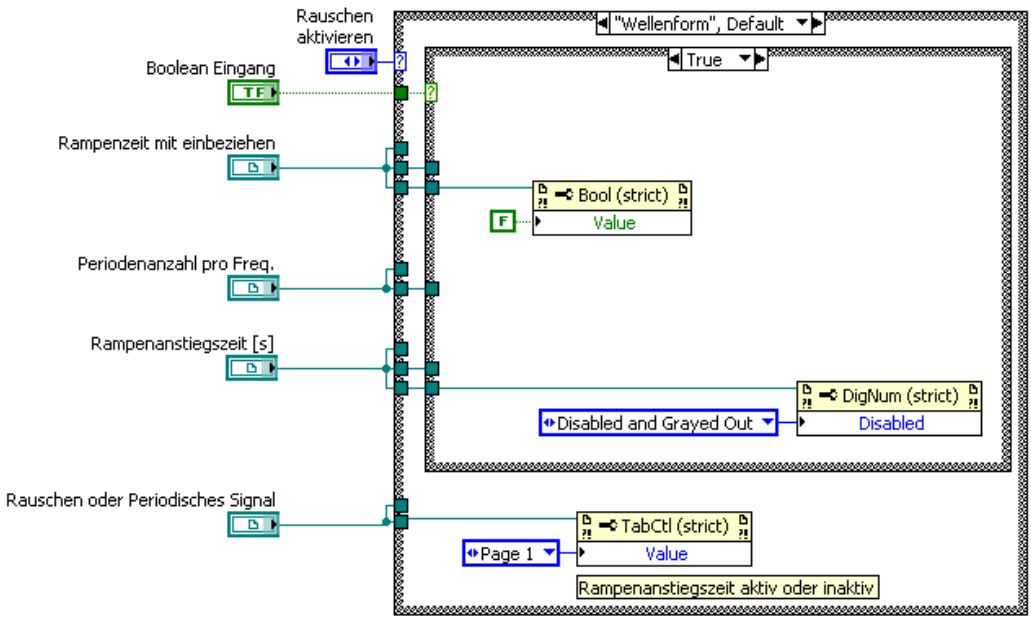


Abbildung 7.39.: Sub-VI für Einstellungen für das periodische Signal oder Rauschen (Blockdiagramm), Autor: René Pfeifer

7.2.10. Kontrolle der eingebenen Werte(SubVI).vi

In diesem Sub-VI wird überprüft, ob die Amplitude des Anregungssignals die maximale Eingangsspannung des Leistungsverstärkers (siehe F) einhält. Zudem wird kontrolliert, ob die Eingabe der Schrittweite, Start- und Endfrequenz zueinander passen (siehe Abbildungen 7.40 und 7.41).

Erklärung des Programmabschnitts in Abbildung 7.41:

Hier wird bei Betätigung des „Start“-Buttons geprüft, ob die gewählten Einstellungen zueinander passen. Das bedeutet, es wird kontrolliert, ob die Amplitude $\pm$ Offset die maximale oder minimale Ausgabespannung einhält. Des Weiteren wird kontrolliert, ob die Schrittweite ganzzahlig in Startfrequenz minus Endfrequenz passt. Zum Schluss wird kontrolliert, ob die Endfrequenz größer als die Startfrequenz ist. Sollte eine der Kontrollen nicht richtig sein, wird der Bediener mit einer entsprechenden Fehlermeldung darauf hingewiesen und die Messung erst gestartet, wenn alle Eingaben richtig sind.

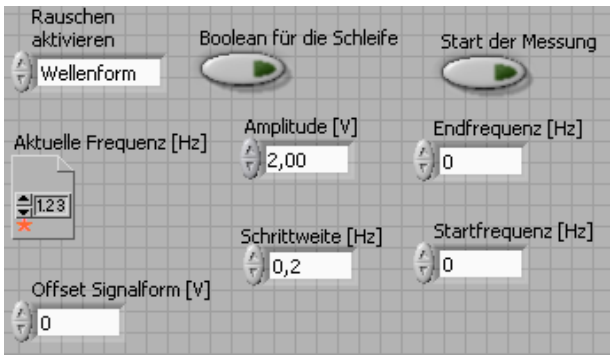


Abbildung 7.40.: Sub-VI zur Kontrolle der gewählten Einstellungen (Frontpanel), Autor: René Pfeifer

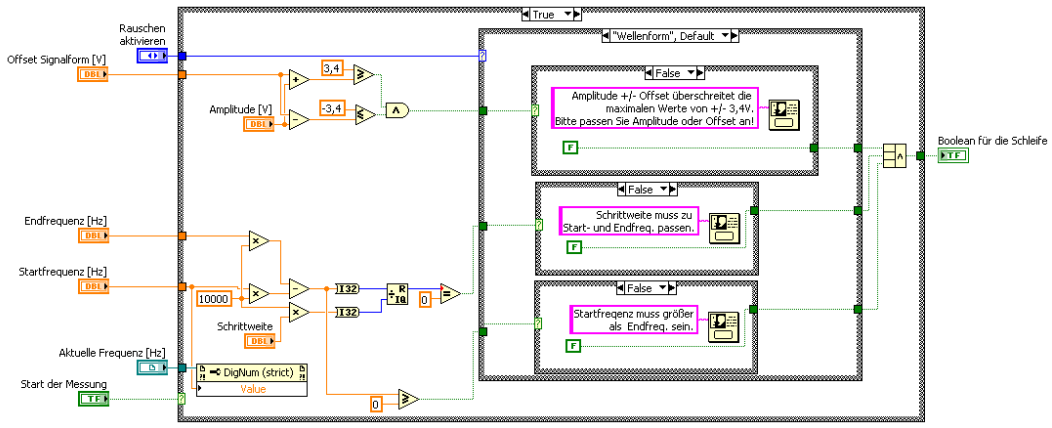


Abbildung 7.41.: Sub-VI zur Kontrolle der gewählten Einstellungen (Blockdiagramm), Autor: René Pfeifer

7.2.11. Rampenzeit bestimmen(SubVI).vi

Hier wird die Rampenzeit für das gewählte Signal bestimmt und das Triggerlevel für die Triggerung ausgegeben (siehe Abbildungen 7.42 und 7.43).

Erklärung des Programmabschnitts in Abbildung 7.43:

Hier wird zum einen die zugehörige Rampenanstiegszeit sowie die passenden Werte für das Triggerlevel und für den Endpunkt der Rampenfunktion ausgegeben.
Beim Rauschen ist eine konstante Anstiegszeit von zwei Sekunden festgelegt sowie ein fester Wert von 1 V für den Anstieg der Rampe. Die Anstiegszeit beim periodischen Signal kann vom Bediener gewählt werden. Der Endpunkt wird hier durch den Offset des Signals festgelegt. Falls kein Offset vorhanden ist, ist auch keine Rampe nötig, da das Signal im Nullpunkt beginnt.
Der Ausgang des Triggerlevels wird in beiden Fällen durch den Offset des jeweiligen Signals in Abschnitt 7.2.22 bestimmt.

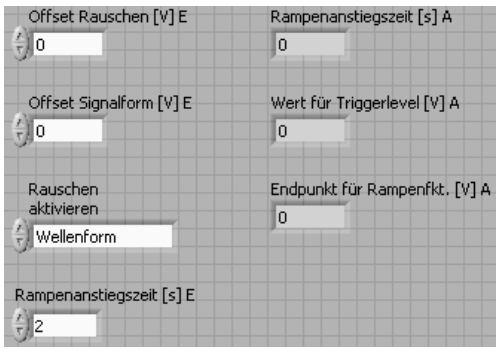


Abbildung 7.42.: Sub-VI zur Bestimmung der Rampenzeit sowie den Werten für das Triggerlevel und Endpunkt der Rampe (Frontpanel), Autor: René Pfeifer

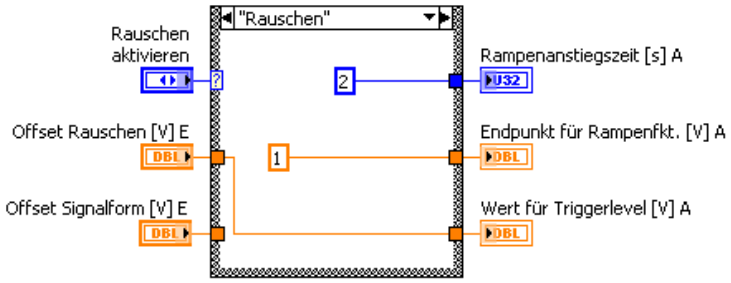


Abbildung 7.43.: Sub-VI zur Bestimmung der Rampenzeit sowie den Werten für das Triggerlevel und Endpunkt der Rampe (Blockdiagramm), Autor: René Pfeifer

7.2.12. Rechteckanteil Bedienelement deaktivieren(SubVI).vi

Dieses Sub-VI macht die Eingabe „positiver Anteil in % (Rechtecksignal)“ bei Auswahl der Signalform Rechteck sichtbar oder unsichtbar bei Auswahl einer anderen Signalform (siehe Abbildungen 7.44 und 7.45).

Erklärung des Programmabschnitts in Abbildung 7.45:

Mit Hilfe der Referenzen auf das Bedienelement wird dieses durch die Knotenpunkteigenschaft „Visible“ mit dem Eingangssignal „True“ sichtbar oder „False“ unsichtbar gesetzt.

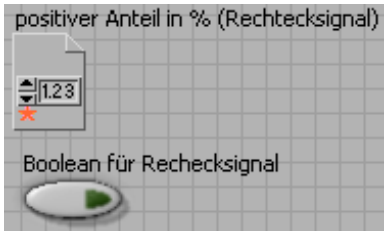


Abbildung 7.44.: Sub-VI zum Ein- oder Ausblenden des Bedienelements „Rechtecksignal“ (Frontpanel), Autor: René Pfeifer

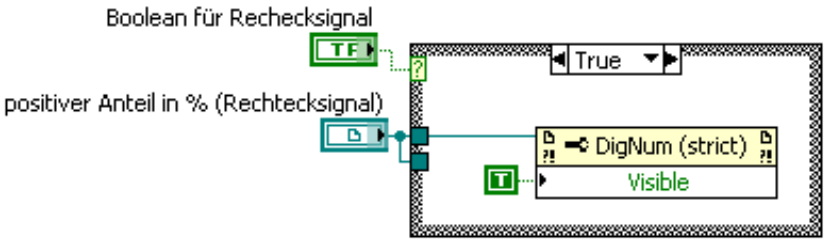


Abbildung 7.45.: Sub-VI zum Ein- oder Ausblenden des Bedienelements „Rechtecksignal“ (Blockdiagramm), Autor: René Pfeifer

7.2.13. Schrittweite Bedienelement deaktivieren(SubVI).vi

Wenn die Start- und Endfrequenz den gleichen Wert besitzen, wird das Bedienelement „Schrittweite“ deaktiviert und bei ungleichen Werten wieder aktiviert (siehe Abbildungen 7.46 und 7.47).

Erklärung des Programmabschnitts in Abbildung 7.47:

Mit Hilfe der Referenzen auf das Bedienelement „Schrittweite“ wird dieses durch die Knotenpunkteigenschaft „Disabled“ mit dem Eingangssignal „Enabled“ aktiviert oder „Disabled and Grayed Out“ deaktiviert.

Abbildung 7.46.: Sub-VI zum Ein- oder Ausblenden des Bedienelements „Schrittweite“ (Frontpanel), Autor: René Pfeifer

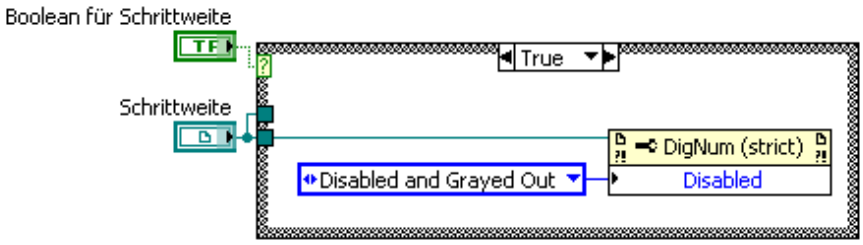


Abbildung 7.47.: Sub-VI zum Ein- oder Ausblenden des Bedienelements „Schrittweite“ (Blockdiagramm), Autor: René Pfeifer

7.2.14. Signal erzeugen(SubVI).vi

Dieses Sub-VI dient der Signalerzeugung. Hier wird das vom Bediener ausgewählte Signal mit dessen Parametern erzeugt (siehe Abbildungen 7.48 bis 7.50).

Erklärung des Programmabschnitts in den Abbildungen 7.49 und 7.50:

Zuerst werden alle benötigten Referenzen und Größen für die Signalerzeugung eines periodischen oder Rauschsignals eingelesen.
Bei der Auswahl einer Wellenform (periodisches Signal, vgl. Abbildung 7.49) wird ggf. zuerst eine Rampenfunktion ausgegeben und danach die gewählte Signalform mit den gewünschten Frequenzen.
Beim Rauschen wird die Rampenfunktion immer mit einbezogen. Es wird eine Rampe von 0 V bis 1 V erzeugt und mit dem aktuellen Rauschen multipliziert. Das dient dazu, dass die eventuell große Amplitude nicht impulsartig auf den Verstärker und Shaker gegeben wird. Nachdem die Rampenzeit von zwei Sekunden abgelaufen ist, wird das Rauschsignal ohne Veränderung ausgegeben. Zudem wird bei der Signalart „Rauschen“ das erzeugte Signal tiefpassgefiltert, um das Frequenzband etwas zu verringern und so mehr Energie in das restliche Frequenzband einbringen zu können. Die maximale Frequenz im Rauschsignal wird vom Bediener festgelegt.

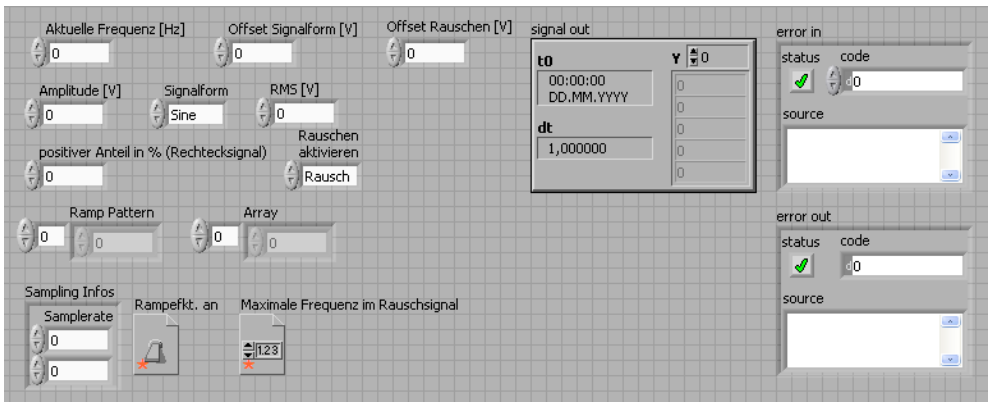


Abbildung 7.48.: Sub-VI zur Signalerzeugung (Frontpanel), Autor: René Pfeifer

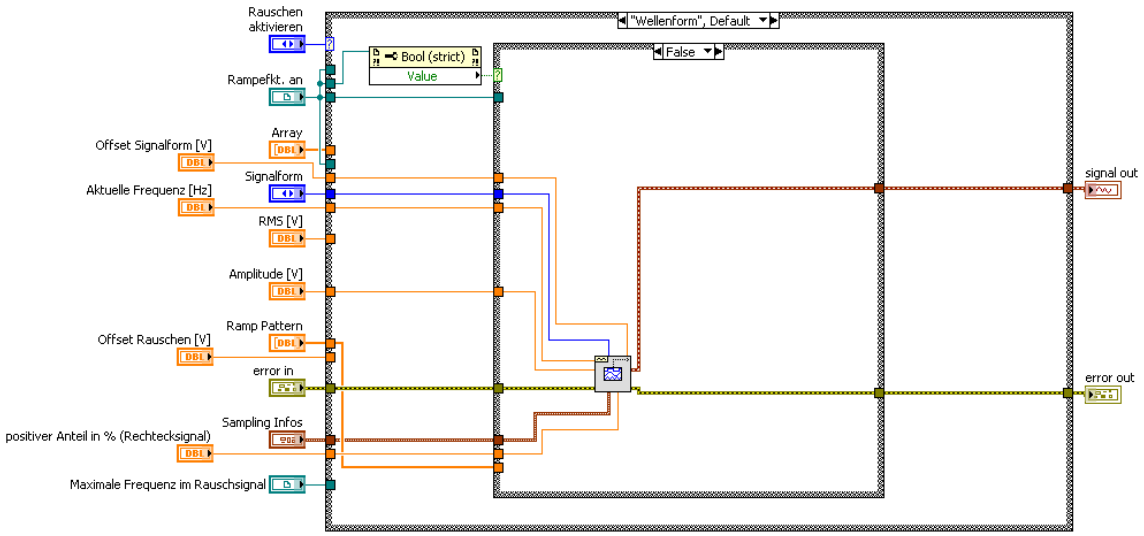


Abbildung 7.49.: Sub-VI zur Signalerzeugung (Periodisches Signal) (Blockdiagramm), Autor: René Pfeifer

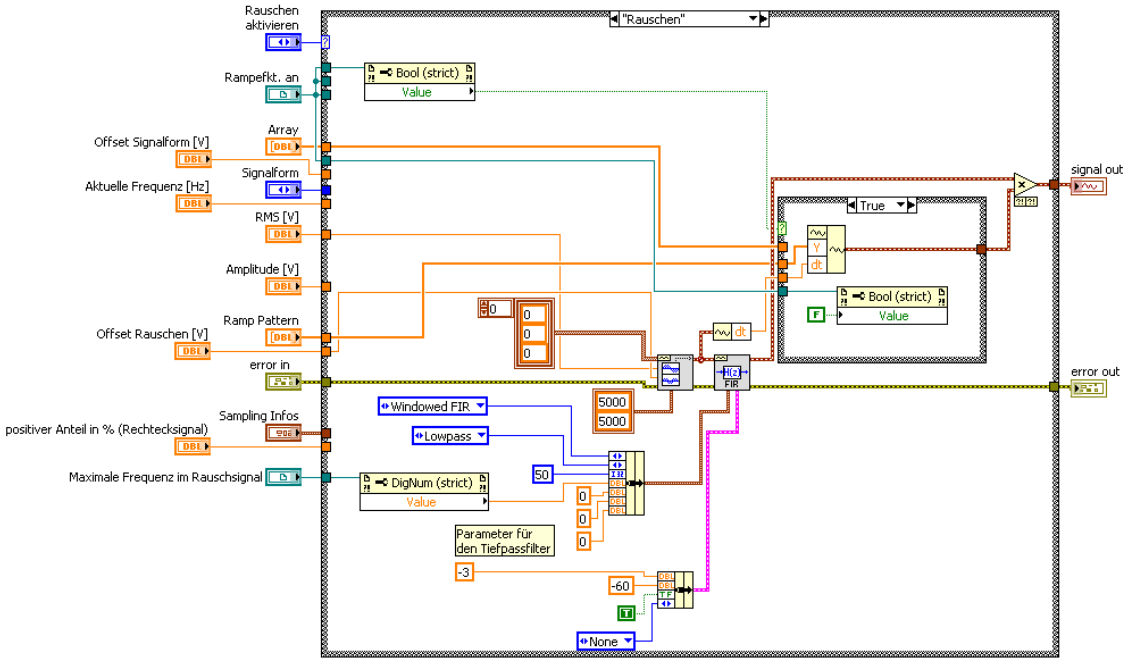


Abbildung 7.50.: Sub-VI zur Signalerzeugung (Rauschen) (Blockdiagramm), Autor: René Pfeifer

7.2.15. Name für TDMS Datei(SubVI).vi

Das Sub-VI (siehe Abbildungen 7.51 und 7.52) dient der Namensbildung für die *TDMS*-Datei, in welche die gesamten Messwerte geschrieben werden.

Erklärung der Programmabschnitte in Abbildung 7.52:

1.: Die aktuelle Zeit und das Datum wird zum einen zum Erstellen des Dateinamens für die *TDMS*-Datei in einen String umgewandelt und zum anderen als Vergleichszeit für das Rauschen verwendet. Diese dient als Bezugs-/Startpunkt der Messdauer.
2.: Hier wird die Zeit und das Datum aufgesplittet, sodass Jahr, Monat, Tag und die Uhrzeit als einzelne Strings vorliegen.
3.: Der Block ersetzt in der Uhrzeit das Zeichen „:“ durch „;“, weil Dateinamen in Windows keinen Doppelpunkt enthalten dürfen.
4.: Die einzelnen Stringelemente werden hier zu einem Dateinamen zusammengesetzt. Das sieht zum Beispiel so aus: Messdaten 2011-11-07 10;31;45.tdms.

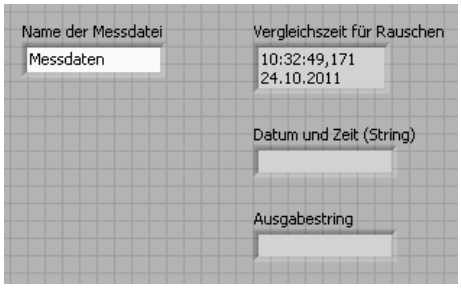


Abbildung 7.51.: Sub-VI zur Erstellung des Namens für die *TDMS*-Datei (Frontpanel), Autor: René Pfeifer

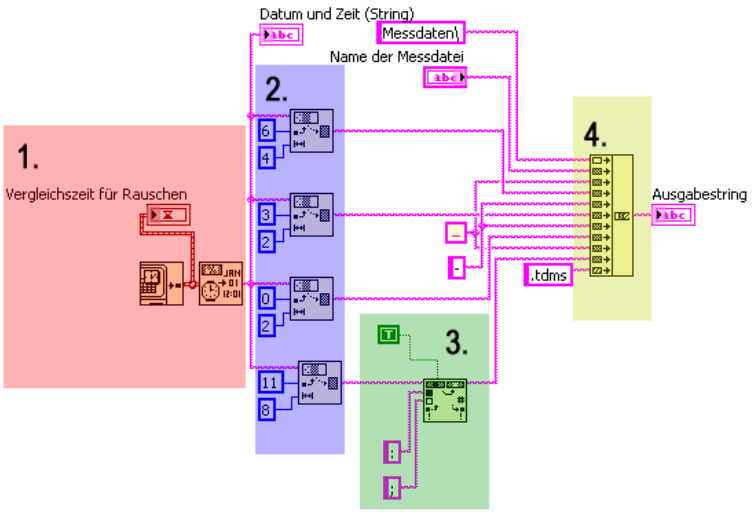


Abbildung 7.52.: Sub-VI zur Erstellung des Namens für die *TDMS*-Datei (Blockdiagramm), Autor: René Pfeifer

7.2.16. Namen der Sensoren für *DIAdem* erstellen(SubVI).vi

In diesem Sub-VI (siehe Abbildungen 7.53 und 7.54) werden dem Namen des Sensors, welchen der Benutzer gewählt hat, automatisch die drei Richtungen X, Y und Z hinzugefügt. Das ist notwendig, um die Messwerte später den Richtungen zuordnen zu können und dem Benutzer die Eingabe zu erleichtern, denn es muss nur einmal der Name des Sensors festgelegt werden, anstatt den Sensornamen drei Mal mit den unterschiedlichen Richtungen einzugeben.

Beispiel:

Der Name „Sensor 1“ wird für die drei Richtungen in die Namen „Sensor 1 X“, „Sensor 1 Y“ und „Sensor 1 Z“ geändert.

Zudem werden hier die Informationen über den Ort des Sensors und die Einheit an *DIAdem* übermittelt. Dieses geschieht, damit dem Bediener diese Informationen zu den einzelnen Kanälen im aktuell geöffneten *DIAdem* zur Verfügung stehen. Diese Informationen sind jedoch nicht mehr vorhanden, wenn die *TDMS*-Datei erneut geöffnet wird. Damit die Sensornamen beim erneuten Öffnen weiterhin vorhanden sind, ist ein weiterer Programmteil nötig, den das Sub-VI aus Abschnitt 7.2.18 enthält.

Erklärung der Programmabschnitte in Abbildung 7.54:

1.: Hier werden aus dem einen Sensornamen drei Sensornamen mit der Richtungskennzeichnung generiert.
2.: Die drei Namen werden hier mit den Informationen über den Ort des Sensors und der Messeinheit zu einem Cluster[4] verbunden.

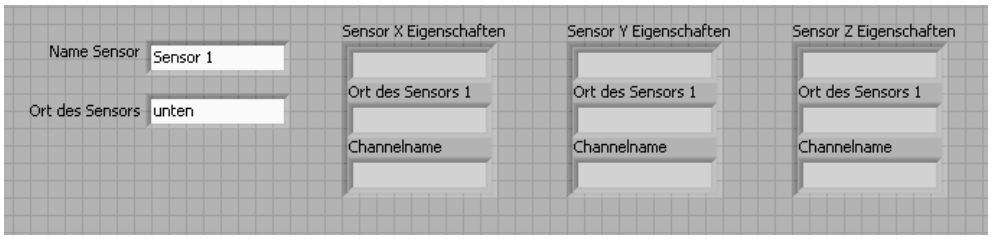


Abbildung 7.53.: Sub-VI für die Erstellung der Sensornamen für *DIAdem* (Frontpanel), Autor: René Pfeifer

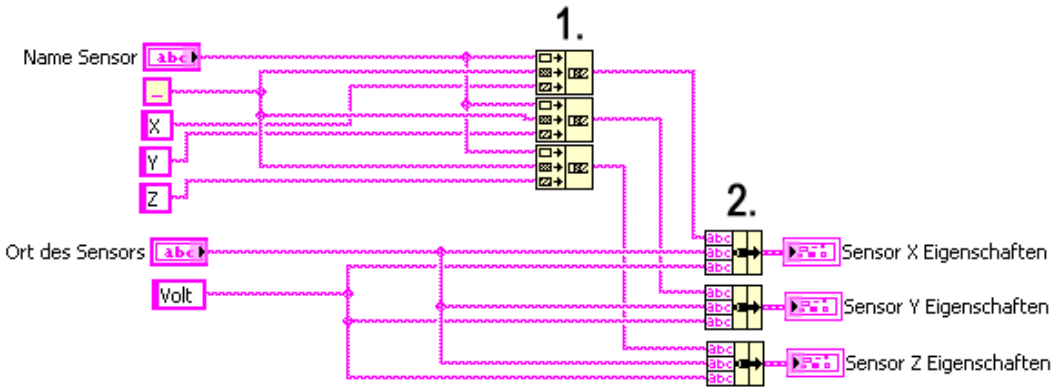


Abbildung 7.54.: Sub-VI für die Erstellung der Sensornamen für *DIAdem* (Blockdiagramm), Autor: René Pfeifer

7.2.17. *DIAdem* initialisieren und starten(SubVI).vi

Dieses Sub-VI (siehe Abbildung 7.55 und 7.56) dient dazu, *DIAdem* zu starten (wenn es noch nicht geöffnet ist), *DIAdem* den Pfad der Scriptdatei und der *TDMS*-Datei zu übergeben und den einzelnen Kanälen („Channels“) zusätzliche Informationen zuzuweisen. Bei diesen Informationen handelt es sich um den Namen des Sensors inklusive der Richtung (Kanalname), den Ort, an dem sich der Sensor befindet (Beschreibung („Description“)) und die Einheit der Messgröße („Unit“). Die Namen der Sensoren, der Ort und die Einheit werden in einem Sub-VI zusammengesetzt (vgl. Abschnitt 7.2.16).

Erklärung der Programmabschnitte in Abbildung 7.56:

1.: Hier wird *DIAdem* initialisiert und, wenn es noch nicht geöffnet ist, gestartet. Es wird *DIAdem* der Pfad mit dem Namen der *TDMS*-Datei der Messung sowie der Pfad der Script-Datei mit dem Namen der Script-Datei übergeben.
2.: In diesem Teil werden aus den eingegebenen Sensornamen die vollständigen Bezeichnungen generiert (vlg. Abschnitt 7.2.16).
3.: Diese Blöcke weisen den ausgewählten Kanälen die in Punkt 2 generierten Bezeichnungen zu.

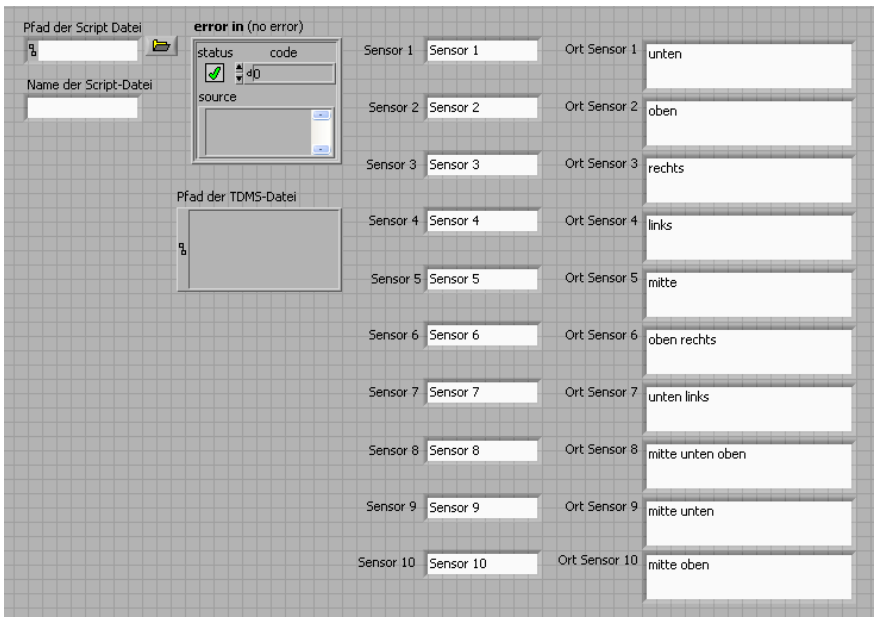


Abbildung 7.55.: Sub-VI für die Initialisierung von *DIAdem* (Frontpanel), Autor: René Pfeifer

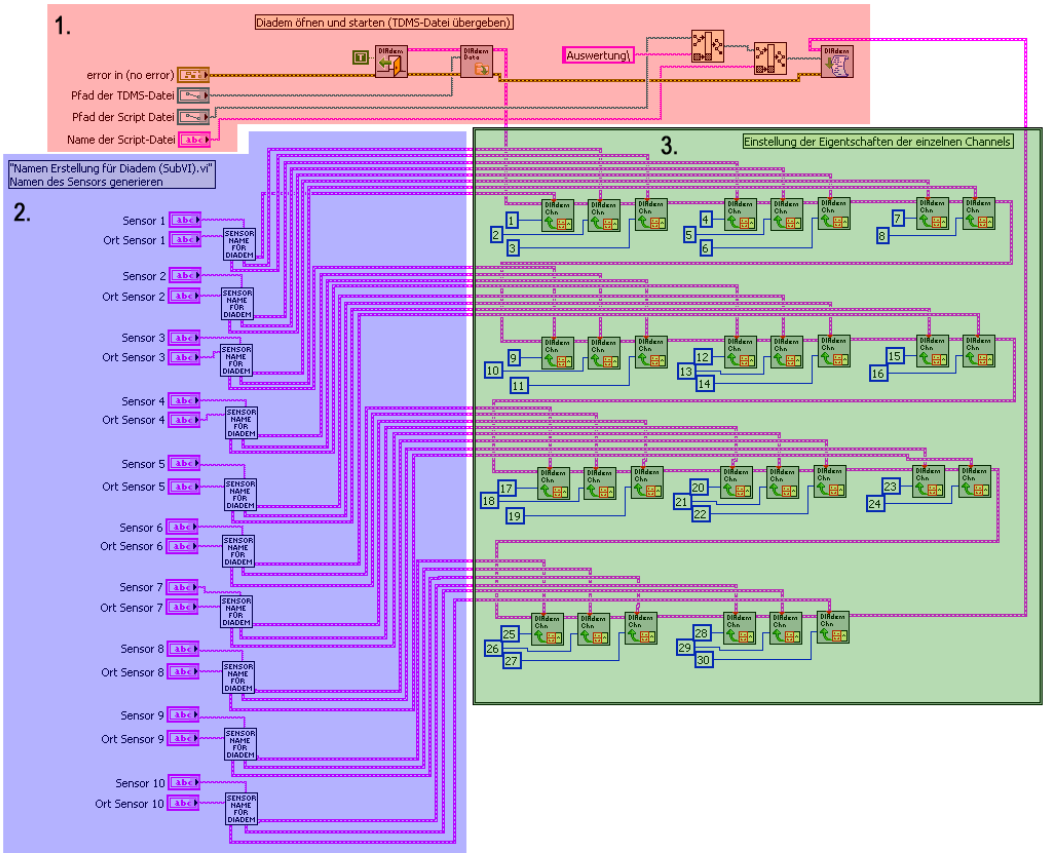


Abbildung 7.56.: Sub-VI für die Initialisierung von *DIAdem* (Blockdiagramm), Autor: René Pfeifer

7.2.18. Sensornamen mit XYZ erstellen(SubVI).vi

Dieses Sub-VI (siehe Abbildung 7.57 und 7.58) fügt den Sensornamen jeweils eine der Richtungen X, Y oder Z hinzu. Diese werden der *TDMS*-Datei direkt übergeben (vgl. Abschnitt 7.2.19) und sind somit dauerhaft gespeichert.

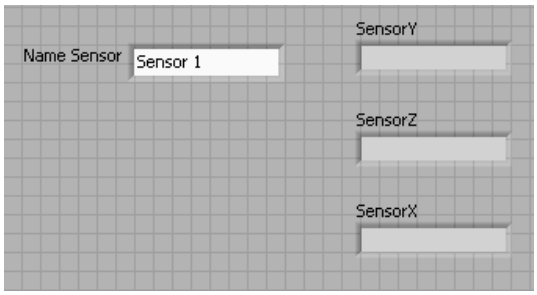


Abbildung 7.57.: Sub-VI zur Erstellung der Namen mit den Richtungsbuchstaben (Frontpanel), Autor: René Pfeifer

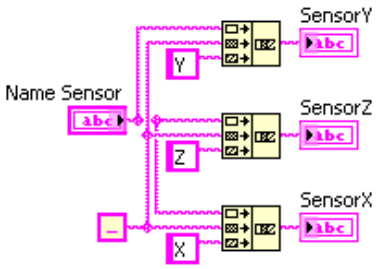


Abbildung 7.58.: Sub-VI zur Erstellung der Namen mit den Richtungsbuchstaben (Blockdiagramm), Autor: René Pfeifer

7.2.19. Sensornamen für die *TDMS*-Datei(SubVI).vi

Dieses Sub-VI (siehe Abbildung 7.59 und 7.60) bildet mit Hilfe des Sub-VIs aus Abschnitt 7.2.18 die Sensornamen aller Sensoren für die *TDMS*-Datei. Hier passiert im Grunde dasselbe wie in Abschnitt 7.2.16, außer, dass der Ort und die Einheit nicht an die *TDMS*-Datei angeschlossen werden können und daher nur die Sensornamen angepasst werden. Dies ist nötig, damit die Kanalnamen angezeigt werden, wenn die *TDMS*-Datei zu einem späteren Zeitpunkt mit *DIAdem* geöffnet wird.

Erklärung der Programmabschnitte in Abbildung 7.60:

1.: Hier werden zu den Sensornamen die jeweilige Richtung X, Y oder Z hinzugefügt. Siehe Abschnitt 7.2.18.
2.: Die generierten Namen werden zu einem String-Array verbunden.

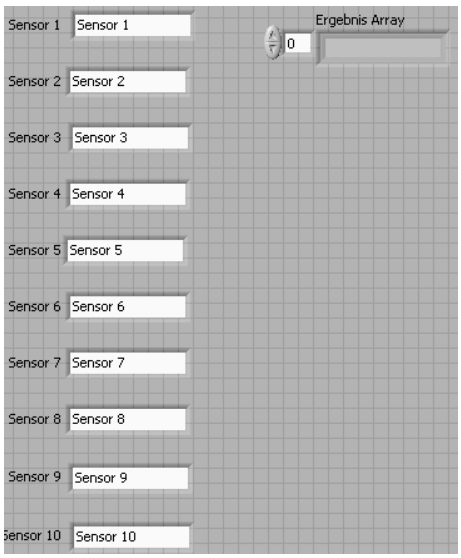


Abbildung 7.59.: Sub-VI für die Erstellung aller Sensornamen (Frontpanel), Autor: René Pfeifer

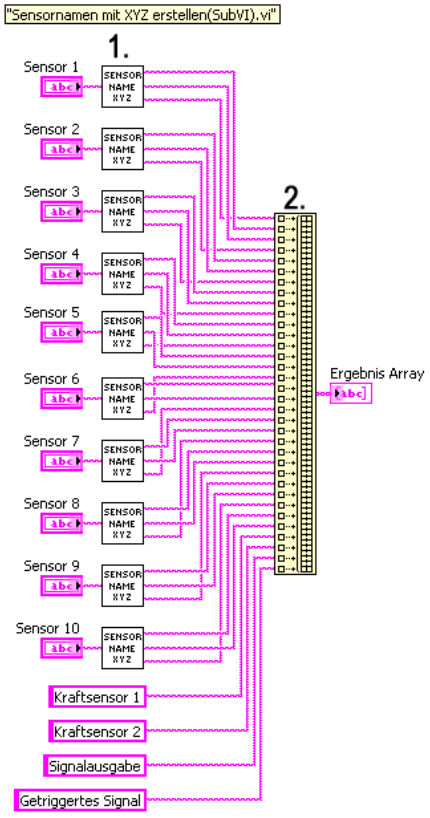


Abbildung 7.60.: Sub-VI für die Erstellung aller Sensornamen (Blockdiagramm), Autor: René Pfeifer

7.2.20. Tiefpassfilter(SubVI).vi

Hier werden die Messdaten tiefpassgefiltert (siehe Abbildung 7.61 und 7.62). Zu beachten ist, dass das Tiefpassfilter am Anfang eine kurze Verzögerung aufweist. Dieses ist darauf zurückzuführen, dass sich das Filter zunächst auf die Eingangsparameter einstellen muss.

Erklärung der Programmabschnitte in Abbildung 7.62:

1.: An den Eingängen „E Slot2“ bis „E Slot5“ werden die ungefilterten Signale eingelesen.
2.: Das ist der Tiefpassfilter, mit dem die Signale digital gefiltert werden. Das finite impulse response („FIR“)-Filter wird verwendet, obwohl dieses eine deutlich größere Ordnung besitzt, um dieselbe Steilheit wie ein infinite impulse response filter („IIR“)-Filter zu erlangen. Für den FIR-Filter spricht jedoch, dass dieses nicht instabil oder zu selbstständigen Schwingungen angeregt werden kann. Zudem ist dieses Filter geeigneter für Signale, deren Signalform erhalten bleiben muss, da bei dem IIR-Filter der Phasengang nicht linear ist (vlg. [23] und [21]).
3.: An den Ausgängen „A Slot2“ bis „A Slot5“ werden die gefilterten Signale ausgegeben.
4.: Hier werden die Error-Leitungen zusammengeführt und erst wenn alle Filter fertig sind, wird der dahinter liegende Programmcode ausgeführt.
5.: Parameter für den Tiefpassfilter. Als Filterfunktion („Topology“) wird „Windowed FIR“ gewählt, weil damit die besten Ergebnisse erzielt werden. Die Art („Type“) wird auf Tiefpassfilter („Lowpass“) gestellt. Die Grenzfrequenz ist für das periodische Signal fest auf 500 Hz eingestellt. Beim Rauschen ist es die „Maximale Frequenz im Rauschsignal“, die vom Bediener eingestellt wird. Der Rest der Parameter wird auf den Grundeinstellungen belassen.

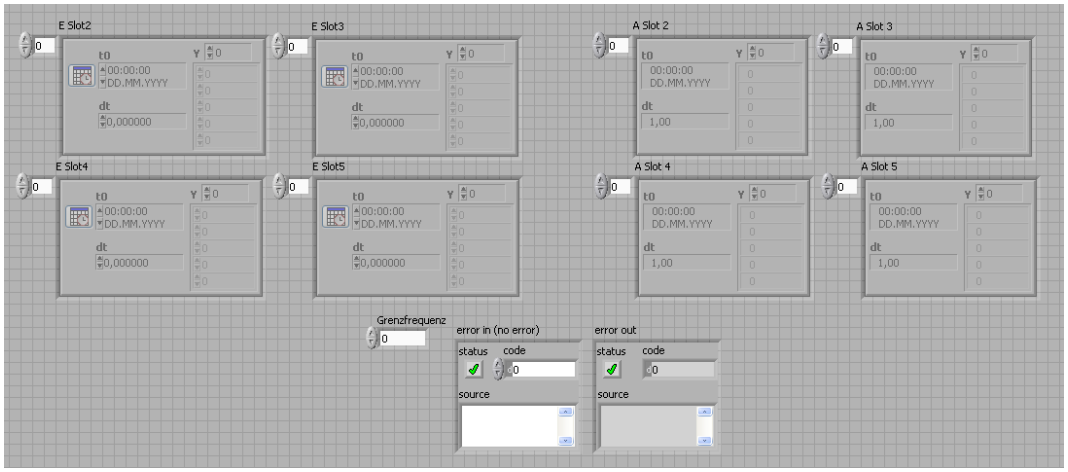


Abbildung 7.61.: Sub-VI für die Tiefpassfilterung der Messdaten (Frontpanel), Autor: René Pfeifer

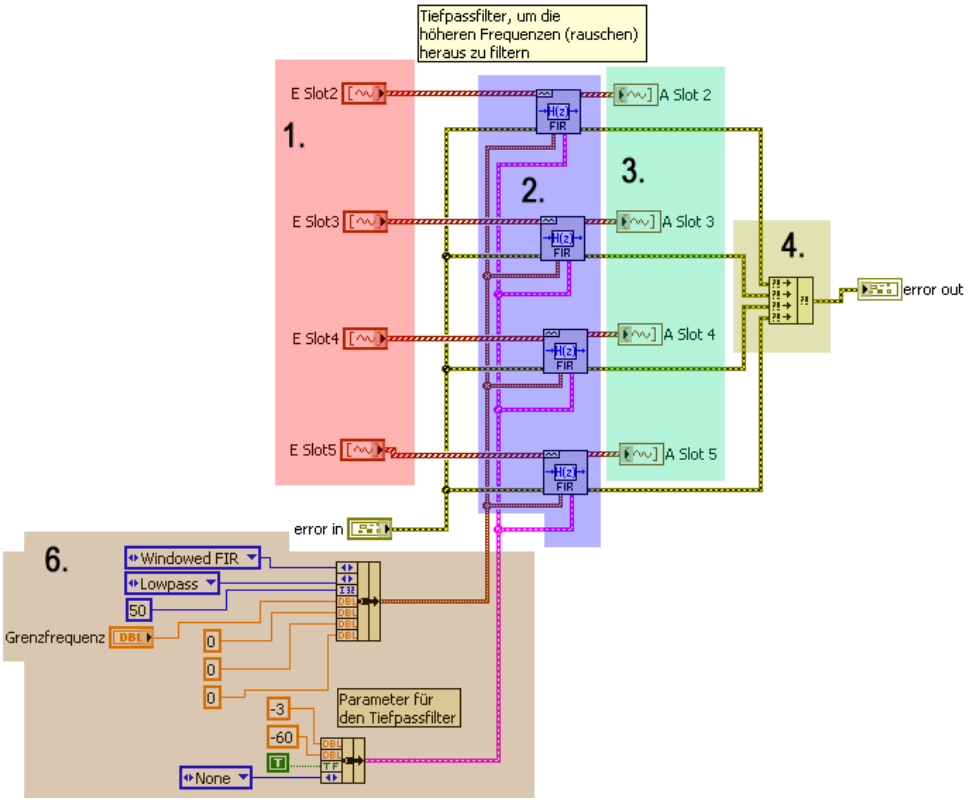


Abbildung 7.62.: Sub-VI für die Tiefpassfilterung der Messdaten (Blockdiagramm), Autor: René Pfeifer

7.2.21. Werte für Diadem (Werte.tdms Datei)(SubVI).vi

Dieses Sub-VI (siehe Abbildung 7.63 und 7.64) fasst alle benötigten Werte/Daten, welche für die Auswertung in *DIAdem* wichtig sind, für die spätere Speicherung in der Datei „Werte.tdms“ zusammen (siehe 8.1).

Erklärung der Programmabschnitte in Abbildung 7.64:

1.: Hier werden die Referenzen zu den Bedienelementen von Schrittweite, Startfrequenz und Endfrequenz übergeben.
2.: Dieser Teil dient dazu, *DIAdem* die aktuelle Uhrzeit, zu der die Messung gestartet wurde, numerisch zu übergeben.
3.: Hier werden die Anzahl der Sensoren, die Referenzen von „Rauschen aktiviert“ und der „Amplitude“ sowie die Einstellung des Charge Amplifiers, die maximale Frequenz im Rauschsignal und die Information, ob eine .pdf-Datei der Auswertung erstellt werden soll, übergeben.
4.: Der Block fasst alle Werte zusammen, sodass diese in die *TDMS*-Datei geschrieben werden können. Die Reihenfolge dieser Daten ist eine Vereinbarung mit dem *DIAdem*-Script (siehe Kapitel 8) und darf nicht geändert werden.

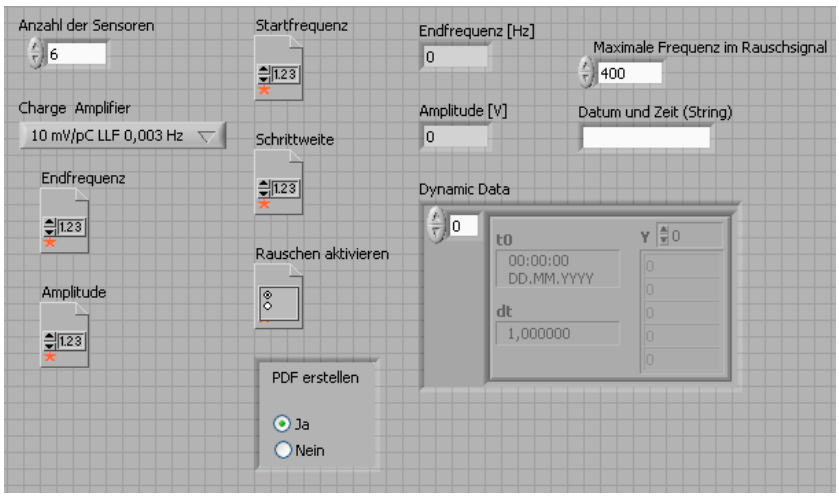


Abbildung 7.63.: Sub-VI für die Werte für *DIAdem* (Frontpanel), Autor: René Pfeifer

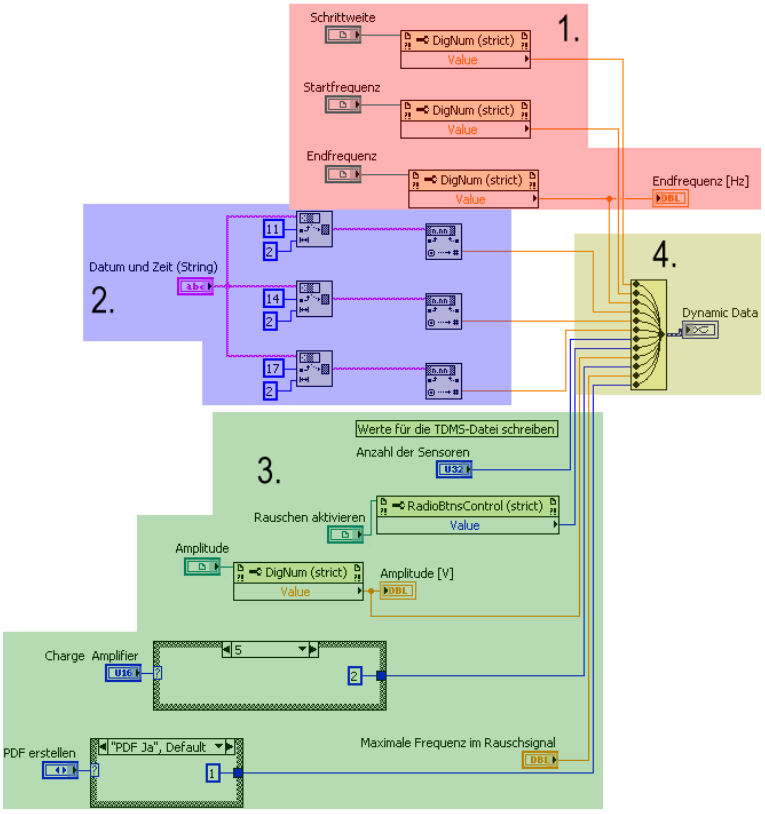


Abbildung 7.64.: Sub-VI für die Werte für *DIAdem* (Blockdiagramm), Autor: René Pfeifer

7.2.22. Ausgangskarte für das gewählte Signal(SubVI).vi

Das Sub-VI (siehe Abbildung 7.65 und 7.66) parametriert den virtuellen Kanal zur Ausgangskarte, welche die gewählte Signalform kontinuierlich über die Ausgangskarte an

den Verstärker gibt.

Erklärung der Programmabschnitte in Abbildung 7.66:

1.: Öffnen eines virtuellen Output-Kanals zur gewählten Messkarte (in diesem Fall /PXI1Slot5/ao0 bei Eingangskanal) für die Ausgabe von Spannung („Voltage“). Dazu wird für die Ausgangskonfiguration „Pseudodifferential“ gewählt. Das hat den Grund, dass die Messart für Beschleunigungssensoren üblich ist und die Messkarten diese ausschließlich unterstützen. Zudem muss der Minimal- und Maximalwert der Erfassung in Volt übergeben werden.
2.: Das Timing ist hier standardmäßig auf „Onboard-Clock“ gestellt und die Messart des Messkanals für den kontinuierlichen Betrieb eingestellt. Zudem wird hier die Samplerate[2] übergeben.
3.: Der Knotenpunkt dient der erweiterten Einstellung dieses Messkanals. Mit der Einstellung „Do Not Allow Regeneration“ wird dem Ausgang ausdrücklich verboten, alte Daten auszugeben, wodurch immer die aktuellsten Daten ausgegeben werden.

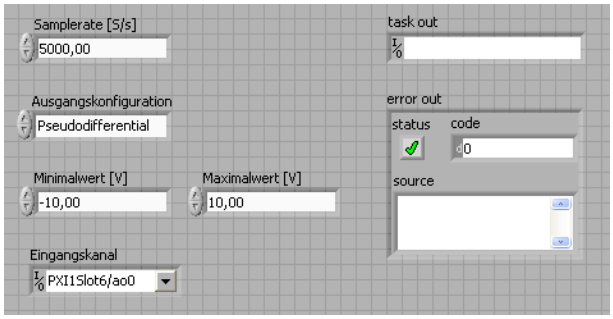


Abbildung 7.65.: Sub-VI zur Ausgabe des gewählten Signals (Frontpanel), Autor: René Pfeifer

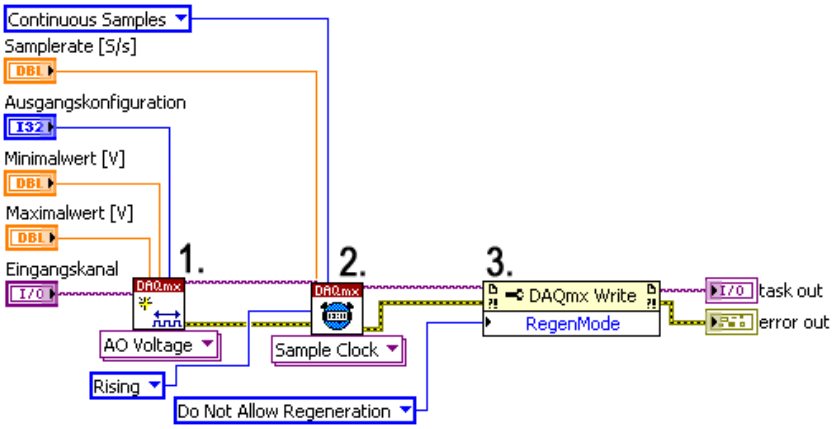


Abbildung 7.66.: Sub-VI zur Ausgabe des gewählten Signals (Blockdiagramm), Autor: René Pfeifer

7.2.23. Eingangskarte für das gewählte Signal(SubVI).vi

In diesem Sub-VI findet die Parametrierung der Eingangskarte zur kontinuierlichen Erfassung des Signals, welches den Shaker anregt, statt (siehe Abbildung 7.67 und 7.68). Dieses Signal wird benötigt, um darauf triggern zu können.

Erklärung der Programmabschnitte in Abbildung 7.68:

1.: Öffnen eines virtuellen Input-Kanals zur gewählten Messkarte (in diesem Fall /PXI1Slot5/ai0 bei Eingangskanal) für die Erfassung von Spannung („Voltage“). Die restlichen Einstellungen entsprechen denen aus Punkt 1 des Abschnittes 7.2.22.
2.: Die Einstellungen entsprechen denen aus Punkt 2 des Abschnittes 7.2.22.
3.: Hier wird festgelegt, unter welchen Bedingungen der Starttrigger reagieren soll. In diesem Fall wird auf das Eintreten des Signals in ein Fenster getriggert.
4.: Dieser Block dient dazu, das Triggersignal zu exportieren und es z.B. als Starttrigger zu kennzeichnen.

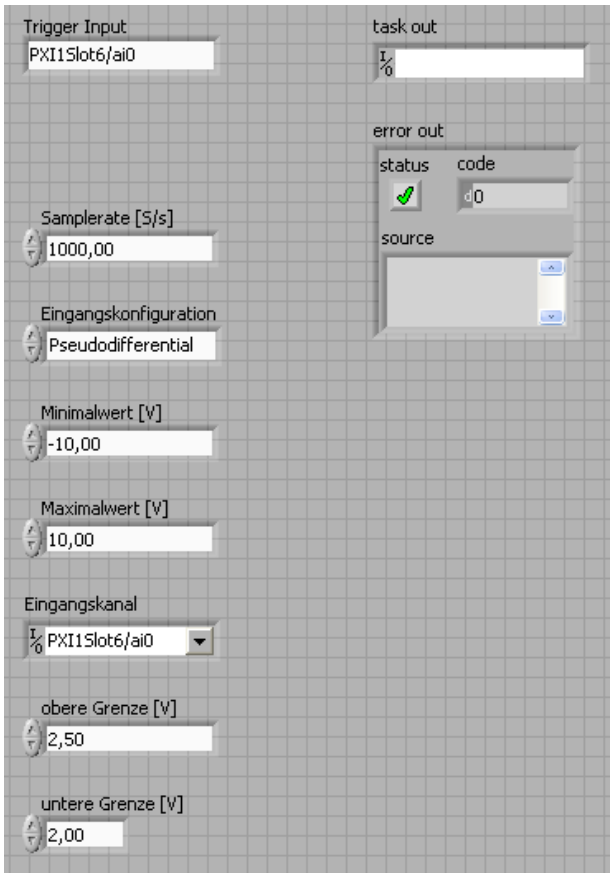


Abbildung 7.67.: Sub-VI zum Einlesen des anregenden Signals (Frontpanel), Autor: René Pfeifer

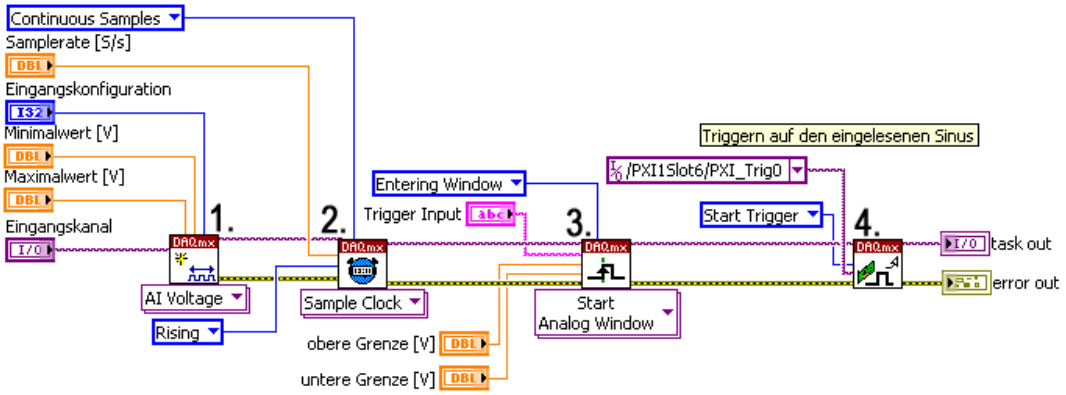


Abbildung 7.68.: Sub-VI zum Einlesen des anregenden Signals (Blockdiagramm), Autor: René Pfeifer

7.2.24. Sensoren syncSlot2(SubVI).vi

Dieses Sub-VI (siehe Abbildung 7.69 und 7.70) parametriert die Karte im Slot 2, an der nur Sensoren angeschlossen sind, für die kontinuierliche Erfassung von Messdaten. Die Messkarte in Slot 2 ist der Master, die Messkarten in den Slots 3 bis 5 sind die zugehörigen Slaves.

Erklärung der Programmabschnitte in Abbildung 7.70:

1.: Öffnen von virtuellen Input-Kanälen zur gewählten Messkarte (in diesem Fall /PXI1Slot2/ai0:ai7 bei Eingangskanal) für die Erfassung von Beschleunigungen („Accelerometer“). Die Einheit der erfassten Größe ist auf m/s^2 eingestellt und wird automatisch umgewandelt. Die Sensorempfindlichkeit ist auf 1000 (siehe Datenblatt im Anhang B) mit der Messeinheit mVolts/g eingestellt. Die Sensoren werden durch die Stromversorgung (IEPE) mit den benötigten 0,004 A versorgt. Der Anschluss „Internal“ besagt, dass die Versorgung von der Karte übernommen wird. Die restlichen Einstellungen entsprechen denen aus Punkt 1 des Abschnittes 7.2.22.
2.: Die Einstellungen entsprechen denen aus Punkt 2 des Abschnittes 7.2.22.
3.: Der Triggerblock wartet auf das in Abschnitt 7.2.23 exportierte Triggersignal und gibt das Startsignal zur Aufnahme von Messwerten.

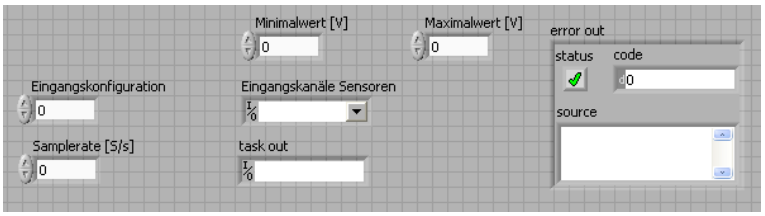


Abbildung 7.69.: Sub-VI zur Parametrierung der Messkarte im Slot 2 mit Sensoren für den kontinuierlichen Betrieb (Frontpanel), Autor: René Pfeifer

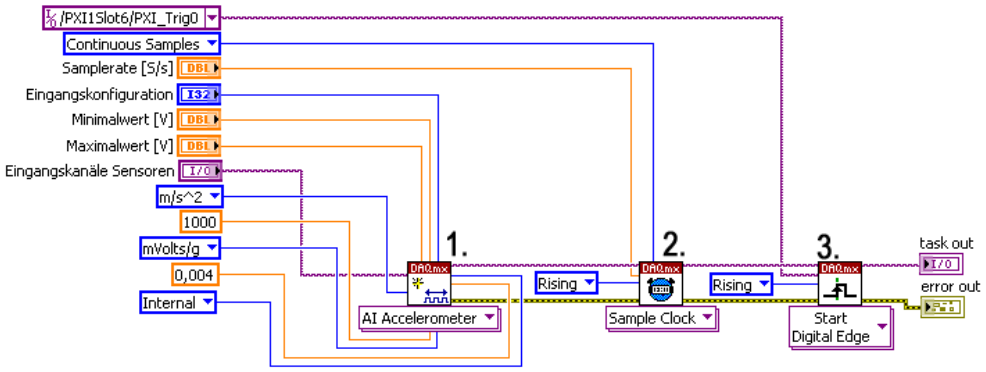


Abbildung 7.70.: Sub-VI zur Parametrierung der Messkarte im Slot 2 mit Sensoren für den kontinuierlichen Betrieb (Blockdiagramm), Autor: René Pfeifer

7.2.25. Sensoren sync(SubVI).vi

Das Sub-VI (siehe Abbildung 7.71 und 7.72) parametriert die Karten in den Slots 3 und 4, an denen nur Sensoren angeschlossen sind, für die kontinuierliche Erfassung von Messdaten. Hier werden die Messkarten zudem noch mit der Master-Messkarte aus Slot 2 synchronisiert.

Erklärung der Programmabschnitte in Abbildung 7.72:

1.: Öffnen von virtuellen Input-Kanälen zu den gewählten Messkarten (in diesem Fall /PXI1Slot3/ai0:ai7 und /PXI1Slot4/ai0:ai7 bei Eingangskanal) für die Erfassung von Beschleunigungen („Accelerometer“). Die restlichen Einstellungen entsprechen denen aus Punkt 1 des Abschnittes 7.2.22 und Punkt 1 des Abschnittes 7.2.24.
2.: Die Einstellungen entsprechen denen aus Punkt 2 des Abschnittes 7.2.22.
3.: Der Knotenpunkt dient dazu, die beiden Messkarten in den Slots 3 und 4 mit der Messkarte aus dem Slot 2 zu synchronisieren, sodass sie zum gleichen Takt Daten erfassen. Genau dies ist der Unterschied zu dem Sub-VI aus Abschnitt 7.2.24. Dabei dient „Sample Clock Timebase Source“ dazu, festzulegen, von welcher Karte die „Sample Clock Timebase“ verwendet werden soll. „Synchronization Pulse Source“ legt fest, von welcher Karte der Synchronisationsimpuls genutzt werden soll.
4.: Die Einstellungen entsprechen denen aus Punkt 3 des Abschnittes 7.2.24.

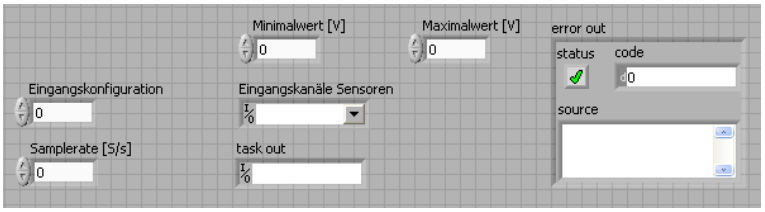


Abbildung 7.71.: Sub-VI zur Parametrierung der Messkarten in den Slots 3 und 4 mit Sensoren für den kontinuierlichen Betrieb (Frontpanel), Autor: René Pfeifer

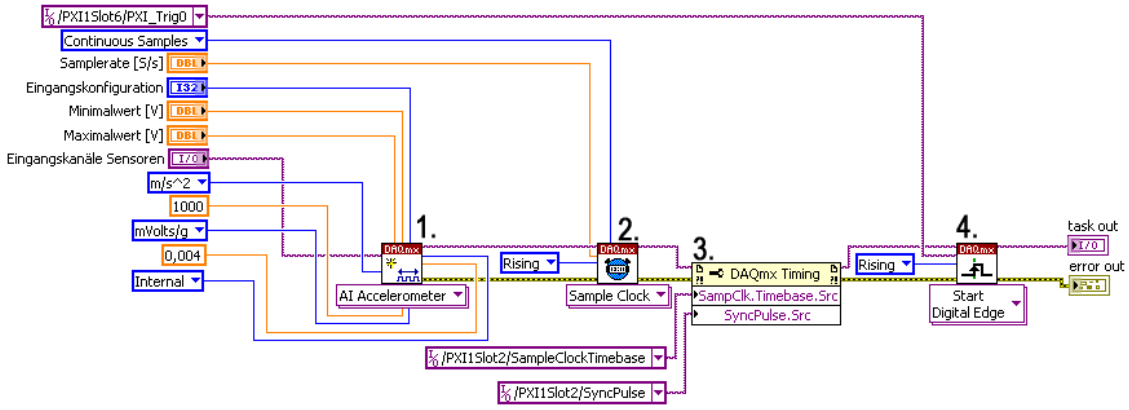


Abbildung 7.72.: Sub-VI zur Parametrierung der Messkarte in den Slots 3 und 4 mit Sensoren für den kontinuierlichen Betrieb (Blockdiagramm), Autor: René Pfeifer

Die Synchronisation der Messkarten wird durch das Anlegen des selben Signals an zwei Eingängen zweier Messkarten überprüft. Ein Beispiel zweier Signale im Modul VIEW in *DIAdem* ist in den Abbildungen 7.73 bis 7.75 zu sehen. Dieser Test wird mit allen Karten durchgeführt. Es können immer nur zwei Messkarten gleichzeitig getestet werden, da es hardwaretechnisch nicht anders möglich ist.
Zum Testen wird eine Gleichspannungsquelle verwendet, an der permanent die Spannung verändert wird. Dieses wird gemacht, um im Gegensatz zu einem reinen Sinus Knicke im Verlauf zu erhalten.

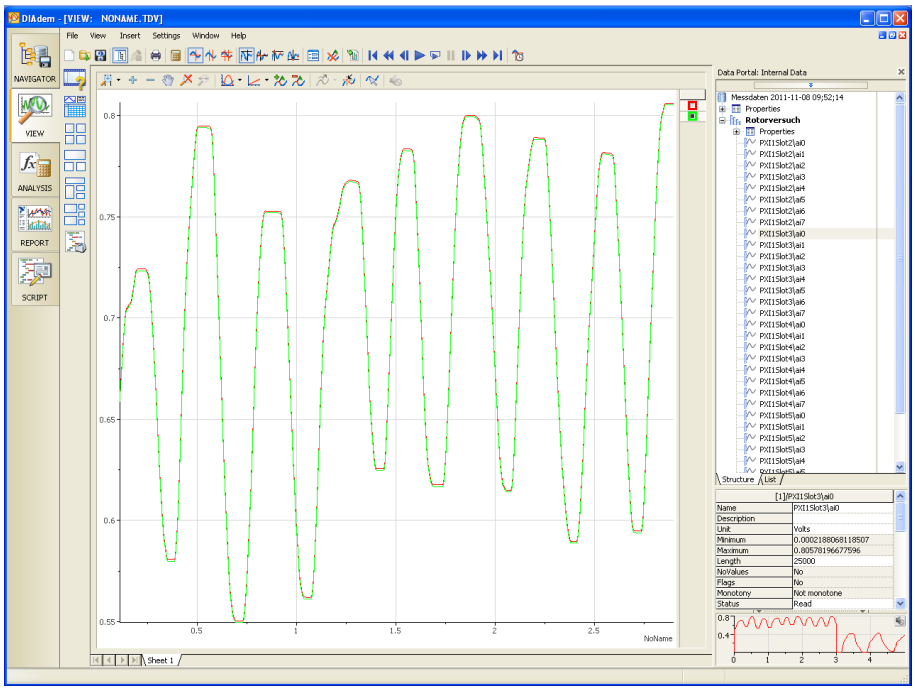


Abbildung 7.73.: Aufnahme des gleichen Signals mit zwei Messkarten, Autor: René Pfeifer

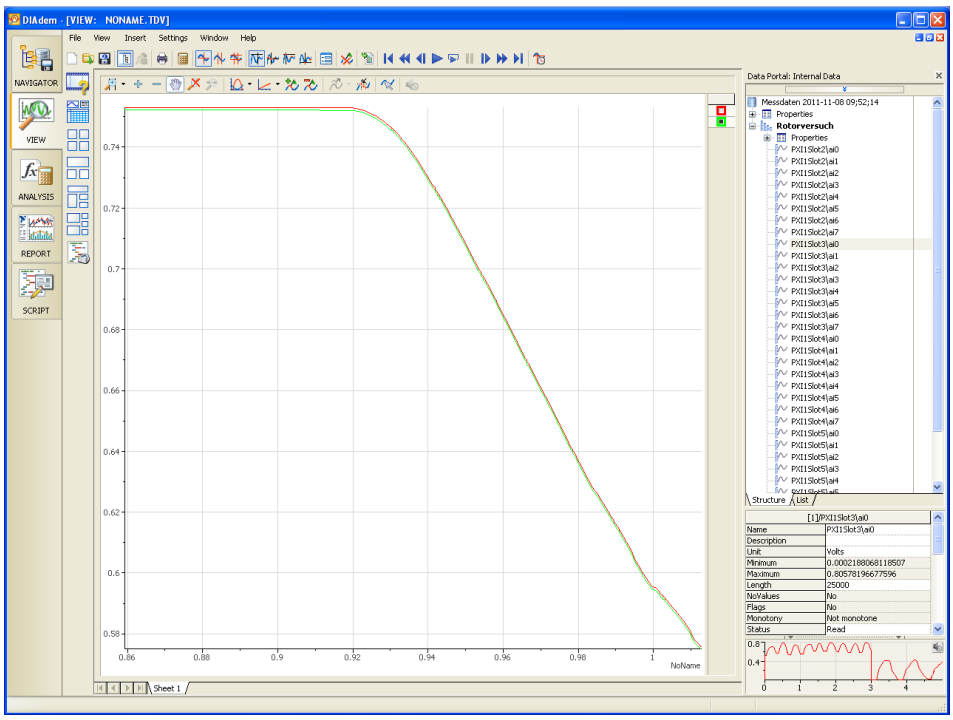


Abbildung 7.74.: Erster Ausschnitt der Aufnahme des gleichen Signals mit zwei Messkarten, Autor: René Pfeifer

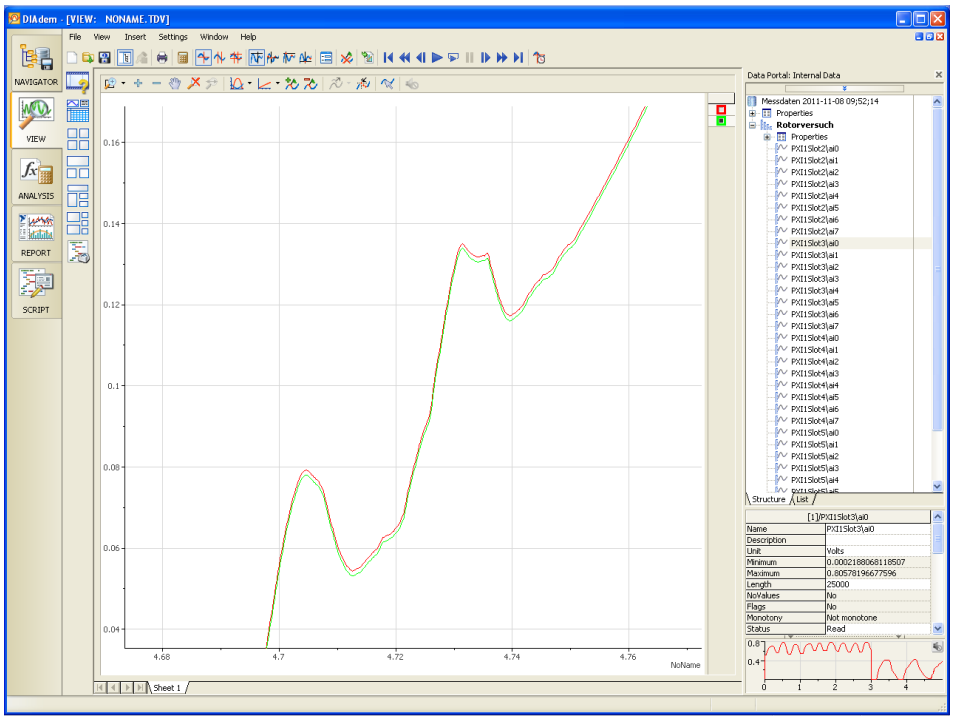


Abbildung 7.75.: Zweiter Ausschnitt der Aufnahme des gleichen Signals mit zwei Messkarten, Autor: René Pfeifer

Zur genaueren Betrachtung wird in die aufgenommen Signale weiter hineingezoomt. Dadurch ist eindeutig erkennbar, dass die Werte genau zum selben Zeitpunkt (synchron) eingelesen werden. Dies lässt sich daran erkennen, dass die markanten Punkte, wie z.B. ein Knick, zeitlich direkt übereinander liegen (siehe Abbildung 7.76).
Es ist weiterhin zu erkennen, dass ein kleiner Versatz in Ordinaten-Richtung der beiden erfassten Signale vorhanden ist. Dies ist auf den eigenen minimalen Offset des Eingangs der jeweiligen Messkarte zurückzuführen und liegt unter der geforderten Messgenauigkeit.

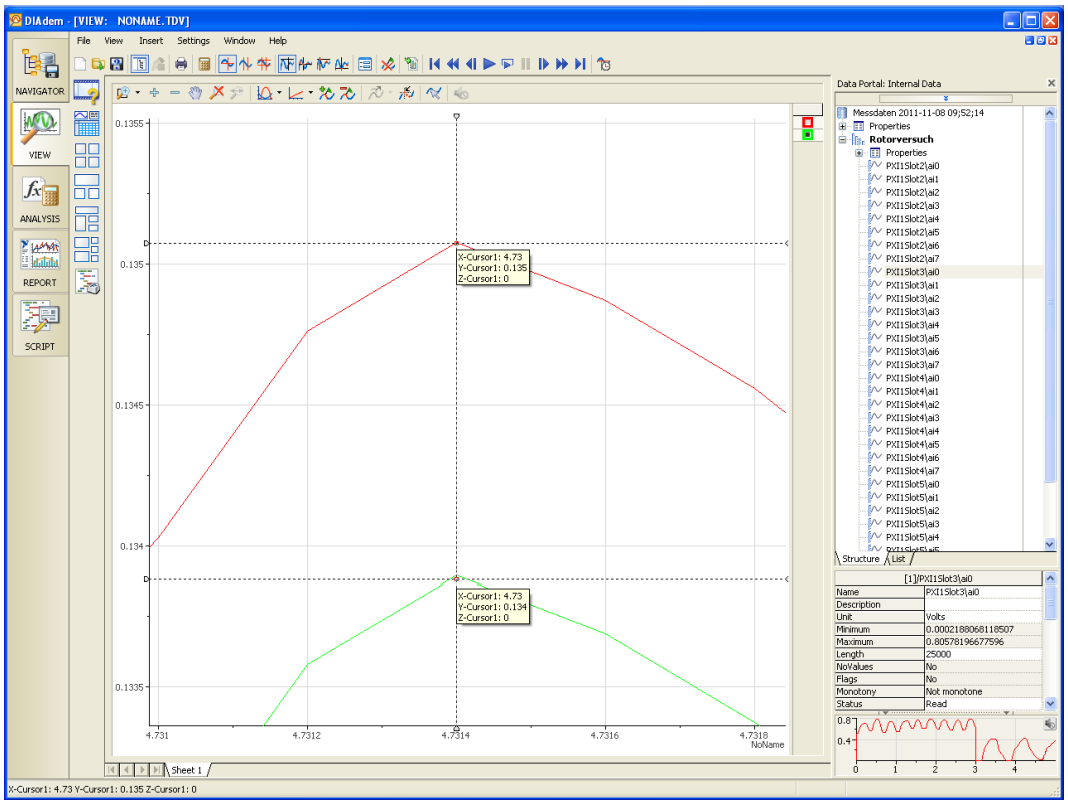


Abbildung 7.76.: Ordinaten-Werte der beiden Signale, Autor: René Pfeifer

7.2.26. Sensoren + Kraftsensoren sync(SubVI).vi

In diesem Sub-VI (siehe Abbildung 7.77 und 7.78) wird die Karte im Slot 5, an der die Sensoren und der Kraftsensor angeschlossen sind, für die kontinuierliche Erfassung von Messdaten parametriert. Zudem werden die Messkarten noch mit der Master-Messkarte aus Slot 2 synchronisiert.

Erklärung der Programmabschnitte in Abbildung 7.78:

1.: Öffnen von virtuellen Input-Kanälen zur gewählten Messkarte (in diesem Fall /PXI1Slot5/ai0:ai5 bei Eingangskanal) für die Erfassung von Beschleunigungen

(„Accelerometer“). Die restlichen Einstellungen entsprechen denen aus Punkt 1 des Abschnittes 7.2.22 und Punkt 1 des Abschnittes 7.2.24.
2.: Öffnen von virtuellen Input-Kanälen zur gewählten Messkarte (in diesem Fall /PXI1Slot5/ai6:ai7) für die Erfassung von Spannung („Voltage“).
3.: Die Einstellungen entsprechen denen aus Punkt 2 des Abschnittes 7.2.24.
4.: Die Einstellungen entsprechen denen aus Punkt 3 des Abschnittes 7.2.25.
5.: Die Einstellungen entsprechen denen aus Punkt 4 des Abschnittes 7.2.25.

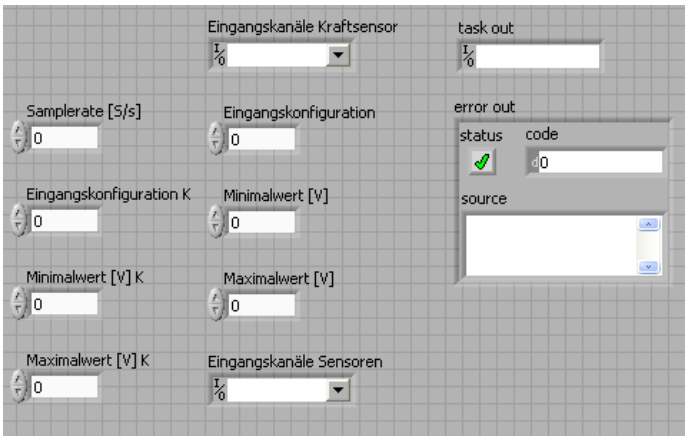


Abbildung 7.77.: Sub-VI zur Parametrierung der Messkarte mit Sensoren + Kraftsensoren für den kontinuierlichen Betrieb (Frontpanel), Autor: René Pfeifer

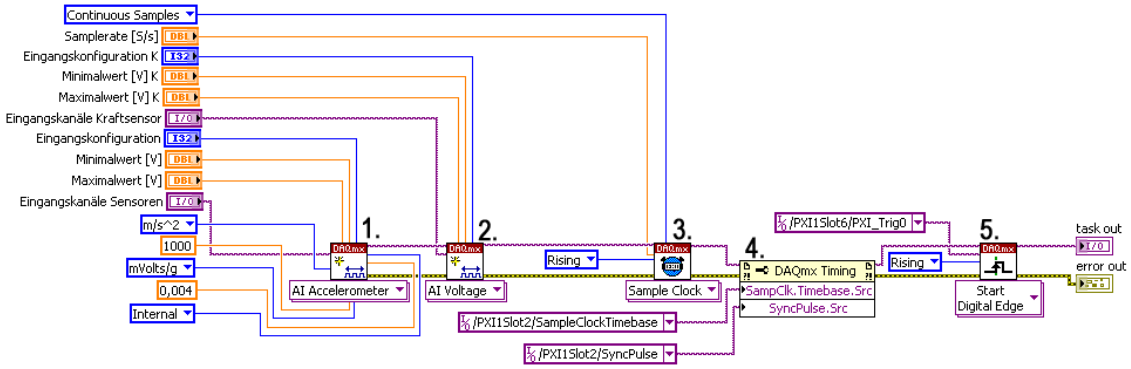


Abbildung 7.78.: Sub-VI zur Parametrierung der Messkarte mit Sensoren + Kraftsensoren für den kontinuierlichen Betrieb (Blockdiagramm), Autor: René Pfeifer

7.3. Erstellung von „Stand-alone-Applikationen“ (.exe) mit dem *LabVIEW Application Builder*

Mit dem *LabVIEW Application Builder* können von VIs direkt ausführbare Dateien (.exe-Dateien) erstellt werden. Diese können ohne das Programm *LabVIEW* gestartet werden. Dazu

wird nur die *LabVIEW Run-Time Engine 2010*, welche unter [14] kostenlos herunterzuladen ist, benötigt. Damit ist es möglich, diese Applikation auch auf anderen Rechnern auszuführen, ohne *LabVIEW* installiert zu haben. Der einzige Nachteil an der .exe-Datei ist, dass der Programmcode dort nicht mehr einsehbar bzw. veränderbar ist.
Des Weiteren muss darauf geachtet werden, dass die Pfade auf Dateien geändert werden. So wird z.B. aus dem Pfad „C:\...\...\Beispiel.vi“ „C:\...\...\Beispiel.exe\Beispiel.vi“. Aus diesem Grund mussten bei den folgenden Programmteilen (siehe Abbildungen 7.79 bis 7.81) ein weiteres Element aus dem Pfad des aktuellen VIs entfernt werden.

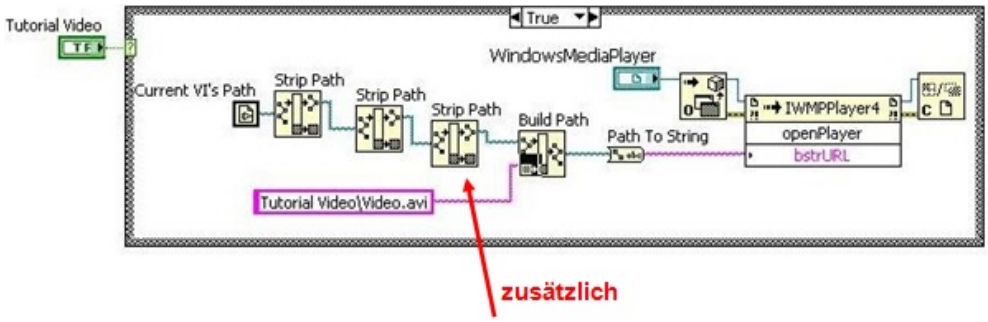


Abbildung 7.79.: Sub-VI zum Öffnen des Tutorial Videos für die .exe-Datei (Blockdiagramm), Autor: René Pfeifer

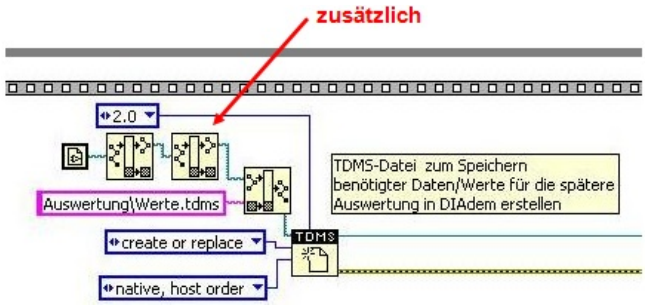


Abbildung 7.80.: Programmabschnitt 3, Öffnen der „Werte.tdms“-Datei für die .exe-Datei (Blockdiagramm), Autor: René Pfeifer

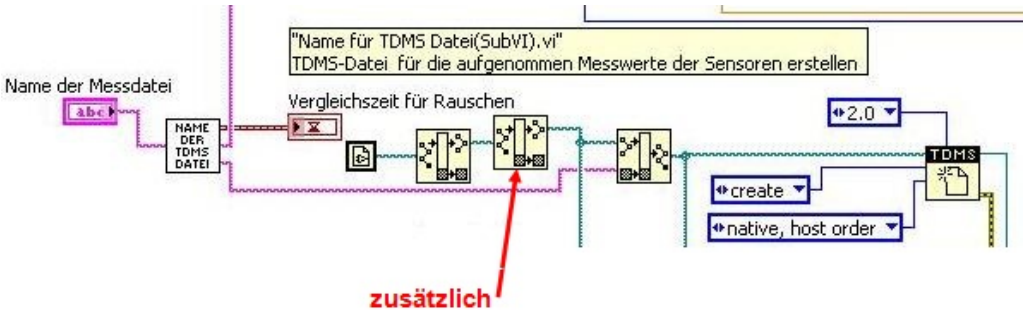


Abbildung 7.81.: Programmabschnitt 4, Öffnen der „Messdaten.tdms“-Datei für die .exe-Datei (Blockdiagramm), Autor: René Pfeifer

Erklärung des Vorgehens zur Erstellung einer „Stand-alone-Applikationen“:

Zuerst muss, falls dies noch nicht geschehen ist, von dem vorhanden VI ein Projekt erstellt und gespeichert werden (siehe Abbildung 7.82). Zudem werden alle Sub-VIs, die in dem VI enthalten sind, in das Projekt geladen.

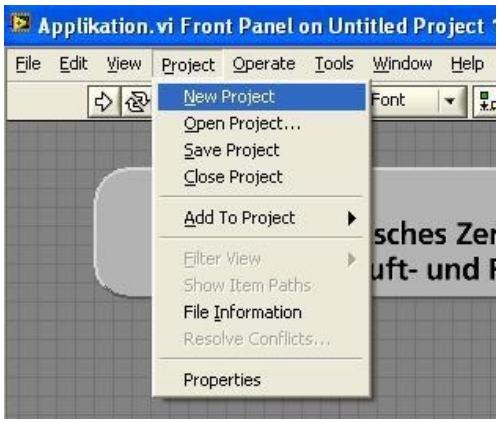


Abbildung 7.82.: Neues Projekt anlegen, Autor: René Pfeifer

Ist das Projekt gespeichert, kann die .exe-Datei unter dem Punkt „Build Specification“ erstellt werden (siehe Abbildung 7.83).

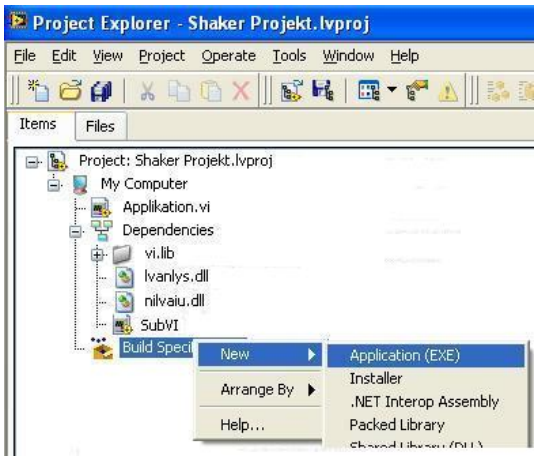


Abbildung 7.83.: Erstellen der .exe-Datei, Autor: René Pfeifer

Nun können Einstellungen für die .exe-Datei (wie z.B. Name und Speicherort) vorgenommen werden (siehe Abbildung 7.84). Sind alle Einstellungen eingegeben, wird über den Button „Build“ die .exe-Datei erstellt.

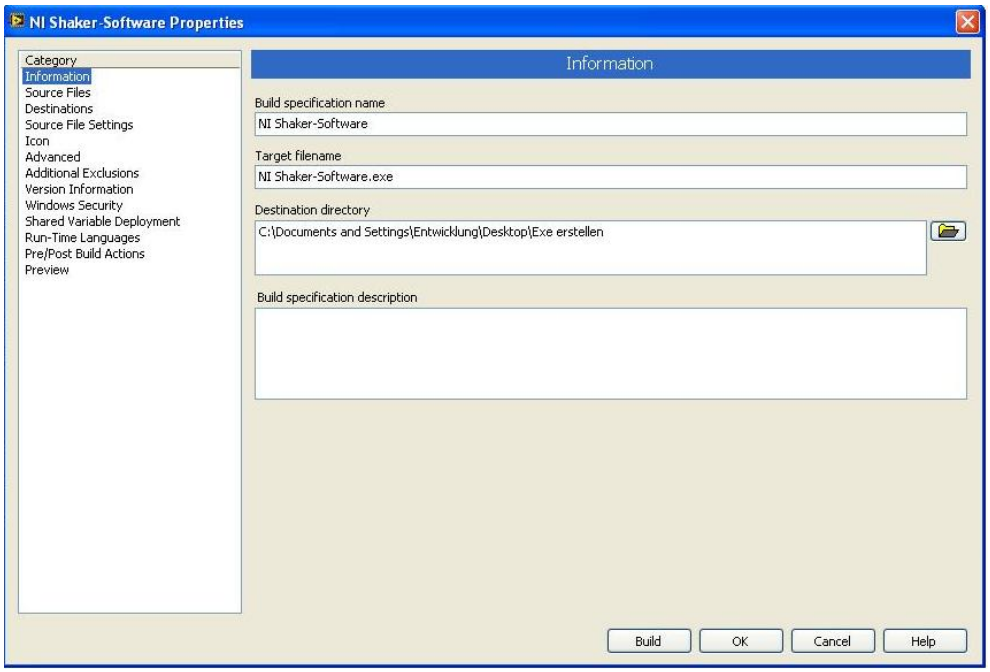


Abbildung 7.84.: Speicherort und Name der .exe-Datei eingeben, Autor: René Pfeifer

Die fertige Ordnerstruktur dieser Bachelorarbeit ist in Abbildung 7.85 zu sehen.

Abbildung 7.85.: Ordnerstruktur, Autor: René Pfeifer

8. Das *DIAdem*-Programm

DIAdem ist speziell darauf ausgelegt, *TMDS*-Dateien (siehe Kapitel 6) schreiben und lesen zu können. Da die Messdaten in *LabVIEW* in eine *TDMS*-Datei geschrieben werden, ist es also kein Problem, diese mit *DIAdem* zu bearbeiten und auszuwerten.
Die Auswertung soll automatisch geschehen. Daher wird mit dem Modul SCRIPT (siehe Kapitel 3.2.3) gearbeitet. Dort wird ein Programm erstellt, das die Messdaten auswertet und anschließend einen Bericht erstellt. Zum Erstellen des Programms wurde unter anderem die *DIAdem*-Hilfe (siehe [5]) verwendet.

8.1. Berechnungen

Die Messungen dienen dazu, Resonanzfrequenzen zu finden. Aus diesem Grund sollen mit diesem Programm aus den Messwerten FFTs bzw. FRFs („Frequenzantworten“) berechnet werden. Aus diesen lassen sich die Resonanzfrequenzen ablesen.
Bei der FFT handelt es sich um eine mathematische Auswertefunktion, welche zeitabhängige Signale in den Frequenzbereich transformiert. Durch diese Funktion lässt sich feststellen, welche Frequenz wie stark innerhalb eines Signals enthalten ist (vgl. [4], Seite 2). Mit der FRF erreicht man eine eindeutige Beschreibung der Eigenschaften eines Systems (siehe [3], Seite 13). Sie ist das Verhältnis von Ausgang zu Eingang als Funktion der Frequenz, das heißt sie ist das Verhältnis der Fourier Transformierten des Ausgangs zur Fourier Transformierten des Eingangs (siehe (8.1)).

$$H(f) = \frac{A(f)}{E(f)} = \frac{FFT\{A(t)\}}{FFT\{E(t)\}} \tag{8.1}$$

Im Folgenden soll nun das Programm detailliert erläutert werden.

In Abbildung 8.1 ist der Anfang des Programms zu sehen. Als erstes steht der Befehl „Option Explicit“. Dieser erzwingt die explizite Deklaration aller Variablen in einem Skript. Hierdurch werden zum Beispiel Tippfehler bei Namen schon existierender Variablen vermieden.
Als nächstes wird die Datei „Werte.tdms“ geöffnet. In diese werden in *LabVIEW* alle Informationen geschrieben, die neben den eigentlichen Messwerten noch zur Auswertung benötigt werden.

```
Option Explicit  'Erzwingt die explizite Deklaration aller Variablen in einem Skript.

'Öffnen der Datei "Werte.tdms":
'-------------------------------------------------------------
DIM pfad
pfad = CurrentScriptPath  'aktuellen Script-Pfad verwenden
Call DataFileLoad(pfad&"Werte.tdms","","")  'öffnet die Datei "Werte", in der sich
                                            'die Startfrequenz usw. befinden (Chan0)
'-------------------------------------------------------------
```

Abbildung 8.1.: Anfang des Programms (Codezeilen 7 – 15), Autorin: Melanie Schulze

Diese Datei enthält beispielsweise die Start- und Endfrequenz, die Schrittweite und die Anzahl der verwendeten Sensoren (siehe Abbildung 8.2).

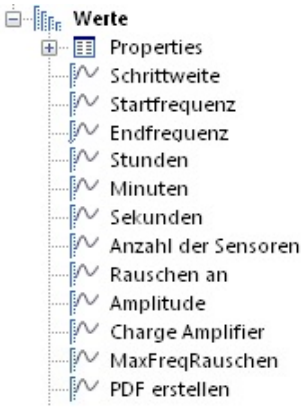


Abbildung 8.2.: Geöffnete „Werte.tdms“- Datei im Datenportal von *DIAdem*, Autorin: Melanie Schulze

Als nächstes erfolgt die Deklaration von Variablen, die innerhalb des Programms an mehreren Stellen benötigt werden (siehe Abbildung 8.3).
Der Befehl „DIM“ ist dafür da, Variablen zu deklarieren und Speicherplatz für diese zu reservieren. Schreibt man hinter den Variablennamen eine Zahl in Klammern (wie hier „sensorrichtung“, „ort_sensoren“ und „name_sensoren“), so wird ein Array entsprechender Länge erzeugt.

```
'Variablendeklaration:
'-------------------------------------------------------------
DIM i, j, k, l, m, n, o, p  'Schleifenzähler bzw. anders verwendete Zähler
DIM name_messung, schrittweite, startfrequenz, endfrequenz, anz_freq, anz_sensoren
DIM rauschen, amplitude_anregung, charge_amplifier0, sensitivity0, empfindlichkeit_kraft
DIM aktuelles_Datum, stunden, minuten, sekunden, uhrzeit, f_rauschen, pdf, channel_laenge
DIM sensorrichtung(3), ort_sensoren(10), name_sensoren(10)
'-------------------------------------------------------------
```

Abbildung 8.3.: Variablendeklaration (Codezeilen 17 – 26), Autorin: Melanie Schulze

In Abbildung 8.4 ist die Initialisierung der Variablen zu sehen.
Mit dem Befehl „CHD()“ können einzelne Werte eines bestimmten Kanals (vgl. Kapitel 6) ausgelesen werden. Dabei entspricht der erste Parameter der Stelle innerhalb des Kanals, von der der Wert ausgelesen werden soll. Der zweite Parameter, der aus einer Kombination aus Gruppe und Kanal besteht, enthält den Kanal, aus dem ausgelesen werden soll. Allgemein ist zu beachten, dass Parameter stets mit einem Komma getrennt werden. Der Aufbau [X]/[Y] wird im Folgenden oft verwendet. An der ersten Stelle (hier X) steht die Nummer der Gruppe, an der zweiten Stelle (hier Y) steht die Nummer des Kanals innerhalb dieser Gruppe. Die hier verwendete Gruppe 2 entspricht der Gruppe „Werte“ aus Abbildung 8.2.
Das Array „sensorrichtung“ wird mit den Buchstaben X, Y und Z initialisiert und steht für die drei Richtungen, die jeder Sensor besitzt.
Aus den Werten der Schrittweite, der Start- und der Endfrequenz wird die Anzahl der Frequenzen berechnet. Dieses geschieht jedoch nur, wenn es sich um eine Messung handelt, bei der mit einem Frequenzsweep angeregt wurde. Wurde mit Rauschen angeregt, wird die Anzahl auf 1 gesetzt, da dann keine Frequenzen gesweept wurden.

```
name_messung = GroupName(ChnGroup(1))  'Der Name der Messung entspricht dem Gruppennamen
schrittweite = CHD(1,"[2]/[1]")  '[Group]/[Channel]
startfrequenz = CHD(1,"[2]/[2]")
endfrequenz = CHD(1,"[2]/[3]")
if rauschen = 1 then
  anz_freq = 1  'Anzahl der Frequenzen auf 1 setzen, da beim
                'Rauschen keine Frequenzen gesweept werden.
else
  anz_freq = (endfrequenz - startfrequenz) / schrittweite + 1
end if
anz_sensoren = CHD(1,"[2]/[7]")
rauschen = CHD(1,"[2]/[8]")
amplitude_anregung = CHD(1,"[2]/[9]")
f_rauschen = CHD(1,"[2]/[11]")
pdf = CHD(1,"[2]/[12]")
sensorrichtung(1) = "X"
sensorrichtung(2) = "Y"
sensorrichtung(3) = "Z"
```

Abbildung 8.4.: Initialisieren der Variablen (Codezeilen 28 – 45), Autorin: Melanie Schulze

Der Anwender hat im *LabVIEW*-Programm die Möglichkeit, der Messung einen Namen zu geben. Dieser Name wird als Gruppenname der Messdaten gespeichert (siehe Abbildung 8.5).

Abbildung 8.5.: Gruppenname als Name der Messung, Autorin: Melanie Schulze

Mit dem Befehl „GroupName()“ kann man diesen Namen auslesen. Da die Messdaten in der ersten Gruppe liegen, muss in Klammern „ChnGroup(1)“ für die erste Kanalgruppe angegeben werden (siehe Abbildung 8.4).

Als nächstes werden die Empfindlichkeiten des Ladungsverstärker sowie des Kraftsensors erfasst (siehe Abbildung 8.6), denn der Ladungsverstärker hat verschiedene Einstellmöglichkeiten für die Empfindlichkeit (siehe Kapitel 3.1.5). In dem Kanal „Charge Amplifier“ aus der „Werte.tdms“-Datei steht ein Wert zwischen 0 und 5. Dieser entspricht jeweils einer bestimmten Empfindlichkeit und wurde im *LabVIEW*-Programm so gesetzt (siehe Kapitel 7.1.6). Die Empfindlichkeit des Ladungsverstärkers sowie die des Kraftsensors (die hier eine Konstante ist und aus dem Datenblatt (siehe Anhang D) entnommen wurde), werden benötigt, um die Messwerte des Kraftsensors von Spannung in Kraft umzurechnen (siehe Abbildung 8.26 mit Erklärung).

```
charge_amplifier0 = CHD(1, "[2]/[10]")

Select Case charge_amplifier0  'Empfindlichkeit des Ladungsverstärkers
  Case 0  sensitivity0 = 0.1  'Angabe in mV/pC
  Case 1  sensitivity0 = 1    ' ""
  Case 2  sensitivity0 = 10   ' ""
  Case 3  sensitivity0 = 0.1  ' ""
  Case 4  sensitivity0 = 1    ' ""
  Case 5  sensitivity0 = 10   ' ""
  Case Else MsgBox "Fehler beim Charge Amplifier, Channel 0!"
End Select

empfindlichkeit_kraft = 4.08  'Empfindlichkeit des Kraftsensors; Angabe in pC/N
```

Abbildung 8.6.: Empfindlichkeit von Ladungsverstärker und Kraftsensor (Codezeilen 46 – 59), Autorin: Melanie Schulze

Für das Deckblatt der graphischen Darstellung sowie als Teil von Dateinamen wird das aktuelle Datum sowie die Uhrzeit benötigt.
Der Befehl „NOW“ (siehe Abbildung 8.7) liefert das aktuelle Datum in dem dahinter angegebenen Format. Dieser Befehl könnte auch die Uhrzeit liefern, indem man das Datumsformat um „hh:nn:ss“ ergänzt. Würde man das machen, würde die Uhrzeit im Namen der Auswertung allerdings nicht exakt mit der Uhrzeit im Namen der *TDMS*-Datei mit den Messdaten aus *LabVIEW* übereinstimmen. Dieses liegt daran, dass die Zeit in *LabVIEW* bestimmt wird, danach läuft das *LabVIEW*-Programm durch, anschließend werden die Daten an *DIAdem* übermittelt und erst dann würde in *DIAdem* die Uhrzeit ausgelesen werden. Somit lägen zwischen dem Auslesen der Uhrzeiten in *LabVIEW* und *DIAdem* mindestens einige Sekunden. Zur besseren Vergleichbarkeit soll aber gerade die gleiche Uhrzeit verwendet werden. Daher werden in der „Werte.tdms“-Datei auch die Stunden, Minuten und Sekunden der Uhrzeit übergeben, welche hier ausgelesen werden. Da *LabVIEW* allerdings führende Nullen weglässt, müssen diese bei Zeiten, die kleiner sind als zehn, angefügt werden. Schließlich wird die Uhrzeit aus Stunden, Minuten und Sekunden

zusammengesetzt. Dass hierbei ein Semikolon anstatt eines Doppelpunktes benutzt wird, liegt daran, dass unter Windows in Dateinamen keine Doppelpunkte erlaubt sind.
Der Befehl „Str()“ dient dazu, eine Zahl in einen String umzuwandeln. Da Datum und Uhrzeit nur zum Beschriften des Deckblattes und als Name der Datei genutzt werden, werden diese auch nur als String benötigt.

```
aktuelles_Datum = Str((NOW),"#yyyy-mm-dd")  'aktuelles Datum speichern

stunden = CHD(1,"[2]/[6]")
if stunden < 10 then
  stunden = "0" + Str(stunden)
end if
minuten = CHD(1,"[2]/[7]")
if minuten < 10 then
  minuten = "0" + Str(minuten)
end if
sekunden = CHD(1,"[2]/[8]")
if sekunden < 10 then
  sekunden = "0" + Str(sekunden)
end if

'Uhrzeit aus Stunden, Minuten und Sekunden zusammensetzen:
uhrzeit = Str(stunden) + ";" + Str(minuten) + ";" + Str(sekunden)
```

Abbildung 8.7.: Speichern des Datums sowie der Uhrzeit (Codezeilen 61 – 79), Autorin: Melanie Schulze

Als nächstes sollen überflüssige Kanäle gelöscht werden (siehe Abbildung 8.8).
Insgesamt sind vier Messkarten des Typs „PXI–4472B“ mit jeweils acht Eingängen vorhanden (siehe Kapitel 3.1.2). Diese werden generell alle eingelesen, auch wenn nicht immer alle benutzt werden.
Die Gruppe mit den Messwerten ist wie folgt aufgebaut:

- Kanäle 1-30: potentielle Beschleunigungssensormesswerte
- Kanäle 31&32: potentielle Kraftsensormesswerte
- Kanal 33: Signal, mit dem der Shaker angeregt wird

Da momentan nur maximal sechs Beschleunigungssensoren zur Verfügung stehen, bleiben die Kanäle 19-30 ungenutzt. Des Weiteren wird zur Zeit nur ein Kraftsensor verwendet, weshalb der 32. Kanal auch überflüssig ist. Wie viele Kanäle von den ersten 18 benutzt werden, hängt davon ab, wie viele Sensoren bei der aktuellen Messung verwendet werden. Die Kanäle, die nicht gebraucht werden, haben auch keinen Inhalt und werden deswegen hier gelöscht. Die Zahl in den Klammern nach „ChannelGroups“ gibt die Gruppe an, die Zahl in den Klammern nach „Remove“ den Kanal, der innerhalb dieser Gruppe gelöscht werden soll. Dass die Anzahl der Sensoren mit drei multipliziert werden muss, liegt daran, dass pro Sensor drei Kanäle verwendet werden (für die X-, Y- und Z-Richtung).

```
'Überflüssige Channels löschen:
'-------------------------------------------------------------
Call Data.Root.ChannelGroups(1).Channels.Remove(32)

for i = (anz_sensoren*3+1) to 30  'for-Schleife (1)

  Call Data.Root.ChannelGroups(1).Channels.Remove(anz_sensoren*3+1)

next  'Ende der for-Schleife (1)
'-------------------------------------------------------------
```

Abbildung 8.8.: Löschen überflüssiger Kanäle (Codezeilen 81 – 94), Autorin: Melanie Schulze

Am Anfang jeder Messung entsteht ein Fehler, welcher auf das Einschwingen des Tiefpassfilters (siehe Abschnitt 7.2.20) zurückzuführen ist. Da dieser Messfehler nicht verhindert werden kann, sollen die fehlerhaften Messwerte aus den Kanälen entfernt werden (siehe Abbildung 8.9).
Der Befehl „DataAreaDel()“ führt dieses Entfernen durch. Der erste und letzte Parameter stellt jeweils einen Kanal dar. Vom ersten bis zum letzten Kanal werden Werte gelöscht. Um welche Werte innerhalb der Kanäle es sich handelt, geben die anderen beiden Parameter an: Der zweite Parameter ist dabei der erste, der dritte Parameter der letzte zu löschende Wert. In diesem Fall werden vom ersten bis zum letzten Messwertkanal jeweils 50 Werte gelöscht, wobei die 50 ein Erfahrungswert ist.

```
'Falsche Channeleinträge löschen:
'-------------------------------------------------------------
Call DataAreaDel("[1]/[1]",1,50,"[1]/[" + Str(anz_sensoren*3 + 2) + "]")
'-------------------------------------------------------------
```

Abbildung 8.9.: Löschen falscher Kanaleinträge (Codezeilen 96 – 102), Autorin: Melanie Schulze

Im *LabVIEW*-Programm hat der Anwender die Möglichkeit, Orte anzugeben, an denen sich die Sensoren befinden sowie Namen für die Sensoren zu vergeben (siehe Abschnitt 7.2.17). Die Orte werden unter der Kanal-Eigenschaft „Description“ gespeichert, die Namen der Sensoren als Namen der jeweiligen Kanäle. Da die Orte später der Auswertung zugefügt werden sollen, werden sie ausgelesen und in dem Array „ort_sensoren“ gespeichert (siehe Abbildung 8.10). Die Namen werden in dem Array „namen_sensoren“ gespeichert.
Für jeden Sensor muss nur einmal der Ort und der Name ausgelesen werden, da diese beiden Informationen für alle drei Sensorrichtungen gleich sind.

```
'"Description" und "Name" der Sensor-Channels speichern:
'-----------------------------------------------------------------
n = 1  'n: Zähler für den jeweils ersten Kanal eines Sensors (also "X")

for i = 1 to anz_sensoren  'for-Schleife (2)

  'Description auslesen:
  ort_sensoren(i) = Data.Root.ChannelGroups(1).Channels(n).Properties("Description").Value
  'Name auslesen und überflüssige Zeichen löschen:
  name_sensoren(i) = Data.Root.ChannelGroups(1).Channels(n).Properties("Name").Value
  name_sensoren(i) = Left(name_sensoren(i), Len(name_sensoren(i)) - 2)

  n = n + 3

next  'Ende der for-Schleife (2)
'-----------------------------------------------------------------
```

Abbildung 8.10.: Auslesen der Sensor-Orte und -Namen (Codezeilen 104 – 120), Autorin: Melanie Schulze

Nachdem alle Vorabeinstellungen getätigt wurden, beginnt nun der Programmcode mit den Berechnungen.
Es werden alle Berechnungen zur Auswertung zwei Mal vorgenommen: einerseits mit den originalen Messwerten, also den Beschleunigungswerten, andererseits wurden aus den Beschleunigungssignalen Wegsignale berechnet, welche ebenfalls ausgewertet werden. Warum hier so vorgegangen wird, ist in Kapitel 9 nachzulesen.

Da sich die Berechnungen im Grunde nur in der Auswahl der jeweiligen Kanäle unterscheiden, werden im Folgenden alle Berechnungen anhand der Beschleunigungssignale erläutert.

Als erstes wird der Übersicht halber eine neue Kanalgruppe erstellt (siehe Abbildung 8.11), in der alle Berechnungen zu den Beschleunigungssignalen gespeichert werden. Dieses wird später auch für die Wegsignale ausgeführt.
Die „2“ gibt hierbei an, die wievielte Gruppe die erstellte Gruppe werden soll.

```
'BESCHLEUNIGUNG:
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Call Data.Root.ChannelGroups.Add("Beschleunigung [m/s²]", 2)
```

Abbildung 8.11.: Erstellen einer neuen Kanalgruppe für die Beschleunigungssignale (Codezeilen 122 – 125), Autorin: Melanie Schulze

In Abbildung 8.12 ist eine Übersicht aller im Datenportal vorhandenen Kanalgruppen zu sehen.

Abbildung 8.12.: Übersicht über die Kanalgruppen, Autorin: Melanie Schulze

Nun soll die FFT berechnet werden. Diese wird über den gesamten Messwertkanal durchgeführt. In Abbildung 8.13 ist die Berechnung dieser zu sehen.
Der Befehl „ChnLength()“ liefert die Anzahl der Einträge innerhalb eines Kanals. Da alle Kanäle, die Messwerte enthalten, gleich lang sind, muss dieses nur einmal mit dem ersten Kanal berechnet werden. Für die Berechnung einer FFT gibt es verschiedene Einstellmöglichkeiten, wie zum Beispiel, ob die Phase berechnet werden soll oder ob eine Fensterfunktion verwendet werden soll. Der eigentliche Befehl, der die FFT berechnet, ist „ChnFFT1()“, bei welchem der jeweilige Kanal angegeben werden muss.

```
'FFT berechnen:
'////////////////////////////////////////////////////////////////////////////////
channel_laenge = ChnLength("[1]/[1]")

for i = 1 to anz_sensoren*3  'for-Schleife (3)

  FFTIndexChn      = 0
  FFTIntervUser    = "NumberStartOverl"
  FFTIntervPara(1) = 1  'Anzahl der Intervalle innerhalb eines Channels
  FFTIntervPara(2) = channel_laenge  'Größe der Intervalle, hier: kompletter Channel
  FFTIntervPara(3) = 1  'Startpunkt innerhalb des Channels
  FFTIntervOverl   = 0
  FFTNoV           = 0
  FFTWndFct        = "Rectangle"  'keine Fensterfunktion
  FFTWndPara       = 10
  FFTWndCorrectTyp = "No"
  FFTAverageType   = "No"
  FFTAmplFirst     = "Amplitude"
  FFTAmpl          = 1  'festlegen, dass die Amplitude berechnet werden soll
  FFTAmplType      = "Ampl.Peak"
  FFTCalc          = 0
  FFTAmplExt       = "No"
  FFTPhase         = 1  'festlegen, dass die Phase berechnet werden soll
  FFTCepstrum      = 0

  Call ChnFFT1("","[1]/[" + Str(i) + "]")  'FFT für ausgewählten Channel berechnen

next  'Ende der for-Schleife (3)
'////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.13.: Berechnung der FFT (Codezeilen 127 – 156), Autorin: Melanie Schulze

Später sollen bei der FFT nur die Frequenzen dargestellt werden, die auch gesweept wurden. Bei der Berechnung der FFT entstehen aber viel mehr Kanaleinträge als benötigt. Diese sollen gelöscht werden, um später das Darstellen der Graphen zu erleichtern.

Hierbei tritt jedoch ein Problem auf. Es gibt zwei verschiedene Arten von Kanälen: einmal die numerischen Kanäle, die einfach nur Werte enthalten und andererseits Waveform-Kanäle. Diese Kanäle enthalten auch Werte, aber zusätzlich speichern diese implizit zu diesen Werten gehörende Abszissen-Werte, indem der Anfangswert und die Schrittweite gespeichert wird. Das Problem bei diesen Kanälen ist nun, dass man nicht so einfach herausfinden kann, der wievielte Eintrag welchem Abszissen-Wert entspricht, d.h. man kennt zwar die Anzahl der Frequenzen, jedoch nicht den Punkt, ab dem innerhalb des Kanals gelöscht werden muss.
Es besteht allerdings die Möglichkeit, einen Waveform-Kanal in zwei numerische Kanäle umzuwandeln, wobei der eine die Abszissen- und der andere die Ordinaten-Werte des Waveform-Kanals enthält (siehe Abbildung 8.14).

Abbildung 8.14.: Umwandeln des Waveform-Channels in zwei numerische Kanäle, Autorin: Melanie Schulze

Um nun den richtigen Abszissen-Wert herauszufinden, soll nach folgender Formel vorgegangen werden (siehe (8.2)).

$$gew\ddot{u}nschter\ Abszissen\_Wert = \frac{maximaler\ Abszissen\_Wert \cdot gew\ddot{u}nschte\ Frequenz}{maximale\ Frequenz} \tag{8.2}$$

Um diese umzusetzen wird zunächst die Länge des ersten FFT-Kanals bestimmt (siehe Abbildung 8.15). Da alle FFT-Kanäle gleich lang sind, ist es egal, welcher hier verwendet wird. Die Länge entspricht auch der Nummer des letzten Kanaleintrags.
Als nächstes wird mit dem Befehl „WfChnToChn()“ der erste FFT-Kanal in zwei numerische Kanäle umgewandelt, der Parameter gibt dabei den umzuwandelnden Kanal an. Jetzt wird von dem Kanal, der die Abszissen-Werte enthält („Frequency“) der letzte Wert bestimmt. Dazu wird der vorher berechnete Wert „channel_laenge“ als Index verwendet, da so der letzte Wert aus dem Kanal der Abszissen-Werte ausgelesen werden kann. Somit hat man nun den größten Index innerhalb des Waveform-Kanals sowie die Frequenz, die an diesem Punkt vorliegt. Da auch die Frequenz, ab der gelöscht werden soll, bekannt ist, kann nun nach Formel (8.2) der Abszissen-Wert zu der gewünschten Frequenz berechnet werden. Da die Abszissen-Werte nicht mehr und die Ordinaten-Werte wieder als Waveform-Kanal benötigt werden, wird das Umwandeln mit dem Befehl „ChnToWfChn()“ rückgängig gemacht. Dabei werden als Parameter die beiden numerischen Kanäle angegeben. Als dritten Parameter kann man angeben, ob der Kanal, der die Abszissen-Werte enthält, gelöscht werden soll („1“) oder nicht („0“).

Da sich als Ergebnis der Formel (8.2) auch Kommazahlen ergeben können, wird der berechnete Abszissen-Wert auf eine ganze Zahl gerundet. Dafür wird der Befehl „Round()“ verwendet. Als Parameter gibt man einerseits die zu rundende Variable an und andererseits die Nachkommastellen, auf die diese gerundet werden soll.
Schließlich werden in allen FFT-Kanälen alle Einträge von dem berechneten Punkt bis zum Kanalende gelöscht.

```
'Überflüssige Channel-Einträge in den Channels der FFT löschen:
'---------------------------------------------------------------
DIM x_Wert
channel_laenge = ChnLength("Werte/AmplitudePeak")  'Länge des ersten FFT-Channels bestimmen

Call WfChnToChn("Werte/AmplitudePeak")  'Waveform-Channel in zwei numerische Channels umformen

max = CHD(channel_laenge,"Werte/Frequency")

Call ChnToWfChn("Werte/Frequency","Werte/AmplitudePeak",1)  'die beiden numerischen Channels wieder
                                                            'in einen Waveform-Channel umformen

if rauschen = 1 then
  x_Wert = channel_laenge * f_rauschen / max  'gesuchten x-Wert berechnen
else
  x_Wert = channel_laenge * (startfrequenz+anz_freq*schrittweite) / max  'gesuchten x-Wert berechnen
end if
x_Wert = Round(x_Wert,0)  'x-Wert auf eine ganze Zahl runden

Call DataAreaDel("Werte/AmplitudePeak",x_Wert+1,channel_laenge-x_Wert,"Werte/Phase" + Str(anz_sensoren*3-1))
'---------------------------------------------------------------
```

Abbildung 8.15.: Löschen überflüssiger Einträge in den FFT-Kanälen (Codezeilen 158 – 179), Autorin: Melanie Schulze

Immer, wenn neue Kanäle durch Berechnungen entstehen, werden diese in der letzten sich im Datenportal befindenden Gruppe (hier: „Werte“) erstellt.
Die durch die FFT entstandenen Kanäle sollen nun in die richtige Gruppe („Beschleunigung [m\s^2]“) verschoben (siehe Abbildung 8.16) und zur besseren Übersicht umbenannt (siehe Abbildung 8.17) werden.
Es ist darauf zu achten, dass der jeweils erste Amplituden- bzw. Phasenkanal nur „AmplitudePeak“ bzw. „Phase“ heißt. Alle anderen sind durchnummeriert. Daher muss der jeweils erste Kanal alleine, alle anderen können innerhalb der for-Schleife verschoben und umbenannt werden.
Der Befehl zum Verschieben heißt „DataMove()“. Der erste Parameter gibt hierbei den zu verschiebenden Kanal an und der zweite den neuen Ort für den Kanal.

```
'Durch FFT entstandene Channels verschieben und umbenennen:
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
n = 1  'n: Index, an den die FFT-Channel verschoben werden sollen

'Verschieben (von Gruppe "Werte" nach Gruppe "Beschleunigung [m\s²]"):
'-------------------------------------------------------------
'Verschieben des ersten Amplitude/ bzw. Phase-Channels:
Call Data.Move(Data.Root.ChannelGroups(3).Channels("AmplitudePeak"),Data.Root.ChannelGroups(2).Channels,n)
Call Data.Move(Data.Root.ChannelGroups(3).Channels("Phase"),Data.Root.ChannelGroups(2).Channels,n+1)

n = n + 2

for i = 1 to anz_sensoren*3 - 1  'for-Schleife (4)

  'Verschieben der restlichen Amplitude/ bzw. Phase-Channels:
  Call Data.Move(Data.Root.ChannelGroups(3).Channels("AmplitudePeak" + Str(i)),Data.Root.ChannelGroups(2).Channels,n)
  Call Data.Move(Data.Root.ChannelGroups(3).Channels("Phase" + Str(i)),Data.Root.ChannelGroups(2).Channels,n+1)

  n = n + 2

next  'Ende der for-Schleife (4)
'-------------------------------------------------------------
```

Abbildung 8.16.: Verschieben der FFT-Kanäle (Codezeilen 181 – 202), Autorin: Melanie Schulze

Das Umbenennen ist in Abbildung 8.17 zu sehen. Die neuen Kanäle sollen die Namen „FFT Amplitude“ bzw. „FFT Phase“ mit dem zugehörigen Sensornamen und der zugehörigen Richtung bekommen.

```
'Umbenennen:
'-------------------------------------------------------------
l = 2  'l: Index des Arrays "sensorrichtung"
m = 1  'm: Sensornummer
n = 1  'alte Namen: "AmplitudePeak1, AmplitudePeak2...", bzw. "Phase1, Phase2..."
       'n: Index der alten Namen

'Umbenennen des ersten Amplitude/ bzw. Phase-Channels:
Data.Root.ChannelGroups(2).Channels("AmplitudePeak").Name = "FFT Amplitude " + name_sensoren(1) + ", X"
Data.Root.ChannelGroups(2).Channels("Phase").Name = "FFT Phase " + name_sensoren(1) + ", X"

for i = 1 to anz_sensoren*3-1  'for-Schleife (5)

  'Umbenennen der restlichen Amplitude/ bzw. Phase-Channels:
  Data.Root.ChannelGroups(2).Channels("AmplitudePeak" + Str(n)).Name = "FFT Amplitude " + name_sensoren(m) + ", " + sensorrichtung(l)
  Data.Root.ChannelGroups(2).Channels("Phase" + Str(n)).Name = "FFT Phase " + name_sensoren(m) + ", " + sensorrichtung(l)

  n = n + 1
  l = l + 1

  if (l = 4) then  'da es nur drei Sensorrichtungen gibt, muss l nach 3 ("Z") wieder auf 1 ("X") gesetzt werden
    m = m + 1
    l = 1
  end if

next  'Ende der for-Schleife (5)
'-------------------------------------------------------------
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.17.: Umbenennen der FFT-Kanäle (Codezeilen 204 – 231), Autorin: Melanie Schulze

In Abbildung 8.18 ist ein beispielhaftes Ergebnis dieses Umbenennens zu sehen.

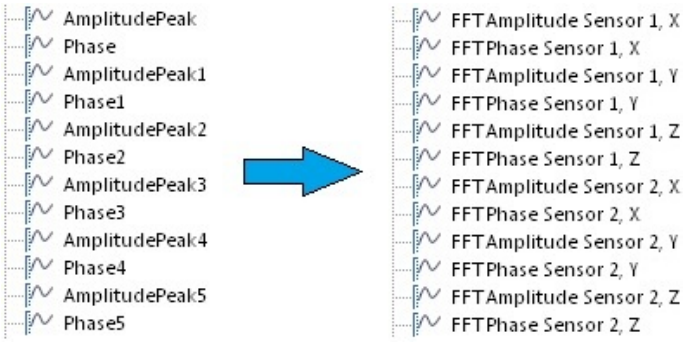


Abbildung 8.18.: FFT-Kanalnamen vorher (links) und Kanalnamen nach dem Umbenennen (rechts), Autorin: Melanie Schulze

Als nächstes sollen in den Amplituden der FFT jeweils drei Maxima gefunden werden (siehe Abbildung 8.19). Diese sollen auf den Seiten der Auswertung mit angegeben werden.
Der Befehl „ChnPeakFind()“ dient dazu, Extremwerte zu finden. Der erste Parameter gibt den Kanal an, in dem sich die x-Werte befinden. Diese Angabe ist in diesem Fall nicht nötig, da es sich bei den zu untersuchenden FFT-Kanälen um Waveform-Kanäle handelt. Der zweite Parameter gibt den Kanal an, der die y-Werte enthält (in diesem Fall die Kanäle, die die Amplituden-Kanäle der FFT enthalten). Die nächsten beiden Parameter geben die Kanäle an, in denen die Ergebnisse gespeichert werden (hier „PeakX“ + Nummer für die x-Werte, die zu den Maxima gehören und „PeakY“ + Nummer für die Maxima-Werte). Der fünfte Parameter gibt die Anzahl an Extrema an, die gesucht werden sollen (hier 3). Der Parameter „Max.Peaks“ sagt aus, dass Maxima und nicht Minima gesucht werden sollen. Bei dem letzten Parameter kann angegeben werden, ob die gefundenen Werte danach sortiert werden sollen, welcher zuerst aufgetreten ist oder nach der Größe (die Angabe „Amplitude“ entspricht hier der Sortierung nach der Größe).
Dass die Variable „j“ immer um zwei erhöht werden muss, liegt daran, dass sich zwischen den Amplituden-Kanälen die Phasen-Kanäle befinden, in denen nicht nach Extremwerten gesucht werden soll.

```
'Jeweils drei Maxima in den Werten der FFT finden:
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
j = 1  'j: Index, an dem die Amplituden-Channels der FFT beginnen

for i = 1 to anz_sensoren*3  'for-Schleife (6)

  Call ChnPeakFind("","[2]/[" + Str(j) + "]","[2]/PeakX" + Str(i),"[2]/PeakY" + Str(i),3,"Max.Peaks","Amplitude")
  j = j + 2  'j immer zwei hochsetzen, da zwischen den Amplituden-Channels die Phasen-Channels liegen

next  'Ende der for-Schleife (6)
```

Abbildung 8.19.: Finden von jeweils drei Maxima in den Amplituden der FFT (Codezeilen 233 – 242), Autorin: Melanie Schulze

Zur besseren Übersicht sollen die Maxima-Kanäle umbenannt werden. Hierzu werden zunächst einige Variablen benötigt (siehe Abbildung 8.20).

```
'Entstandene Maxima-Channels umbenennen:
'-------------------------------------------------------------
k = 0 'pro Frequenz und Sensor drei FFTs (X,Y,Z), in denen nach Maxima gesucht wurde
      'k: Zähler um zu wissen, nach wievielen Channels die Sensornummer m einen
      'hochgezählt werden muss (siehe (**) )
l = 1 'l: Index des Arrays "sensorrichtung"
m = 1 'm: Sensornummer
n = 1 'alte Namen: "PeakX1 / PeakY1, PeakX2 / PeakY2..."
      'n: Index der alten Namen
```

Abbildung 8.20.: Initialisieren der benötigten Variablen zum Umbenennen der Maxima-Kanäle (Codezeilen 244 – 251), Autorin: Melanie Schulze

Das Umbenennen der Kanäle ist in Abbildung 8.21 zu sehen.

```
for i = 1 to anz_sensoren*3  'for-Schleife (7)

  Data.Root.ChannelGroups(2).Channels("PeakX" + Str(n)).Name = "X A
  Data.Root.ChannelGroups(2).Channels("PeakY" + Str(n)).Name = "Y A

  n = n + 1

  l = l + 1
  k = k + 1

  if (k = 3) then  '(**)
    k = 0
    m = m + 1
  end if

  if (l = 4) then  'Da es nur drei Sensorrichtungen gibt, muss
                   'l nach 3 ("Z") wieder auf 1 ("X") gesetzt werden.
    l = 1
  end if

next  'Ende der for-Schleife (7)
'-------------------------------------------------------------
'/////////////////////////////////////////////////////////////////////////
```

Abbildung 8.21.: Umbenennen der Maxima-Kanäle (Codezeilen 253 – 274), Autorin: Melanie Schulze

Um die Übersicht zu verbessern, wird in Abbildung 8.21 die Abkürzung „**A**“ verwendet. Die Bedeutung dieser ist in Abbildung 8.22 zu sehen.

```
-Werte Maxima, FFT, " + name_sensoren(m) + ", " + sensorrichtung(l)
```

Abbildung 8.22.: Bedeutung der Abkürzung „**A**“, Autorin: Melanie Schulze

Aus den gefundenen Maxima soll nun jeweils das Maximum pro Sensor ermittelt werden. Dieses soll später zusammen mit der zugehörigen Frequenz und Sensorrichtung auf einer gesonderten Zusammenfassungs-Seite erscheinen.
Hierzu werden Variablen benötigt, deren Deklaration und Initialisierung in Abbildung 8.23 zu sehen sind.

```
'Jeweiliges Maximum pro Sensor finden (für das Zusammenfassungsblatt):
'////////////////////////////////////////////////////////////////////////////////////////////////
DIM gruppe, neuer_channel, max, max1, max_freq, max_rtg
Set gruppe = Data.Root.ChannelGroups("Beschleunigung [m\s²]")  'Gruppe festlegen, in der der
                                                               'neue Channel erstellt werden soll
max = 0  'Vergleichswert
m = 1 + anz_sensoren*6 'm: Index, ab dem die Channels mit den Maxima-Werten beginnen
```

Abbildung 8.23.: Deklaration und Initialisierung der benötigten Variablen (Codezeilen 276 – 281), Autorin: Melanie Schulze

Beim Finden des Maximums pro Sensor wird folgendermaßen vorgegangen. Zunächst werden alle Kanäle, die die Y-Werte der Maxima eines Sensors enthalten, verglichen, um so den größten unter diesen zu finden. Anschließend wird zu dem Maximalwert noch der zugehörige Frequenz-Wert und die zugehörige Sensorrichtung ermittelt.
Dieses Vorgehen ist in Abbildung 8.24 zu sehen.
Der Befehl „CCh()“ berechnet charakteristische Werte des Kanals, der als erster Parameter angegeben ist. Der zweite Parameter (hier „2“) gibt an, welcher charakteristische Wert berechnet werden soll. Zur Auswahl stehen hier unter anderem Mittelwert („0“), Minimum („1“) oder Maximum („2“). Da es nur darum geht, einen Maximalwert zu finden und nicht wie in Abbildung 8.19 ein bzw. mehrere Wertepaar/e, muss nicht der Befehl „ChnPeakFind()“ benutzt werden.
Um die zum Maximum gehörende Frequenz zu finden, wird verglichen, welcher der drei in dem Maximum-Kanal befindlichen Werte das Maximum ist. Somit wird entweder die Zahl 1, 2 oder 3 gespeichert. Mit dieser Zahl wird dann aus dem Kanal der x-Werte die richtige Frequenz ausgelesen.

```
for j = 1 to anz_sensoren  'for-Schleife (8)

  l = 1  'Zähler für die Sensorrichtung
  max = 0  'Wenn alle Frequenzen eines Sensors durch sind, den
           'Vergleichswert für den nächsten Sensor wieder auf 0 setzen

  for i = 1 to 3  'for-Schleife (9)

    if (ChnLength("[2]/["+ Str(m+1) + "]") > 0) then  'Wenn Maxima vorhanden sind:

      max1 = CCh("[2]/["+ Str(m+1) + "]",2)  'Maximum suchen
      max1 = Round(max1,6)  'Maximmum auf 6 Nachkommastellen runden, weil es sonst
                            'beim folgenden Vergleichen immer Abweichungen gibt

      'Vergleichen, der wievielte Wert des Channels das Maximum war, damit der richtige
      'x-Wert dazu gefunden werden kann
      '(auch hier: Runden auf 6 Nachkommastellen):
      if (max1 = Round(CHD(1,"[2]/["+ Str(m+1) + "]"),6)) then
        p = 1
      elseif (max1 = Round(CHD(2,"[2]/["+ Str(m+1) + "]"),6)) then
        p = 2
      elseif (max1 = Round(CHD(3,"[2]/["+ Str(m+1) + "]"),6)) then
        p = 3
      end if

    end if

      if max1 > max then  'Wenn das gefundene Maximum größer ist als das bisherige:

        max = max1        'aktuelles Maximum überschreiben
        max_freq = CHD(p,"[2]/["+ Str(m) + "]")  'Frequenz dieses Maximums speichern
        max_rtg = l       'Sensorrichtung beim Maximum speichern

      end if

    m = m + 2  'm zwei hochzählen, weil immer nur die Y-Werte der Maxima gebraucht werden
               'und die X-Werte in der Channel-Reihenfolge jeweils dazwischen liegen
    l = l + 1

    end if
  next  'Ende der for-Schleife (9)
```

Abbildung 8.24.: Finden des Maximums pro Sensor (Codezeilen 283 – 319), Autorin: Melanie Schulze

Um die ermittelten Werte für Frequenz, Sensorrichtung und Maximum zu speichern, wird ein neuer Kanal erstellt, in den diese eingetragen werden. Dieses ist in Abbildung 8.25 zu sehen.
Der Befehl „Add()“ fügt der in Abbildung 8.23 definierten Gruppe einen neuen Kanal hinzu. Dieser Kanal ist über die Variable „neuer_channel“ ansprechbar und hat den Namen, den man als ersten Parameter angibt (hier „Maximum“ + Sensorname). Beim zweiten Parameter hat man die Möglichkeit, festzulegen, welchen Datentyp der neue Kanal haben soll. Die Auswahl besteht hier zwischen Gleitkommazahl, Datum oder Text (hier Gleitkommazahl, „DataTypeFloat64“). Auf die einzelnen Zeilen des neuen Kanals kann in runden Klammern zugegriffen werden. Beispielsweise entspricht der erste Wert des Kanals dem Ausdruck „neuer_channel(1)“.

```
'neuen Channel erstellen:
Set neuer_channel = gruppe.Channels.Add("Maximum " + name_sensoren(j),DataTypeFloat64)
neuer_channel(1)  = max       'gefundenes Maximum in die erste Zeile
                              'des neuen Channels schreiben
neuer_channel(2)  = max_freq  'anregende Frequenz bei dem gefundenen Maximum
                              'in die zweite Zeile des neuen Channels schreiben
neuer_channel(3)  = max_rtg   'Sensorrichtung bei dem gefundenen Maximum in
                              'die dritte Zeile des neuen Channels schreiben

next  'Ende der for-Schleife (8)
'////////////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.25.: Speichern des Maximums pro Sensor (Codezeilen 321 – 327), Autorin: Melanie Schulze

Als nächstes werden die Messwerte des Kraftsensors bearbeitet, denn diese Messwerte sind in Volt, sollen aber in Newton umgerechnet werden.
Hierfür müssen die Empfindlichkeiten des Kraftsensors sowie des Verstärkers beachtet werden (siehe (8.3)).

$$Wert\,[N] = \frac{Messwert\,[V] \cdot 1000}{Empfindlichkeit\ Verstärker\,[\frac{mV}{pC}] \cdot Empfindlichkeit\ Sensor\,[\frac{pC}{N}]} \tag{8.3}$$

Die Umrechnung der Kraftsensor-Messwerte ist in Abbildung 8.26 zu sehen.
Als erstes wird nach (8.3) der Umrechnungsfaktor berechnet. Anschließend müssen die Messwerte mit diesem Faktor multipliziert werden. Mit dem Befehl „ChnLinScale()“ ist es möglich, einen Kanal neu zu skalieren. Der erste Parameter ist dabei der zu skalierende Kanal und der zweite der Name für den neu skalierten Kanal, d.h. nach dem Skalieren ist der neu skalierte sowie der alte Kanal vorhanden. Der dritte Parameter steht für den Skalierungsfaktor (hier „faktor“) und mit dem vierten Parameter kann man einen Offset hinzufügen.
Da der alte Kanal nach dem Skalieren noch vorhanden ist, aber nicht mehr gebraucht wird, wird dieser abschließend gelöscht.

```
'Bearbeitung der Messwerte des Kraftsensors:
'////////////////////////////////////////////////////////////////////////////////////////////////////////////
'Messwerte des Kraftsensors von V auf N umrechnen:
'-------------------------------------------------------------
n = 1 + anz_sensoren*3  'Index, an dem der Channel mit den Werten des Kraftsensors liegt

DIM faktor
faktor = 1000 / sensitivity0 / empfindlichkeit_kraft  'Umrechnungsfaktor

Call ChnLinScale("[1]/[" + Str(n) + "]","Messwerte Kraftsensor [N]",faktor,0)  'Messwerte mit dem Umrech-
                                                                               'nungsfaktor multiplizieren

Call Data.Root.ChannelGroups(1).Channels.Remove(n)  'Kraftsensor-Messwerte [V] löschen
'-------------------------------------------------------------
```

Abbildung 8.26.: Umrechnen der Kraftsensor-Messwerte (Codezeilen 329 – 344), Autorin: Melanie Schulze

Mit den Kraftsensor-Messwerten soll für die Auswertung ebenfalls eine FFT berechnet werden. Die Berechnung dieser sowie das Verschieben und Umbenennen der dabei entstandenen Kanäle ist in Abbildung 8.27 zu sehen.

```
'FFT der Kraftsensor-Werte berechnen, Channel verschieben und umbenennen:
'---------------------------------------------------------------
channel_laenge = ChnLength("[1]/Messwerte Kraftsensor [N]")

FFTIndexChn      = 0
FFTIntervUser    = "NumberStartOverl"
FFTIntervPara(1) = 1  'Anzahl der Intervalle innerhalb eines Channels
FFTIntervPara(2) = channel_laenge  'Größe der Intervalle, hier: kompletter Channel
FFTIntervPara(3) = 1  'Startpunkt innerhalb des Channels
FFTIntervOverl   = 0
FFTNoV           = 0
FFTWndFct        = "Rectangle"  'keine Fensterfunktion
FFTWndPara       = 10
FFTWndCorrectTyp = "No"
FFTAverageType   = "No"
FFTAmplFirst     = "Amplitude"
FFTAmpl          = 1  'festlegen, dass die Amplitude berechnet werden soll
FFTAmplType      = "Ampl.Peak"
FFTCalc          = 0
FFTAmplExt       = "No"
FFTPhase         = 1  'festlegen, dass die Phase berechnet werden soll
FFTCepstrum      = 0

Call ChnFFT1("","[1]/Messwerte Kraftsensor [N]")  'FFT für ausgewählten Channel berechnen

'Verschieben:
Call Data.Move(Data.Root.ChannelGroups(3).Channels("AmplitudePeak"),Data.Root.ChannelGroups(1).Channels,anz_sensoren*3+4)
Call Data.Move(Data.Root.ChannelGroups(3).Channels("Phase"),Data.Root.ChannelGroups(1).Channels,anz_sensoren*3+5)
'Umbenennen:
Data.Root.ChannelGroups(1).Channels("AmplitudePeak").Name = "FFT Amplitude Kraftsensor"
Data.Root.ChannelGroups(1).Channels("Phase").Name = "FFT Phase Kraftsensor"
'---------------------------------------------------------------
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.27.: FFT der Kraftsensor-Messwerte (Codezeilen 346 – 378), Autorin: Melanie Schulze

Nun sollen FRFs mit den Werten des Kraftsensors und den Werten der Beschleunigungssensoren berechnet werden (siehe Abbildung 8.28). Hierbei entspricht der Kraftsensor dem Eingang und der jeweilige Beschleunigungssensor dem Ausgang des Systems (vgl. hierzu (8.1)).
Als erstes wird die Länge des ersten Beschleunigungssensor-Kanals bestimmt. Da alle Kanäle gleich lang sind, braucht diese Länge nur einmal für alle bestimmt zu werden.
Die Einstellungen der FRF ähneln sehr denen der FFT (vgl. Abbildung 8.13 und Abbildung 8.27). Es wird hier nur zusätzlich angegeben, welche Art von Übertragungsfunktion berechnet werden soll (hier: FFT(A)/FFT(B), nach (8.1)). Außerdem wird der Befehl „ChnFFT2()“ anstatt dem Befehl „ChnFFT1()“ verwendet, da nun mit zwei Kanälen gerechnet wird - in diesem Fall mit dem Kanal der Kraftsensormesswerte und dem jeweiligen Kanal der Beschleunigungssensormesswerte.

```
'FRF berechnen:
'////////////////////////////////////////////////////////////////////////////////////
channel_laenge = ChnLength("[1]/[1]")

for i = 1 to anz_sensoren*3  'for-Schleife (10)

  FFTIndexChn      = 0
  FFTIntervUser    = "NumberStartOverl"
  FFTIntervPara(1) = 1  'Anzahl der Intervalle innerhalb des Channels
  FFTIntervPara(2) = channel_laenge  'Größe des Intervalls, hier: kompletter Channel
  FFTIntervPara(3) = 1  'Startpunkt innerhalb des Channels
  FFTIntervOverl   = 0
  FFTNoV           = 0
  FFTWndFct        = "Rectangle"  'keine Fensterfunktion
  FFTWndPara       = 10
  FFTWndCorrectTyp = "No"
  FFTAverageType   = "No"
  FFTAmplFirst     = "Amplitude"
  FFTAmpl          = 1  'festlegen, dass die Amplitude berechnet werden soll
  FFTAmplType      = "Ampl.Peak"
  FFTCrossSpectr   = 0
  FFTCoherence     = 0
  FFTTransFctType  = "Spectrum H0"  'Art der Übertragungsfunktion auswählen
  FFTCrossPhase    = 0
  FFTTransPhase    = 1  'festlegen, dass die Phase berechnet werden soll

  'FRF mit ausgewählten Channels berechnen (hier: Kraftmesser und Sensor):
  Call ChnFFT2("","Messwerte Kraftsensor [N]","[1]/[" + Str(i) +"]")

next  'Ende der for-Schleife (10)
'////////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.28.: Berechnen der FRF (Codezeilen 380 – 411), Autorin: Melanie Schulze

Als nächstes werden überflüssige Kanaleinträge gelöscht. Da dieses aber der Berechnung aus Abbildung 8.15 nahezu identisch ist, wird dieses hier nicht erläutert.

Nun werden die FRF-Kanäle in die richtige Gruppe verschoben (Abbildung 8.29) und zur besseren Übersicht umbenannt (Abbildung 8.30).
Auch hier muss wieder der jeweils erste Kanal der Amplituden- und Phasen-Kanäle einzeln verschoben und umbenannt werden, da diese keine Nummer haben, alle anderen aber schon (siehe Abbildung 8.29, vgl. hierzu auch die Abbildungen 8.16 und 8.17 sowie die Erklärung dazu).

```
'Durch FRF entstandene Channels verschieben und umbenennen:
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

 j = 1 + anz_sensoren*13   'j: Index, an den die Channels geschoben werden sollen

 'Verschieben:
 '---------------------------------------------------------------
 Call Data.Move(Data.Root.ChannelGroups(3).Channels("TransferFunctionAmplitude"),Data.Root.ChannelGroups(2).Channels,j)
 Call Data.Move(Data.Root.ChannelGroups(3).Channels("TransferFunctionPhase"),Data.Root.ChannelGroups(2).Channels,j+1)

 j = j + 2

 for i = 1 to anz_sensoren*3-1  'for-Schleife (11)

   Call Data.Move(Data.Root.ChannelGroups(3).Channels("TransferFunctionAmplitude" + Str(i)),Data.Root.ChannelGroups(2).Channels,j
   Call Data.Move(Data.Root.ChannelGroups(3).Channels("TransferFunctionPhase" + Str(i)),Data.Root.ChannelGroups(2).Channels,j+1)

   j = j + 2

 next  'Ende der for-Schleife (11)
 '---------------------------------------------------------------
```

Abbildung 8.29.: Verschieben der FRF-Kanäle (Codezeilen 436 – 455), Autorin: Melanie Schulze

```
'Umbenennen:
'---------------------------------------------------------------
k = 2 'pro Sensor 3 FRFs (X,Y,Z)
      'k: Zähler um zu wissen, nach wievielen Channels die
      'Sensornummer m einen hochgezählt werden muss (siehe (***) )
m = 1 'm: Sensornummer
l = 2 'l: Index des Arrays "sensorrichtung"

Data.Root.ChannelGroups(2).Channels(B).Name = "FRF Amplitude, " + name_sensoren(1) + ", X"
Data.Root.ChannelGroups(2).Channels(C).Name = "FRF Phase, " + name_sensoren(1) + ", X"

for i = 1 to anz_sensoren*3-1  'for-Schleife (12)

  Data.Root.ChannelGroups(2).Channels(B + Str(i)).Name = "FRF Amplitude, " + name_sensoren(m) + ", " + Str(sensorrichtung(l))
  Data.Root.ChannelGroups(2).Channels(C + Str(i)).Name = "FRF Phase, " + name_sensoren(m) + ", " + Str(sensorrichtung(l))

  l = l + 1
  k = k + 1

  if (k = 4) then  '(***)
    k = 1
    m = m + 1
  end if

  if (l = 4) then
    l = 1
  end if

next  'Ende der for-Schleife (12)
'---------------------------------------------------------------
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.30.: Umbenennen der FRF-Kanäle (Codezeilen 457 – 486), Autorin: Melanie Schulze

Die zur Übersicht verwendeten Abkürzungen „**B**“ und „**C**“ sind in Abbildung 8.31 erklärt.

```
"TransferFunctionAmplitude"
"TransferFunctionPhase"
```

Abbildung 8.31.: Bedeutung der Abkürzungen „**B**“ (oben) und „**C**“ (unten), Autorin: Melanie Schulze

Als letztes sollen auch in den Werten der FRF für die spätere Auswertung Maxima gefunden werden (siehe Abbildung 8.32).
Nachdem mit dem Befehl „ChnPeakFind()“ (vgl. auch Abbildung 8.19 und die Erklärung dazu) die Maxima gefunden wurden, werden die Maxima-Kanäle umbenannt.

```
'Maxima der FRF:
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 j = 1 + anz_sensoren*13  'j: Index, ab dem die FRF-Werte beginnen

for i = 1 to anz_sensoren*3  'for-Schleife (13)

  Call ChnPeakFind("","[2]/[" + Str(j) + "]","[2]/PeakX" + Str(i),"[2]/PeakY" + Str(i),5,"Max.Peaks","Amplitude")  'Maxima suchen
  j = j + 2  'plus 2, da nur in den Amplitude-Channels gesucht werden soll und die Phase-Channels jeweils dazwischen liegen

next  'Ende der for-Schleife (13)

'Umbenennen:
k = 1 'pro Sensor 3 FRFs (X,Y,Z)
      'k: Zähler um zu wissen, nach wievielen Channels die Sensornummer m einen hochgezählt werden muss (siehe (****) )
m = 1 'm: Sensornummer
l = 1 'l: Index des Arrays "sensorrichtung"

for i = 1 to anz_sensoren*3  'for-Schleife (14)

  Data.Root.ChannelGroups(2).Channels("PeakX" + Str(i)).Name = "X-Werte Maxima, FRF, " +  name_sensoren(m) + ", " + sensorrichtung(l]
  Data.Root.ChannelGroups(2).Channels("PeakY" + Str(i)).Name = "Y-Werte Maxima, FRF, " +  name_sensoren(m) + ", " + sensorrichtung(l]

  l = l + 1
  k = k + 1

  if (k = 4) then  '(****)
    k = 1
    m = m + 1
  end if

  if (l = 4) then
    l = 1
  end if

next  'Ende der for-Schleife (14)
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.32.: Finden von Maxima der FRF-Kanäle und Umbenennen der Maxima-Kanäle (Codezeilen 488 – 523), Autorin: Melanie Schulze

Im Programmcode folgt nun das Finden des Maximums pro Sensor. Dieses wird benötigt, damit später die y-Achse der Amplituden der FRF danach skaliert werden kann. Das Vorgehen beim Finden dieses Maximums entspricht nahezu dem aus den Abbildungen 8.23 und 8.24 und wird daher hier nicht erläutert.

Nun folgen die Berechnungen zu den Wegsignalen. Wie oben erwähnt, ähneln diese sehr den Berechnungen mit den Beschleunigungssignalen. Deshalb sollen im Folgenden nur Programmteile erläutert werden, die sich von den Berechnungen mit den Beschleunigungssignalen unterscheiden.

Zunächst wird eine neue Gruppe mit dem Namen „Weg [mm]“ erstellt (siehe Abbildung 8.33).

```
'WEG:
'||||||||||||||||||||||||||||||||||||||||||||||

Call Data.Root.ChannelGroups.Add("Weg [mm]", 3)
```

Abbildung 8.33.: Erstellen einer neuen Kanalgruppe für die Wegsignale (Codezeilen 575 – 578), Autorin: Melanie Schulze

Als erstes müssen die Messwerte der Beschleunigungssensoren in Wegsignale umgerechnet werden. Um von der Beschleunigung zum Weg zu gelangen, wird zwei Mal integriert und jeweils der Mittelwert des Signals abgezogen. Weitere Informationen zu diesem Thema bietet das Kapitel „Umwandeln des Beschleunigungssignals in ein Wegsignal“ (Kapitel 9) .

Zum Integrieren und Umbenennen der durch das Integrieren entstandenen Kanäle werden Variablen benötigt, deren Initialisierung in Abbildung 8.34 in zu sehen ist.

```
'Messwerte von Beschleunigung in Weg umwandeln sowie entstandene Channels umbenennen:
'////////////////////////////////////////////////////////////////////////////////////
DIM summe, mittelwert
l = 1  'l: Index des Arrays "sensorrichtung"
m = 1  'm: Zähler für die Sensoren
o = 1  'o: Zähler um zu wissen, wann m hochgesetzt werden muss
```

Abbildung 8.34.: Initialisierung benötigter Variablen (Codezeilen 580 – 588), Autorin: Melanie Schulze

Zunächst muss der Mittelwert des Signals berechnet werden, um diesen abziehen zu können. Dafür wird nach (8.4) vorgegangen.

$$Mittelwert = \frac{\Sigma\ Kanaleinträge}{Anzahl\ Kanaleinträge} \tag{8.4}$$

Anfangs wird die Länge des zu integrierenden Kanals bestimmt (siehe Abbildung 8.35), welche der Anzahl der Kanaleinträge entspricht. Der Parameter „4“ beim Befehl „CCh()“

gibt an, dass die Summe über den angegebenen Kanal berechnet werden soll. Mit der Kanallänge und der Summe wird nach (8.4) der Mittelwert des Kanals berechnet. Nun wird der Kanal neu skaliert, indem als Offset der negative Mittelwert eingetragen wird. Anschließend befindet sich das mittelwertfreie Signal in dem Kanal „LinearScaled“. Dieser Kanal wird nun integriert. Hierfür wird der Befehl „ChnIntegrate()“ verwendet. Der erste Parameter gibt den Kanal an, der die x-Werte enthält (wird nicht benötigt, da es sich um einen Waveform-Kanal handelt). Der zweite Parameter ist der Kanal, der die y-Werte enthält und der letzte Parameter gibt den Namen des Kanals an, in dem sich anschließend das Integral befindet (hier: „Integrated“). Da sich das Integral in dem Kanal „Integrated“ befindet, wird der Kanal „LinearScaled“ nicht mehr benötigt und daher gelöscht.

```
for i = 1 to anz_sensoren*3  'for-Schleife (17)

  channel_laenge = ChnLength("[1]/[" + Str(i) + "]")  'Länge des jeweiligen Beschleu
                                                      'nigungs-Channels bestimmen
  summe = CCh("[1]/[" + Str(i) + "]",4)  'Summe aus allen Werten
                                         'innerhalb dieses Channels bilden
  mittelwert = summe / channel_laenge  'Mittelwert des Channels berechnen
  'Channel neu skalieren, hier: Mittelwert abziehen :
  Call ChnLinScale("[1]/[" + Str(i) + "]","LinearScaled",1,-mittelwert)
  Call ChnIntegrate("","LinearScaled","Integrated")  'neu skalierten Channel integrieren
  Call Data.Root.ChannelGroups("Werte").Channels.Remove("LinearScaled")  'neu skalierten
                                                                         'Channel löschen
```

Abbildung 8.35.: Mittelwertbildung und 1. Integrieren (Codezeilen 590 – 597), Autorin: Melanie Schulze

Beim zweiten Integrieren ist das Vorgehen dem beim ersten Integrieren sehr ähnlich (siehe Abbildung 8.36). Der einzige Unterschied ist, dass zwischendurch der Kanal gelöscht wird, der das erste Integral enthält.

```
  summe = CCh("Integrated",4)  'Summe aus allen Werten innerhalb des Channels bilden,
                               'der das erste Integral enthält
  mittelwert = summe / channel_laenge  'Mittlwert des Channels berechnen
  'Channel neu skalieren, hier: Mittelwert abziehen:
  Call ChnLinScale("Integrated","LinearScaled",1,-mittelwert)
  Call Data.Root.ChannelGroups(4).Channels.Remove("Integrated")  'Channel, der das erste
                                                                 'Integral enthält, löschen
  Call ChnIntegrate("","LinearScaled","Integrated")  'neu skalierten Channel integrieren
  Call Data.Root.ChannelGroups(4).Channels.Remove("LinearScaled")  'neu skalierten
                                                                   'Channel löschen
```

Abbildung 8.36.: Mittelwertbildung und 2. Integrieren (Codezeilen 599 – 604), Autorin: Melanie Schulze

Schließlich wird beim zweiten Integral auch noch einmal der Mittelwert abgezogen (siehe Abbildung 8.37).

```
summe = CCh("Integrated",4)  'Summe aus allen Werten innerhalb des Channels bilden,
                             'der das zweite Integral enthält
mittelwert = summe / channel_laenge  'Mittelwert des Channels berechnen
Call ChnLinScale("/Integrated","LinearScaled",1,-mittelwert)  'Channel neu skalieren,
                                                              'hier: Mittelwert abziehen
Call Data.Root.ChannelGroups(4).Channels.Remove("Integrated")  'Channel, der das zweite
                                                               'Integral mit Mittelwert
                                                               'enthält, löschen
```

Abbildung 8.37.: Bildung des mittelwertfreien 2. Integrals (Codezeilen 606 – 609), Autorin: Melanie Schulze

Nach dem Integrieren soll der Weg von Meter in Millimeter umgerechnet werden. Dazu wird der Kanal mit dem Faktor 1000 neu skaliert (siehe Abbildung 8.38).

```
'Weg von [m] in [mm] umrechnen:
Call ChnLinScale("Werte/LinearScaled","Weg [mm]/fertig" + Str(i),1000,0)
'neu skalierten Channel löschen:
Call Data.Root.ChannelGroups("Werte").Channels.Remove("LinearScaled")
'Channel, der das 2. mittelwertfreie Integral enthält umbenennen:
Data.Root.ChannelGroups("Weg [mm]").Channels("fertig" + Str(i)).Name = D
```

Abbildung 8.38.: Umrechnen des Weges von Meter in Millimeter (Codezeilen 611 – 614), Autorin: Melanie Schulze

Die Bedeutung der verwendeten Abkürzung „**D**“ (siehe Abbildung 8.38) ist in Abbildung 8.39 zu sehen.

```
name_sensoren(m) + ", " + sensorrichtung(l)
```

Abbildung 8.39.: Bedeutung der Abkürzung „**D**“, Autorin: Melanie Schulze

Schließlich müssen Variablen angepasst bzw. Zähler weitergesetzt werden (siehe Abbildung 8.40).

```
    o = o + 1
    l = l + 1

    if (l = 4) then  'Es gibt nur 3 Sensorrichtungen. Daher muss
                     'l nach 3 ("Z") wieder auf 1 ("X") gesetzt werden.
      l = 1
      m = m + 1
    end if

  next  'Ende der for-Schleife (17)
  '////////////////////////////////////////////////////////////////////////
```

Abbildung 8.40.: Anpassen der Variablen (Codezeilen 616 – 625), Autorin: Melanie Schulze

Anschließend wird mit den Wegsignalen die FFT berechnet und die Maxima dort gesucht sowie die FRF berechnet. Dieses ist abgesehen von Indizes beim Ansprechen von Gruppen und Kanälen mit den Berechnungen der Beschleunigungssignale gleich (vgl. Abbildung 8.13 bis Abbildung 8.25 und Abbildung 8.28).

Hier sollen allerdings die Kanäle, die die Ergebnisse der FRF enthalten von [Millimeter/Newton] auf [Meter/Newton] umgerechnet werden (siehe Abbildung 8.41), damit die Zahlen nicht so groß sind und es damit bei der Auswertung anschaulicher wird. Auch hier muss wieder jeweils der erste Kanal auf Grund fehlender Nummerierung einzeln umgerechnet und gelöscht werden.

```
'FRF-Channels von [mm/N] auf [m/N] umrechnen und alte Channels löschen:
'-------------------------------------------------------------
'Ersten Amplitude-Channel umrechnen:
Call ChnLinScale("Werte/TransferFunctionAmplitude","TFA",1/1000,0)
'Ersten der alten Channels löschen:
Call Data.Root.ChannelGroups("Werte").Channels.Remove("TransferFunctionAmplitude")

for i = 1 to anz_sensoren*3-1  'for-Schleife (26)
  'Restliche Amplitude-Channels umrechnen:
  Call ChnLinScale("Werte/TransferFunctionAmplitude" + Str(i),"TFA" + Str(i),1/1000,0)
  'Restliche der alten Channels löschen:
  Call Data.Root.ChannelGroups("Werte").Channels.Remove("TransferFunctionAmplitude" + Str(i))

next  'Ende der for-Schleife (26)
'-------------------------------------------------------------
```

Abbildung 8.41.: Umrechnen der FRF-Kanäle und löschen der alten Kanäle (Codezeilen 883 – 894), Autorin: Melanie Schulze

Anschließend werden die FRF-Kanäle verschoben, umbenannt und die Maxima der FRF gesucht. Dieses stimmt mit den Berechnungen der Beschleunigungssignale überein (vgl. Abbildung 8.29 – Abbildung 8.32).

Es soll die Möglichkeit bestehen, später die berechneten Werte für eine Messung immer wieder aufzurufen. Aus diesem Grund wird die bis hierher entstandene *TDMS*-Datei mit allen Berechnungen gespeichert (siehe Abbildung 8.42).

```
'Speichern der TDMS-Datei mit allen Berechnungen:
'-------------------------------------------------------------
DIM name_datei
name_datei = "\Berechnungen\" + Str(name_messung) + " " + Str(aktuelles_Datum) + " " + Str(uhrzeit) + " Berechnungen"

Call DataFileSave(pfad&Str(name_datei), "TDM")
'-------------------------------------------------------------
'|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

Abbildung 8.42.: Speichern der *TDMS*-Datei (Codezeilen 1034 – 1041), Autorin: Melanie Schulze

Diese *TDMS*-Dateien liegen in dem Ordner „Berechnungen“ und haben den gleichen Namen wie die zugehörige Messdatendatei mit dem Zusatz „Berechnungen“.

8.2. Graphische Darstellung

Nachdem alle Berechnungen gemacht wurden, sollen die Ergebnisse übersichtlich dargestellt und gespeichert werden. Dafür soll mittels des SCRIPT-Moduls (siehe Abbildung 3.45) auf das REPORT-Modul (siehe Abbildung 3.44) zugegriffen werden. Dazu ist es notwendig, dass als erstes ein Report geladen wird (siehe Abbildung 8.43).

```
'GRAPHEN DARSTELLEN:
'||||||||||||||||||||||||||||||||||||||||
Call PicLoad("Report")  'Lädt einen Report
```

Abbildung 8.43.: Laden eines Reports (Codezeilen 1043 – 1045), Autorin: Melanie Schulze

Als nächstes wird ein Array der Länge drei angelegt, um dort drei Farben zu speichern, die bei der Darstellung der Graphen verwendet werden (siehe Abbildung 8.44).

```
DIM farbe(3)  'Array der Länge drei erstellen, um
              'die Farben für die Graphen festzulegen
farbe(1) = "red"
farbe(2) = "blue"
farbe(3) = "green"
```

Abbildung 8.44.: Erstellen des „farbe“-Arrays (Codezeilen 1047 – 1050), Autorin: Melanie Schulze

Damit die Auswertung später übersichtlicher ist, soll als erstes ein Deckblatt entworfen werden (siehe Abbildung 8.45).

Bei allen graphischen Objekten (beispielsweise Text („FreeText“), Bild („FreeGraph“) oder Achsensystem („2D-Axis“) etc.) ist der Aufbau der Bearbeitung gleich. Als erstes muss das graphische Objekt erstellt („GraphObjNew()“), anschließend geöffnet („GraphObjOpen()“), kann dann bearbeitet und muss schließlich wieder geschlossen werden („GraphObjClose()“). Des Weiteren gibt es bei allen graphischen Objekten verschiedene Einstellmöglichkeiten (wie zum Beispiel die Schriftgröße („TxtSize“) oder die Position in x- („. . . PosX“) und y-Richtung („. . . PosY“) etc.).

Als erstes muss ein neues Arbeitsblatt innerhalb des Reports erstellt werden („GraphSheetNew()“). Dabei wird als Parameter der Name des Arbeitsblattes angegeben. Das Deckblatt soll eine Überschrift enthalten, den Namen der Messung, das aktuelle Datum sowie das Logo des DLR.

```
'Deckblatt erstellen:
'-----------------------------------------------------------------
Call GraphSheetNew("Deckblatt")  'Sheet für das Deckblatt öffnen

Call GraphObjNew("FreeText","ueberschrift")  'Überschrift-Objekt erstellen
Call GraphObjOpen("ueberschrift")  'Überschrift-Objekt öffnen

  TxtTxt = "Rotorversuchsstand"
  TxtSize = 14  'Schriftgröße einstellen
  TxtPosY = 80  'Position setzen

Call GraphObjClose("ueberschrift")  'Überschrift-Objekt schließen

Call GraphObjNew("FreeText","name_messung")  '"Name der Messung"-Objekt erstellen
Call GraphObjOpen("name_messung")  '"Name der Messung"-Objekt öffnen

  TxtTxt = name_messung
  TxtSize = 11  'Schriftgröße einstellen
  TxtPosY = 50  'Position setzen

Call GraphObjClose("name_messung")  '"Name der Messung"-Objekt schließen

Call GraphObjNew("FreeText","datum")  'Datum-Objekt erstellen
Call GraphObjOpen("datum")  'Datum-Objekt öffnen


  uhrzeit = Str(stunden) + ":" + Str(minuten) + ":" + Str(sekunden)
  aktuelles_Datum = Str((NOW), "#dd.mm.yyyy")  'Datumsformat ändern
  TxtTxt = Str(aktuelles_Datum) + " " + Str(uhrzeit)
  TxtSize = 7   'Schriftgröße einstellen
  TxtPosY = 20  'Position setzen

Call GraphObjClose("datum")  'Datum-Objekt schließen

Call GraphObjNew("FreeGraph", "logo_deckblatt")  'Deckblatt-Logo-Objekt erstellen
Call GraphObjOpen("logo_deckblatt")  'Deckblatt-Logo-Objekt öffnen

  MtaFileName = CurrentScriptPath + "\logo dlr 1.jpg"
  MtaPosX = 85    'Position setzen
  MtaPosY = 19.5
  MtaHeight = 24.46 'Größe einstellen
  MtaWidth = 20.75

Call GraphObjClose("logo_deckblatt")  'Deckblatt-Logo-Objekt schließen
'-----------------------------------------------------------------
```

Abbildung 8.45.: Erstellen des Deckblatts (Codezeilen 1052 – 1096), Autorin: Melanie Schulze

Ein Beispiel eines fertigen Deckblattes ist in Abbildung 8.46 zu sehen.

Abbildung 8.46.: Beispiel-Deckblatt, Autorin: Melanie Schulze

Hinter dem Deckblatt soll sich ein Arbeitsblatt befinden, das die Orte der Sensoren angibt (siehe Abbildung 8.47). Dieses Arbeitsblatt enthält eine Überschrift, das Logo des DLR, sowie die Orte der Sensoren.

```
'Blatt erstellen, das die Orte der Sensoren enthält:
'-------------------------------------------------------------
Call GraphSheetNew("Ort Sensoren")  'Sheet für das "Ort der Sensoren"-Blatt öffnen

Call GraphObjNew("FreeText","ueberschrift")  'Überschrift-Objekt erstellen
Call GraphObjOpen("ueberschrift")  'Überschrift-Objekt öffnen

  TxtTxt = "Orte der Sensoren"
  TxtSize = 6  'Schriftgröße einstellen
  TxtPosY = 95  'Position setzen

Call GraphObjClose("ueberschrift")  'Überschrift-Objekt schließen

Call GraphObjNew("FreeGraph", "logo")  'Logo-Objekt erstellen
Call GraphObjOpen("logo")  'Logo-Objekt öffnen

    MtaFileName = CurrentScriptPath + "\logo dlr 2.jpg"
    MtaPosX = 89  'Position setzen
    MtaPosY = 96
    MtaHeight = 6.6  'Größe einstellen
    MtaWidth = 20.75

Call GraphObjClose("logo")  'Logo-Objekt schließen

j = 0  'j: Zähler für die Position der Unterüberschriften (Sensor 1, Sensor 2...)

for i = 1 to anz_sensoren  'for-Schleife (33)

  Call GraphObjNew("FreeText","sensor"+Str(i))  '"Sensor-Name"-Objekt erstellen
  Call GraphObjOpen("sensor"+Str(i))  '"Sensor-Name"-Objekt öffnen

    TxtTxt =  name_sensoren(i)  'Unterüberschrift setzen
    TxtSize = 3.5  'Schriftgröße einstellen
    TxtUndl = True
    TxtPosY = 85-j

  Call GraphObjClose("sensor"+Str(i))  '"Sensor-Name"-Objekt schließen

  Call GraphObjNew("FreeText","ort"+Str(i))  '"Ort"-Objekt erstellen
  Call GraphObjOpen("ort"+Str(i))  '"Ort"-Objekt öffnen

    TxtTxt = ort_sensoren(i)  'Ort der Sensoren angeben
    TxtSize = 2.5  'Schriftgröße einstellen
    TxtPosY = 80-j

  Call GraphObjClose("ort"+Str(i))  '"Ort"-Objekt schließen

  j = j + 10

next  'Ende der for-Schleife (33)
'-------------------------------------------------------------
```

Abbildung 8.47.: Erstellen des Arbeitsblattes für die Sensor-Orte (Codezeilen 1098 – 1148), Autorin: Melanie Schulze

Ein Beispiel eines fertigen Arbeitsblattes ist in Abbildung 8.48 zu sehen.

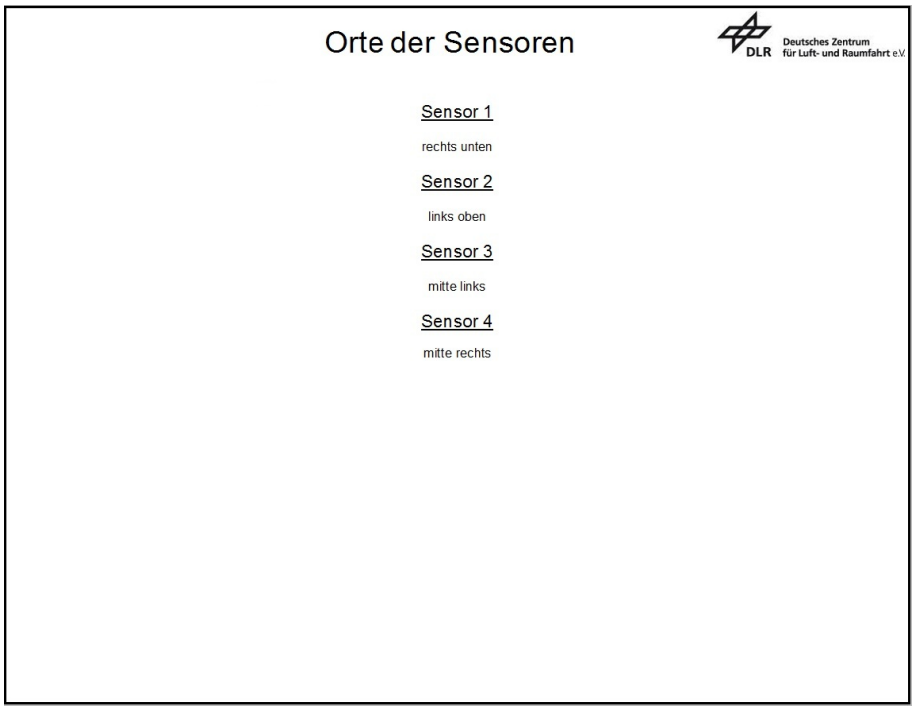


Abbildung 8.48.: Beispiel-Arbeitsblatt mit den Orten der Sensoren, Autorin: Melanie Schulze

Als nächstes folgt eine Seite, die die FFT der Kraftsensor-Messwerte enthält. Das Erstellen dieser Seite ist in Abbildung 8.49 dargestellt.
Die Seite erhält eine Überschrift, das Logo des DLR sowie zwei Koordinatensysteme. Das obere enthält die Amplituden der FFT und das untere enthält die Phase der FFT. Da sich das Erstellen der beiden Koordinatensysteme kaum unterscheidet, wird hier nur das Koordinatensystem für die Amplituden erläutert.

```
'Blatt erstellen, das die Kraftsensor-Werte enthält:
'-----------------------------------------------------------------
Call GraphSheetNew("Kraftsensor")  'Sheet für das Deckblatt öffnen

Call GraphObjNew("FreeText","ueberschrift")  'Überschrift-Objekt erstellen
Call GraphObjOpen("ueberschrift")  'Überschrift-Objekt öffnen

  TxtTxt = "FFT Kraftsensor"
  TxtSize = 4  'Schriftgröße einstellen
  TxtPosY = 95  'Position setzen

Call GraphObjClose("ueberschrift")  'Überschrift-Objekt schließen

Call GraphObjNew("FreeGraph", "logo")  'Logo-Objekt erstellen
Call GraphObjOpen("logo")  'Logo-Objekt öffnen

    MtaFileName = CurrentScriptPath + "\logo dlr 2.jpg"
    MtaPosX = 89  'Position setzen
    MtaPosY = 96
    MtaHeight = 6.6  'Größe einstellen
    MtaWidth = 20.75

Call GraphObjClose("logo")  'Logo-Objekt schließen

Call GraphObjNew("2D-Axis","koordinatensystem_kraft_ampl")  '"Koordinatensystem_kraft"-Objekt erstellen
Call GraphObjOpen("koordinatensystem_kraft_ampl")  '"Koordinatensystem_kraft"-Objekt öffnen

    D2AxisTop      = 10  'Position des Koordinatensystems setzen
    D2AxisBottom   = 55
    D2AxisLeft     = 13
    D2AxisRight    = 10
    D2AxisDispType = "Frame"  'Einstellen, dass das Koordinatensystem Gitterlinien anzeigen kann

Call GraphObjNew("2D-Curve","kurve_kraft_ampl")  '"Kurve_kraft"-Objekt erstellen
Call GraphObjOpen("kurve_kraft_ampl")  '"Kurve_kraft"-Objekt öffnen

  'Darzustellenden Channel auswählen:
  D2CChnYName = "[1]/FFT Amplitude Kraftsensor"
  D2CurveColor = "red"  'Setzen der Farbe

Call GraphObjClose("kurve_kraft_ampl")  '"Kurve_kraft"-Objekt schließen
```

Abbildung 8.49.: Erstellen des Arbeitsblattes mit der FFT der Kraftsensor-Messwerte (Codezeilen 1150 – 1190), Autorin: Melanie Schulze

Bei dem Koordinatensystem gibt es einige Einstellungen, wie die Skalierung der Achsen oder die Achsenbeschriftung, zu treffen. Diese sind in Abbildung 8.50 zu sehen.
Die Ergebnisse der FFT sollen nur in dem Frequenzbereich dargestellt werden, in dem auch gesweept wurde, bzw. bei einer Anregung mit Rauschen bis zu einer vorher vom Anwender ausgewählten Grenzfrequenz („f_rauschen“). Daher wird die Abszisse begrenzt. Bei einer Anregung mit einem Frequenz-Sweep soll die Achse jedoch nicht genau bei der Startfrequenz beginnen und bei der Endfrequenz enden, sondern es soll links und rechts noch ein wenig Platz bleiben, um das Ablesen der Werte zu erleichtern.
Die Ordinate wiederum wird automatisch skaliert.
Außerdem werden die Gitterlinien in Abszissen- und Ordinaten-Richtung gesetzt.

```
'###Einstellungen der Achsen vornehmen:###
 Call GraphObjOpen("koordinatensystem_kraft_ampl_XAxis1")  'x-Achse öffnen

  D2AxisXScaleType = "manual"  'Setzt die Skalierung von "automatisch" auf "manuell"

  if rauschen = 1 then
    D2AxisXEnd = f_rauschen  'Ende der x-Achse auf größte gerauschte Frequenz einstellen
    D2AxisXTick = f_rauschen / 10  'Intervall einstellen
  else
    if startfrequenz = endfrequenz then
      'Schnittpunkt der x-Achse mit der y-Achse festlegen:
      D2AxisXOrigin = startfrequenz-1
      'Startpunkt der x-Achse festlegen:
      D2AxisXBegin = startfrequenz-1
      'Endpunkt der x-Achse festlegen:
      D2AxisXEnd = endfrequenz+1
      'Intervall einstellen:
      D2AxisXTick = 0.5
    else
      'Schnittpunkt der x-Achse mit der y-Achse festlegen:
      D2AxisXOrigin = startfrequenz-0.05*(endfrequenz-startfrequenz)
      'Startpunkt der x-Achse festlegen:
      D2AxisXBegin = startfrequenz-0.05*(endfrequenz-startfrequenz)
      'Endpunkt der x-Achse festlegen:
      D2AxisXEnd = endfrequenz+0.05*(endfrequenz-startfrequenz)
      'Intervall einstellen:
      D2AxisXTick = (endfrequenz-startfrequenz) / 10
    end if
  end if
    D2AxisXColor = "black"     'Farbe der Skalierung
    D2AxisXTxt = "f [Hz]"      'Achsenbeschriftung
    D2AxisXTxtColor = "black"  'Farbe der Achsenbeschriftung

Call GraphObjClose("koordinatensystem_kraft_ampl_XAxis1")  'x-Achse schließen

Call GraphObjOpen("koordinatensystem_kraft_ampl_YAxis1")  'y-Achsen-Objekt öffnen

  D2AxisYColor = "black"  'Farbe der Skalierung
  D2AxisYTxt = "F [N]"    'Achsenbeschriftung
  D2AxisYTxtColor = "black"  'Farbe der Achsenbeschriftung

Call GraphObjClose("koordinatensystem_kraft_ampl_YAxis1")  'y-Achsen-Objekt schließen

D2AxisDisp(1) = "Grid"  'Setzen der Gitterlinien in x-Richtung
D2AxisDisp(2) = "Grid"  'Setzen der Gitterlinien in y-Richtung
'##########################################
```

Abbildung 8.50.: Einstellungen des Koordinatensystems der Amplitude der FFT der Kraftsensor-Messwerte (Codezeilen 1192 – 1229), Autorin: Melanie Schulze

Bei dem Koordinatensystem für die Phase unterscheidet sich lediglich die Skalierung der Ordinate von der in Abbildung 8.50. Die Skalierung der Ordinate soll hier nicht automatisch erfolgen, sondern von -180° bis 180° eingestellt werden (siehe Abbildung 8.51).

```
Call GraphObjOpen("koordinatensystem_kraft_phase_YAxis1")  'y-Achsen-Objekt öffnen

  D2AxisYScaleType = "manual"  'Setzt die Skalierung von "automatisch" auf "manuell"
  D2AxisYBegin = -180
  D2AxisYEnd = 180
  D2AxisYTick = 60  'Hauptintervall einstellen
  D2AxisYColor = "black"     'Farbe der Skalierung
  D2AxisYTxt = "Phase [°]"   'Achsenbeschriftung
  D2AxisYTxtColor = "black"  'Farbe der Achsenbeschriftung

 Call GraphObjClose("koordinatensystem_kraft_phase_YAxis1")  'y-Achsen-Objekt schließen
```

Abbildung 8.51.: Einstellungen der Ordinate der Phase der FFT der Kraftsensor-Messwerte (Codezeilen 1278 – 1288), Autorin: Melanie Schulze

Es folgt ein Beispiel einer Darstellung der FFT über die Kraftsensor-Messwerte (siehe Abbildung 8.52).

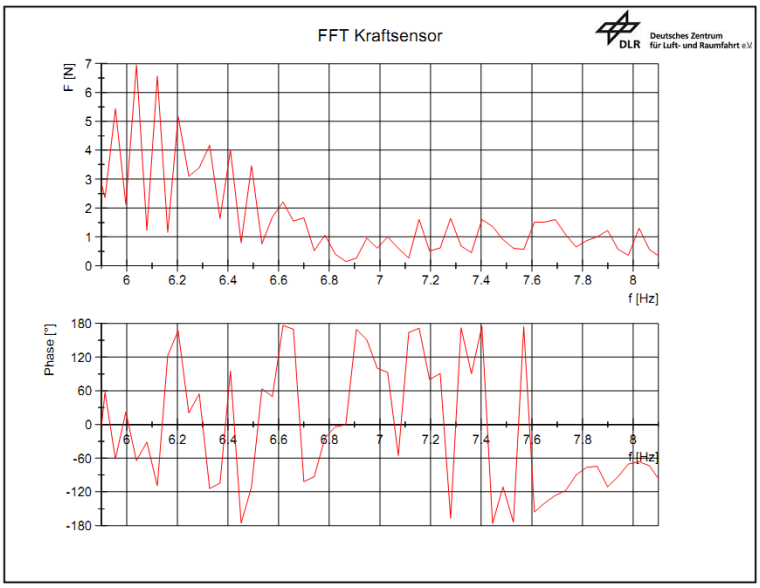


Abbildung 8.52.: FFT der Kraftsensor-Messwerte, Autorin: Melanie Schulze

Alle folgenden Anweisungen befinden sich innerhalb einer for-Schleife (siehe Bild Abbildung 8.53).
Diese for-Schleife ist dazu da, dass alle Anweisungen einmal für die Weg- und einmal für die Beschleunigungssignale ausgeführt werden.

```
for o = 1 to 2  'for-Schleife (63)

  ...

next  'Ende der for-Schleife (63)
```

Abbildung 8.53.: for-Schleife für Graphdarstellungen (Codezeilen 1297 - 2311), Autorin: Melanie Schulze

Zur besseren Übersicht soll als erstes jeweils ein Arbeitsblatt eingefügt werden, das die Weg- bzw. Beschleunigungssignale ankündigt (siehe Abbildung 8.54).

```
'Ankündigungs-Seite erstellen:
'---------------------------------------------------------------
if o = 1 then
  Call GraphSheetNew("Beschleunigung")  'neues Sheet öffnen
else
  Call GraphSheetNew("Weg")  'neues Sheet öffnen
end if

Call GraphObjNew("FreeText","auswertung")  'Überschrift1-Objekt erstellen
Call GraphObjOpen("auswertung")  'Überschrift1-Objekt öffnen

  TxtTxt = "Auswertung"
  TxtSize = 9  'Schriftgröße einstellen
  TxtPosY = 70  'Position setzen

Call GraphObjClose("auswertung")  'Überschrift1-Objekt schließen

Call GraphObjNew("FreeText","der")  'Überschrift2-Objekt erstellen
Call GraphObjOpen("der")  'Überschrift2-Objekt öffnen

  TxtTxt = "der"
  TxtSize = 9  'Schriftgröße einstellen
  TxtPosY = 60  'Position setzen

Call GraphObjClose("der")  'Überschrift2-Objekt schließen

if o = 1 then

  Call GraphObjNew("FreeText","beschleunigung")  'Überschrift3-Objekt erstellen
  Call GraphObjOpen("beschleunigung")  'Überschrift3-Objekt öffnen

    TxtTxt = "Beschleunigungssignale"
    TxtSize = 9  'Schriftgröße einstellen
    TxtPosY = 50  'Position setzen

  Call GraphObjClose("beschleunigung")  'Überschrift3-Objekt schließen

else

  Call GraphObjNew("FreeText","weg")  'Überschrift3-Objekt erstellen
  Call GraphObjOpen("weg")  'Überschrift3-Objekt öffnen

    TxtTxt = "berechneten Wegsignale"
    TxtSize = 9  'Schriftgröße einstellen
    TxtPosY = 50  'Position setzen

  Call GraphObjClose("weg")  'Überschrift3-Objekt schließen

end if
'---------------------------------------------------------------
```

Abbildung 8.54.: Erstellen des Ankündigungs-Arbeitsblattes (Codezeilen 1299 – 1348), Autorin: Melanie Schulze

Die beiden Ankündigungsblätter sehen wie folgt aus (siehe Abbildung 8.55).

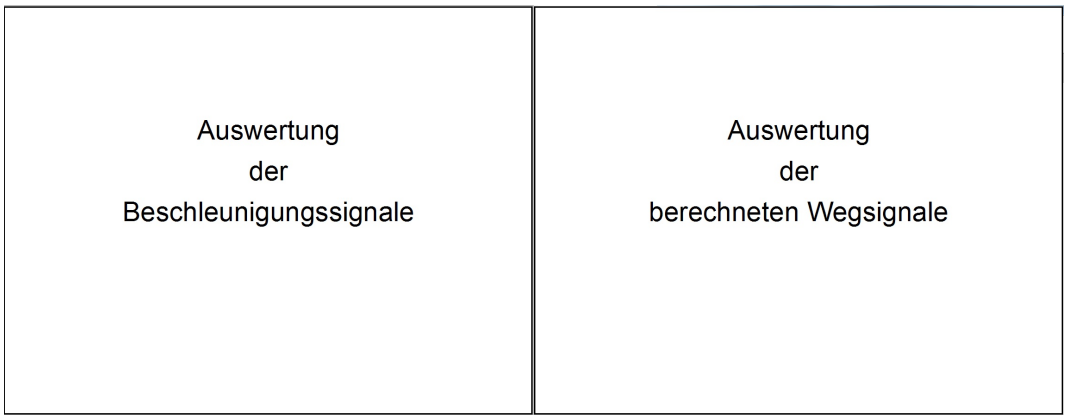


Abbildung 8.55.: Ankündigungs-Arbeitsblätter, Autorin: Melanie Schulze

Als nächstes wird jeweils ein Arbeitsblatt erstellt, das die Zusammenfassung der Auswertung darstellt (siehe Abbildung 8.56 bis Abbildung 8.59). Dieses soll eine Überschrift und das Logo des DLR enthalten sowie die Amplitude der Anregung und die größte aufgetretene Amplitude sowie die Frequenz und die Sensorrichtung, bei der diese aufgetreten ist, pro Sensor angeben.

In Abbildung 8.56 wird zunächst jeweils ein neues Arbeitsblatt aufgemacht und diesem das Logo, die Überschrift sowie die Amplitude der Anregung hinzugefügt.

```
'Zusammenfassungsblatt erstellen:
'-------------------------------------------------------------
if o = 1 then
  'Sheet für die Zusammenfassung öffnen:
  Call GraphSheetNew("Zusammenfassung Beschleunigung")
else
  'Sheet für die Zusammenfassung öffnen:
  Call GraphSheetNew("Zusammenfassung Weg")
end if

Call GraphObjNew("FreeGraph", "logo")  'Logo-Objekt erstellen
Call GraphObjOpen("logo")  'Logo-Objekt öffnen

    MtaFileName = CurrentScriptPath + "\logo dlr 2.jpg"
    MtaPosX = 89  'Position setzen
    MtaPosY = 96
    MtaHeight = 6.6  'Größe einstellen
    MtaWidth = 20.75

Call GraphObjClose("logo")  'Logo-Objekt schließen

Call GraphObjNew("FreeText","zusammenfassung")  'Überschrift-Objekt erstellen
Call GraphObjOpen("zusammenfassung")  'Überschrift-Objekt öffnen

  TxtTxt = "Größte aufgetretene Amplituden (pro Sensor)"
  TxtSize = 5   'Schriftgröße einstellen
  TxtPosX = 46
  TxtPosY = 95  'Position setzen

Call GraphObjClose("zusammenfassung")  'Überschrift-Objekt schließen

Call GraphObjNew("FreeText", "ampl_anr")  '"Amplitude der Anregung"-Objekt
                                          'erstellen
Call GraphObjOpen("ampl_anr")  '"Amplitude der Anregung"-Objekt öffnen

  TxtTxt = "Amplitude der Anregung:  " + Str(amplitude_anregung) + " V"
  TxtSize = 4
  TxtPosX = 20.5
  TxtPosY = 88

Call GraphObjClose("ampl_anr")  '"Amplitude der Anregung"-Objekt schließen
```

Abbildung 8.56.: Zusammenfassungsblatt erstellen, Teil 1 (Codezeilen 1350 – 1387), Autorin: Melanie Schulze

In Abbildung 8.57 wird für jeden Sensor eine Unterüberschrift erstellt. Es sollen vier Sensoren untereinander und zwei nebeneinander passen. Das heißt, dass die y-Position bis zum vierten Sensor verändert werden muss und danach wieder von vorne beginnt. Die x-Position ist für die ersten vier Sensoren gleich, muss ab dem fünften Sensor geändert werden und bleibt ab dann wieder gleich (siehe Abbildung 8.60).

```
j = 0  'j: Zähler für die Position der Unterüberschriften (Sensor 1, Sensor 2...)
m = 0  'm: Zähler für die Position der restlichen Texte

for i = 1 to anz_sensoren  'for-Schleife (60)

  Call GraphObjNew("FreeText","sensor"+Str(i))  '"Sensor-Name"-Objekt erstellen
  Call GraphObjOpen("sensor"+Str(i))  '"Sensor-Name"-Objekt öffnen

    TxtTxt =  name_sensoren(i)  'Unterüberschrift setzen
    TxtSize = 4  'Schriftgröße einstellen
    TxtUndl = True

    '###Position setzen:###
    if (i=1)  then
      TxtPosY = 83
    elseif (i=5) then
      TxtPosY = 83
      j = 0
    else
      TxtPosY = 83-j
    end if

    if (i<5) then
      TxtPosX = 10
    else
      TxtPosX = 60
    end If
    '########################

  Call GraphObjClose("sensor"+Str(i))  '"Sensor-Name"-Objekt schließen

  j = j + 20
```

Abbildung 8.57.: Zusammenfassungsblatt erstellen, Teil 2 (Codezeilen 1389 – 1420), Autorin: Melanie Schulze

Nun wird für jeden Sensor die Unterüberschrift „Maximale Amplitude:“ erstellt (siehe Abbildung 8.58).

```
Call GraphObjNew("FreeText","amplitude_text"+Str(i))  'Amplituden-Objekt erstellen
Call GraphObjOpen("amplitude_text"+Str(i))  'Amplituden-Objekt öffnen

  TxtTxt = "Maximale Amplitude:"
  TxtSize = 3   'Schriftgröße einstellen

  '###Position setzen:###
  if (i=1)  then
    TxtPosY = 78
  elseif (i=5) then
    TxtPosY = 78
    m = 0
  else
    TxtPosY = 78-m
  end if

  if (i<5) then
    TxtPosX = 13.5
  else
    TxtPosX = 63.5
  end If
  '########################

Call GraphObjClose("amplitude_text"+Str(i))  'Amplituden-Objekt schließen
```

Abbildung 8.58.: Zusammenfassungsblatt erstellen, Teil 3 (Codezeilen 1422 – 1445), Autorin: Melanie Schulze

Als nächstes wird die größte aufgetretene Amplitude pro Sensor ausgelesen, auf drei Nachkommastellen gerundet und mit der jeweiligen Einheit auf das Zusammenfassungsblatt geschrieben (siehe Abbildung 8.59).

```
Call GraphObjNew("FreeText","amplitude_max"+Str(i))  '"Maximale Amplitude"-Objekt erstellen
Call GraphObjOpen("amplitude_max"+Str(i))  '"Maximale Amplitude"-Objekt öffnen

  DIM ampl_max

  'Maximale Amplitude des jeweiligen Sensors auslesen:
  if o = 1 then  'Beschleunigung
    ampl_max = CHD(1,"[2]/[" + Str(i + anz_sensoren*12) + "]")
  else  'Weg
    ampl_max = CHD(1,"[3]/[" + Str(i + anz_sensoren*15) + "]")
  end if

  ampl_max = Round(ampl_max, 3)  'maximale Amplitude auf drei Nachkommastellen runden

  if o = 1 then
    TxtTxt = Str(ampl_max) + " m/s²"
  else
    TxtTxt = Str(ampl_max) + " mm"
  end if

  TxtSize = 3   'Schriftgröße einstellen

  '###Position setzen:###
  if (i=1)  then
    TxtPosY = 78
  elseif (i=5) then
    TxtPosY = 78
    m = 0
  else
    TxtPosY = 78-m
  end if

  if (i<5) then
    TxtPosX = 28
  else
    TxtPosX = 78
  end If
  '########################

Call GraphObjClose("amplitude_max"+Str(i))  '"Maximale Amplitude"-Objekt schließen
```

Abbildung 8.59.: Zusammenfassungsblatt erstellen, Teil 4 (Codezeilen 1447 – 1486), Autorin: Melanie Schulze

Anschließend werden für jeden Sensor die Unterüberschriften „Sensor-Richtung:“ und „Frequenz:“ erstellt sowie die jeweiligen Werte ausgelesen und dem Zusammenfassungsblatt zugefügt. Da sich dieses kaum vom eben gezeigten Vorgehen für die maximale Amplitude unterscheidet, wird an dieser Stelle auf den Programmcode verzichtet.

Ein fertiges Zusammenfassungsblatt ist beispielhaft in Abbildung 8.60 zu sehen.

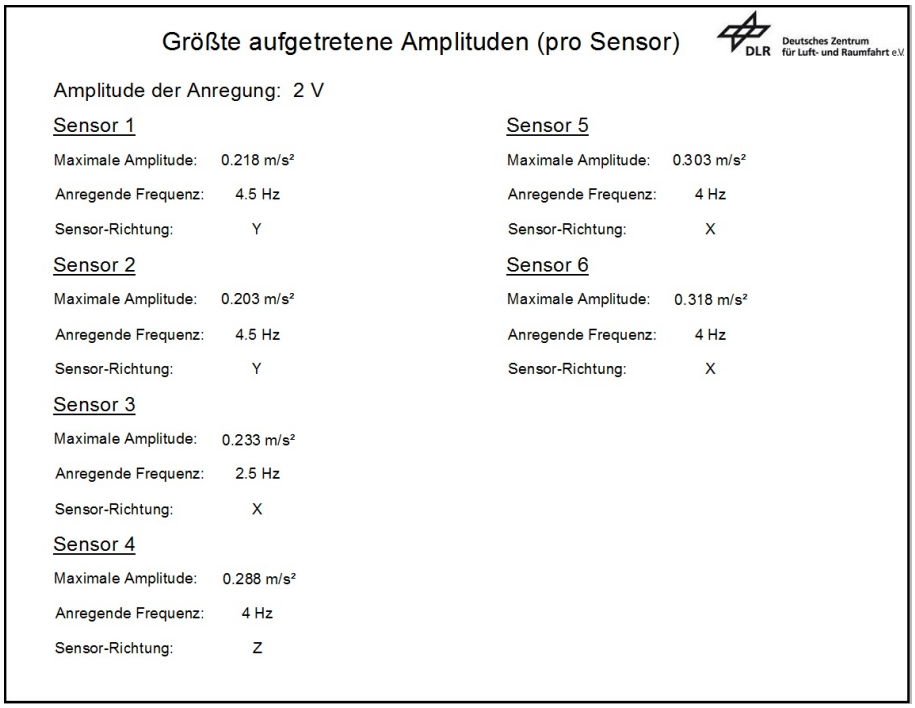


Abbildung 8.60.: Beispiel-Zusammenfassungsblatt, Autorin: Melanie Schulze

Als nächstes sollen die Auswertungen der Signale dargestellt werden. Als erstes werden dabei Arbeitsblätter erstellt, die die Ergebnisse der FRF zeigen (siehe Abbildung 8.61 – Abbildung 8.65).

Hierfür wird zunächst zum einen der Index festgelegt, ab dem die Kanäle mit den Ergebnissen der FRF innerhalb der Gruppe beginnen. Zum anderen wird der Index festgelegt, ab dem die Kanäle mit den Maxima-Werte der FRF beginnen (siehe Abbildung 8.61).

```
'SHEETS FÜR DIE FRF ERSTELLEN:
'////////////////////////////////////////////////
'n: Index, ab dem die FRF-Werte beginnen,
'm: Index, ab dem die FRF-Maxima-Werte beginnen:
if o = 1 then  'Beschleunigung
  n = 1 + anz_sensoren*13
  m = 1 + anz_sensoren*19
else  'Weg
  n = 1 + anz_sensoren*16
  m = 1 + anz_sensoren*22
end if
```

Abbildung 8.61.: Berechnung benötigter Indexe zur Darstellung der Ergebnisse der FRF (Codezeilen 1618 – 1627), Autorin: Melanie Schulze

Anschließend wird pro Sensor ein neues Arbeitsblatt aufgemacht, das das Logo des DLR sowie eine Überschrift erhält (siehe Abbildung 8.62).

```
for i = 1 to anz_sensoren  'for-Schleife (36)

  'Sheet für die FRF öffnen:
  if o = 1 then
    Call GraphSheetNew("FRF " +  name_sensoren(i) + " (Beschl.)")
  else
    Call GraphSheetNew("FRF " +  name_sensoren(i) + " (Weg)")
  end if

  Call GraphObjNew("FreeGraph", "logo")  'Logo-Objekt erstellen
  Call GraphObjOpen("logo")  'Logo-Objekt öffnen

    MtaFileName = CurrentScriptPath + "\logo dlr 2.jpg"
    MtaPosX = 89  'Position setzen
    MtaPosY = 96
    MtaHeight = 6.6  'Größe einstellen
    MtaWidth = 20.75

  Call GraphObjClose("logo")  'Logo-Objekt schließen

  Call GraphObjNew("FreeText","frf")  'Überschrift-Objekt erstellen
  Call GraphObjOpen("frf")  'Überschrift-Objekt öffnen

    TxtTxt = "FRF " +  name_sensoren(i)
    TxtSize = 4   'Schriftgröße einstellen
    TxtPosY = 95  'Position setzen

  Call GraphObjClose("frf")  'Überschrift-Objekt schließen
```

Abbildung 8.62.: Erstellen eines Arbeitsblattes für die Ergebnisse der FRF (Codezeilen 1629 – 1656), Autorin: Melanie Schulze

Auf einem Arbeitsblatt wird immer die Auswertung eines Sensors dargestellt, d.h. dass sich in einem Koordinatensystem drei Graphen befinden (X-, Y- und Z-Richtung des Sensors). Für die drei Graphen werden die Farben rot, blau und grün verwendet.
Für diese drei Farben wird im Folgenden eine Legende erstellt (siehe Abbildung 8.63).
Die Legende besteht aus dem Text „X“ mit einem roten Quadrat daneben, dem Text „Y“ mit einem blauen Quadrat daneben und dem Text „Z“ mit einem grünen Quadrat daneben.

```
'Legende erstellen:
'---------------------------------------------------------------
for l = 1 to 3  'for-Schleife (37)

  Call GraphObjNew("FreeText","schrift" + Str(l))  'Graphbeschriftung;
                                                   '"schrift"-Objekt erstellen
  Call GraphObjOpen("schrift" + Str(l))  '"schrift"-Objekt öffnen

    TxtTxt = sensorrichtung(l)  'Setzen des Textes (X, Y, Z)
    TxtSize = 4   'Schriftgröße einstellen
    TxtPosX = 91  'Position setzen
    TxtPosY = 95 - l*10

  Call GraphObjClose("schrift" + Str(l))  '"schrift"-Objekt schließen

  Call GraphObjNew("FreeFrame","farbe" + Str(l))  '"Farbe"-Objekt erstellen
  Call GraphObjOpen("farbe" + Str(l))  '"Farbe"-Objekt öffnen

    AreaBackColor = farbe(l)  'Setzen der Farbe (rot, blau, grün)
    AreaIso = true  'Rechteck soll ein Quadrat sein
    AreaLinePt(1) = 86          'Abstand der linken unteren Ecke des
                                'Quadrats zur linken Seite des Blattes
    AreaLinePt(2) = 93.5 - 10*l  'Abstand der linken unteren Ecke des
                                'Quadrats zur unteren Seite des Blattes
    AreaLinePt(3) = 89          'Abstand der rechten oberen Ecke des
                                'Quadrats zur linken Seite des Blattes
    AreaLinePt(4) = 96.5 - 10*l  'Abstand der rechten oberen Ecke des
                                'Quadrats zur unteren Seite des Blattes

  Call GraphObjClose("farbe" + Str(l))  '"Farbe"-Objekt schließen

next  'Ende der for-Schleife (37)
'---------------------------------------------------------------
```

Abbildung 8.63.: Legende für die Ergebnisse der FRF erstellen (Codezeilen 1658 – 1685), Autorin: Melanie Schulze

Als nächstes sollen auf dem Arbeitsblatt die Maxima angegeben werden (siehe Abbildung 8.64).
Zunächst wird die Überschrift für die Maxima gesetzt. Als nächstes werden die x- und y-Werte der Maxima ausgelesen, auf drei Nachkommastellen gerundet und anschließend auf das Arbeitsblatt geschrieben. Das Ganze wird hier für die X-Richtung des Sensors vorgenommen. Da das Vorgehen mit den Sensorrichtungen „Y“ und „Z“ bis aus Indizes identisch ist, wird darauf verzichtet, diese hier vorzustellen.

```
'Maxima angeben:
'-----------------------------------------------------------------
DIM max_Xx, max_Xy, max_Yx, max_Yy, max_Zx, max_Zy

for l = 1 to 3  'for-Schleife (63)

  Call GraphObjNew("FreeText","max" + Str(l))  'Maximabeschriftung; "max"-Objekt-erstellen
  Call GraphObjOpen("max" + Str(l))  '"max"-Objekt öffnen

    TxtTxt = "Maxima " + sensorrichtung(l) + ":"  'Setzen des Textes (mit X, Y, Z)
    TxtSize = 3   'Schriftgröße einstellen
    TxtPosX = 90.5  'Position setzen
    TxtPosY = 73.5 - l*17

  Call GraphObjClose("max" + Str(l))  '"max"-Objekt schließen

  Call GraphObjNew("FreeText","max_x"+Str(l))  'Maxima; "Maxima, X-Rtg"-Objekt erstellen
  Call GraphObjOpen("max_x"+Str(l))  '"Maxima, X-Rtg"-Objekt öffnen

    'x-Werte der Maxima in X-Richtung auslesen:
    if o = 1 then
      max_Xx = CHD(l,"[2]/[" + Str(m) + "]")
    else
      max_Xx = CHD(l,"[3]/[" + Str(m) + "]")
    end if
    max_Xx = Round(max_Xx, 3)  'runden auf drei Nachkommastellen
    'y-Werte der Maxima in X-Richtung auslesen:
    if o = 1 then
      max_Xy = CHD(l,"[2]/[" + Str(m+1) + "]")
    else
      max_Xy = CHD(l,"[3]/[" + Str(m+1) + "]")
    end if
    max_Xy = Round(max_Xy, 3)  'runden auf drei Nachkommastellen

    if o = 1 then
      TxtTxt = "(" + Str(max_Xx) + " Hz / " + Str(max_Xy) + " (m/s²)/N)"
    else
      TxtTxt = "(" + Str(max_Xx) + " Hz / " + Str(max_Xy) + " m/N)"
    end if
    TxtSize = 2.5  'Schriftgröße einstellen
    TxtPosX = 89.5   'Position setzen
    TxtPosY = 52.5-(l-1)*4

  Call GraphObjClose("max_x"+Str(l))  '"Maxima, X-Rtg"-Objekt schließen
```

Abbildung 8.64.: Maxima der FRF angeben (Codezeilen 1687 – 1730), Autorin: Melanie Schulze

Nachdem die Maxima angegeben wurden, sollen nun die Koordinatensysteme eingefügt werden. Dieses unterscheidet sich kaum vom Einfügen der Koordinatensysteme auf der Seite der FFT der Kraftsensor-Messwerte (siehe Abbildung 8.49 bis Abbildung 8.51) und wird daher hier nicht erläutert.

In Abbildung 8.65 ist zu erkennen, dass der Index „n“, der angibt, wo die Kanäle mit den Werten der FFT liegen, nach jedem Durchlauf für den nächsten Sensor angepasst werden muss.

```
if o = 1 then  'Beschleunigung
  n = 1 + anz_sensoren*13 + i*6
else  'Weg
  n = 1 + anz_sensoren*16 + i*6
end if

next  'Ende der for-Schleife (36)
'////////////////////////////////////
```

Abbildung 8.65.: Anpassen des Indexes „n“ (Codezeilen 1952 – 1960), Autorin: Melanie Schulze

Ein fertiges Arbeitsblatt mit der Auswertung der FRF ist beispielhaft in Abbildung 8.66 zu sehen.

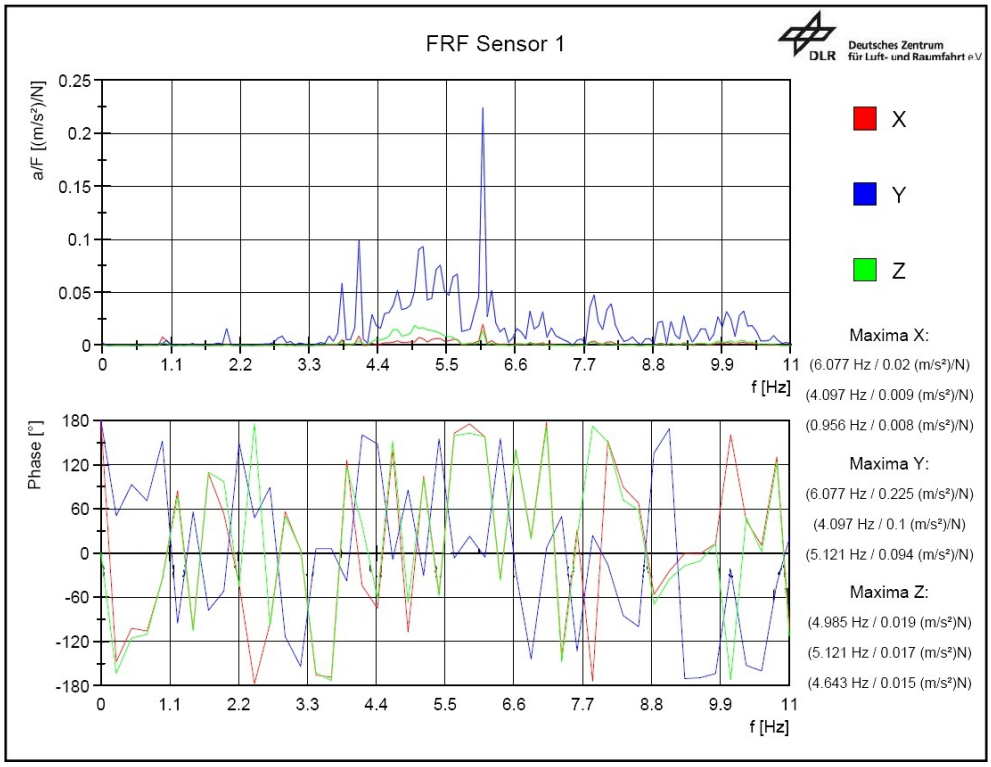


Abbildung 8.66.: Beispiel-Auswertungsblatt FRF, Autorin: Melanie Schulze

Nach den Ergebnissen der FRF sollen nun die Ergebnisse der FFT dargestellt werden. Auf die genaue Erläuterung dessen soll hier verzichtet werden, da sich die Darstellung kaum von der der Ergebnisse der FRF unterscheidet.

Ein Beispiel-Auswertungsblatt für die FFT ist in Abbildung 8.67 zu sehen.

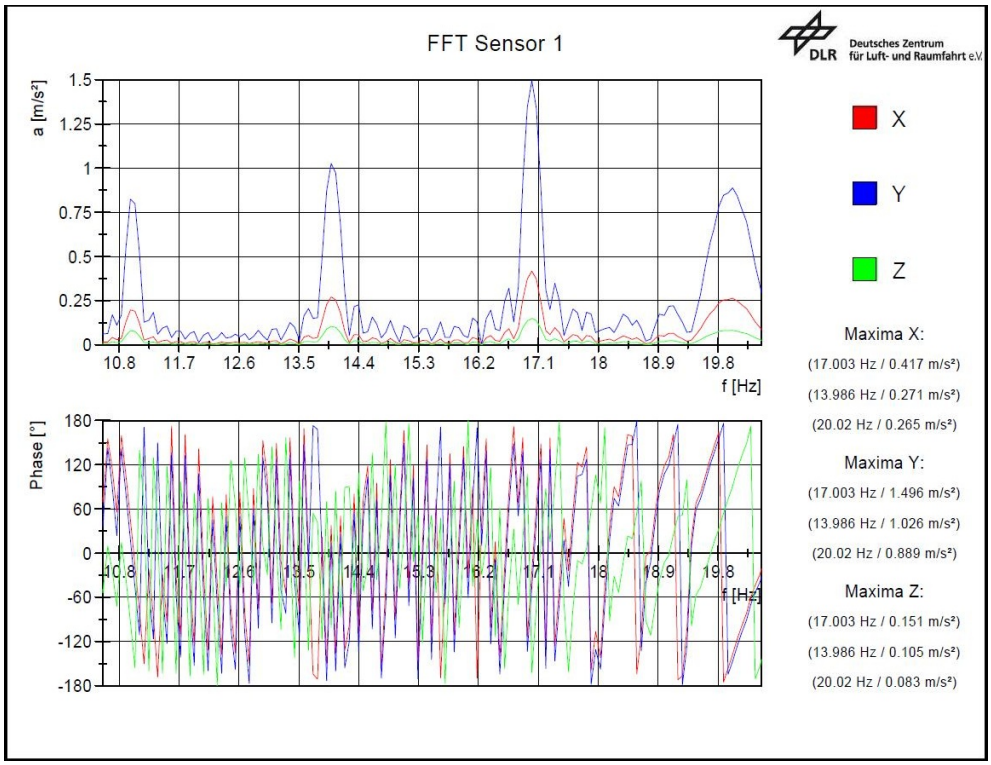


Abbildung 8.67.: Beispiel-Auswertungsblatt der Ergebnisse der FFT, Autorin: Melanie Schulze

In dem Modul REPORT ist von Anfang an ein Arbeitsblatt vorhanden, das „Sheet 1“ heißt. Immer, wenn ein neues Arbeitsblatt benötigt wird, wird es dem Report hinzugefügt. Dadurch bleibt am Ende das eine Arbeitsblatt vom Anfang über. Dieses wird nicht mehr gebraucht und daher gelöscht.
Außerdem muss der Report aktualisiert werden. Erst dann werden alle Inhalte der Arbeitsblätter für den Anwender sichtbar gemacht (siehe Abbildung 8.68).

```
Call PicUpdate  'aktualisiert den Report
Call GraphSheetDelete("Sheet 1")  'erstes Sheet löschen
'||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

Abbildung 8.68.: Aktualisieren des Reports und Löschen des ersten Arbeitsblattes (Codezeilen 2313 – 2315), Autorin: Melanie Schulze

Es besteht die Möglichkeit, die Auswertungen aus dem Report als .pdf-Datei abzuspeichern (siehe Abbildung 8.69).
Der Name der .pdf-Datei soll den Namen der Messung, das aktuelle Datum sowie die aktuelle Uhrzeit enthalten. Dazu ist es zunächst notwendig, dass in der Uhrzeit die

Doppelpunkte durch Semikolons ersetzt werden, da in einem Dateinamen keine Doppelpunkte erlaubt sind. Des Weiteren wird das Datumsformat geändert, damit die Auswertungen innerhalb des Auswertungsordners nach Datum sortiert werden können. Falls beim Erstellen der .pdf-Datei ein Fehler auftritt, wird eine Meldung an den Anwender ausgegeben.

```
'SPEICHERN DES REPORTS ALS PDF-DATEI:
'////////////////////////////////////////////////////////////////////////////////////////////////////
if pdf = 1 then

uhrzeit = Str(stunden) + ";" + Str(minuten) + ";" + Str(sekunden)  'Doppelpunkte in der Uhrzeit durch
                                                                   'Semikolons ersetzen
aktuelles_Datum = Str((NOW),"#yyyy-mm-dd") + " " + Str(uhrzeit)  'Datumsformat ändern

Dim PDFFile
'Aktuellen Script-Pfad verwenden sowie Namen und Datum anhängen
PDFFile = CurrentScriptPath + "\PDF Dateien\" + name_messung + " " + aktuelles_Datum + ".pdf"

On Error Resume Next
Call PicPDFExport(PDFFile)
if (Err.Number <> 0) then  'Falls ein Fehler beim Erstellen auftritt:
  Call MsgBoxDisp("PDF-Dokument konnte nicht erstellt werden.")  'Meldung an den Anwender ausgeben
end if

end if
'////////////////////////////////////////////////////////////////////////////////////////////////////
```

Abbildung 8.69.: Speichern des Reports als .pdf-Datei (Codezeilen 2317 – 2337), Autorin: Melanie Schulze

Ob diese .pdf-Datei erstellt werden soll oder nicht, wird in dem *LabVIEW*-Programm festgelegt und steht hier in der Variablen „pdf“.

9. Umwandeln des Beschleunigungssignals in ein Wegsignal

Um später den gefundenen Resonanzfrequenzen die momentane Auslenkung zuzuordnen, sollen die Messdaten der Beschleunigungssensoren in ein Wegsignal überführt werden. Hierzu wurden verschiedene Ansätze verwendet, die fast alle zu keinem befriedigenden Ergebnis führten. Diese Ansätze sollen in diesem Kapitel kurz erläutert werden.

Alle Ansätze haben folgenden Grundgedanken gemeinsam: Die Beschleunigung ist die Änderung der Geschwindigkeit pro Zeit, d.h. sie ist die zeitliche Ableitung der Geschwindigkeit. Da außerdem die Geschwindigkeit die Änderung des Weges pro Zeit ist, also die zeitliche Ableitung des Weges, kann man zusammenfassend sagen, dass die Beschleunigung die zweite Ableitung des Weges nach der Zeit ist (siehe (9.1)).

$$a(t) = \frac{d^2 s(t)}{dt^2} \tag{9.1}$$

Somit kann man auch im Umkehrschluss sagen, dass der Weg das doppelte Integral der Beschleunigung ist (siehe (9.2)).

$$s(t) = \iint a(t) dt dt \tag{9.2}$$

Da es in *LabVIEW* möglich ist, zu integrieren, ist der erste Ansatz, zwei Integratorblöcke hintereinander zu schalten (siehe Abbildung 9.1).
LabVIEW berechnet das Integral numerisch. Aus diesem Grund muss eine Integrationsmethode ausgewählt werden. Testweise wird hier die Trapezregel („Trapezoidal Rule“) verwendet. Des Weiteren muss die Schrittweite angegeben werden. Dazu wird das „dt“, also das Intervall zwischen zwei Datenpunkten des Signals, angeschlossen.

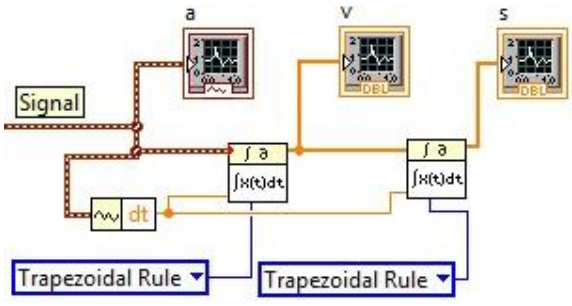


Abbildung 9.1.: Zweimaliges Integrieren in *LabVIEW*, Autorin: Melanie Schulze

Zu Testzwecken wird ein Sinus generiert, um diesen zwei Mal zu integrieren (siehe Abbildung 9.2).

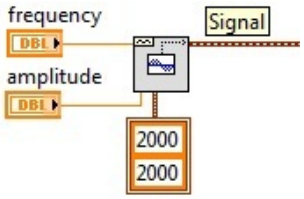


Abbildung 9.2.: Erzeugen eines Sinussignals mit veränderlicher Frequenz und Amplitude, Autorin: Melanie Schulze

Es wird ein einfacher Sinus verwendet, da von diesem das Ergebnis der Integration bekannt ist. Für die Geschwindigkeit wird ein negatives cos-Signal erwartet und für den Weg ein negatives sin-Signal. Als Ergebnis ergibt sich jedoch Folgendes (siehe Abbildung 9.3).

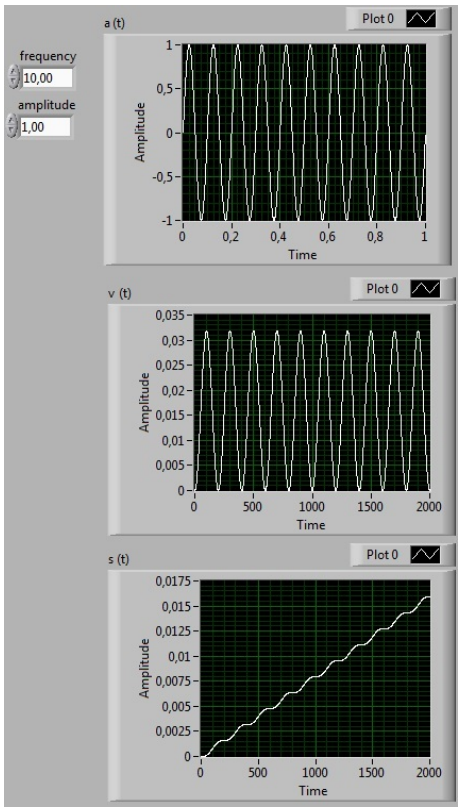


Abbildung 9.3.: Ergebnis des zwei Mal integrierten Sinus-Signals, Autorin: Melanie Schulze

Dieses Ergebnis entspricht nicht den Erwartungen. Schon in dem Geschwindigkeits-Diagramm ist zu sehen, dass es nur noch positive Werte gibt und keinen um Null schwingenden Cosinus.

Da dieses Ergebnis nicht so ist wie erwartet, wird folgendes versucht:
Es wird angenommen, dass zwischen zwei gemessenen Datenpunkten eine gleichmäßig beschleunigte Bewegung vorliegt. Dieses entspricht nicht der Realität. Da die Punkte aber so dicht beieinander liegen, kann dieses näherungsweise so angenommen werden.
Für die gleichmäßig beschleunigte Bewegung gelten folgende Formeln (siehe (9.3) und (9.4)):

$$s(t) = 0,5 \cdot a \cdot t^2 + v_0 \cdot t + s_0 \tag{9.3}$$

$$v(t) = a \cdot t + v_0 \tag{9.4}$$

Diese beiden Formeln werden nun auf jeden einzelnen Punkt der Messung angewendet. v_0 ist dabei jeweils die für den vorherigen Punkt berechnete Geschwindigkeit. Ebenso verhält es sich mit dem s_0. Dieses wird mit einem Sinus in *Microsoft Excel* getestet. Dabei ergibt sich das folgende Ergebnis (siehe Abbildung 9.4).

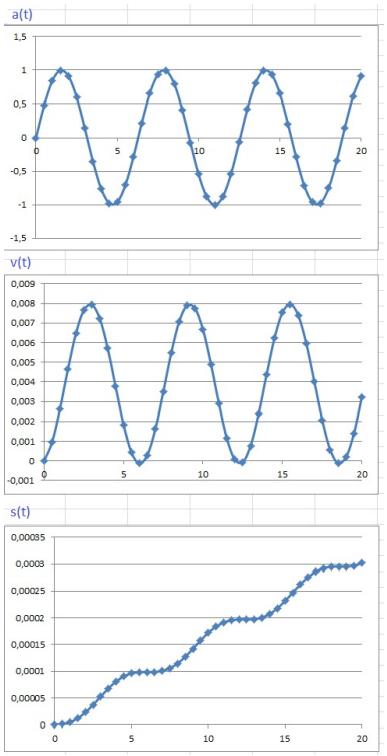


Abbildung 9.4.: Ergebnis der Berechnung mit den Formeln (9.3) und (9.4), Autorin: Melanie Schulze

Auch dieses Ergebnis entspricht nicht den Erwartungen. Es ist jedoch deutlich zu erkennen, dass dieses Ergebnis vom Verlauf her genau dem Ergebnis der Integration in *LabVIEW* (siehe Abbildung 9.3) entspricht.
Bei der Berechnung mit den Formeln (9.3) und (9.4) wurde jeweils für s_0 und v_0 des ersten Punktes 0 angenommen. Nimmt man dort andere Startwerte an, so ändert sich auch der Verlauf der Graphen v(t) und s(t).

Werden nun die richtigen Startwerte eingegeben, ergibt sich sowohl bei der Berechnung mit den Formeln als auch bei der Integration in *LabVIEW* der erwartete Verlauf. Diese Erkenntnis führt zu folgendem Problem: Man kennt die Anfangsbedingungen des Systems nicht. Betrachtet man beispielsweise das Feder-Masse-Pendel (siehe Kapitel 5), so kann man festlegen, wo $s = 0$ ist. Nimmt man an, dass dieses genau in der Ruhelage ist, so hat man an diesem Punkt $s = 0$, aber $v = max$. Im allerersten Moment ist an diesem Punkt zwar auch $v = 0$, aber auch dieses hilft nicht weiter, da die Messwerte ursprünglich immer erst im eingeschwungenen Zustand aufgenommen wurden, wodurch an dieser Stelle nicht mehr $v = 0$ galt.

Um dieses Problem zu umgehen, soll nicht mehr im Zeit-, sondern im Frequenzbereich integriert werden. Hierfür wird mit den Messwerten zuerst die FRF durchgeführt. Somit ergibt sich für den Betrag ein Koordinatensystem, bei dem sich die Frequenz auf der Abszisse und der Weg pro Kraft auf der Ordinate (siehe hierzu auch Abbildung 8.66) befindet. Bei der Phase befindet sich die Frequenz auf der Abszisse und die Gradzahl auf der Ordinate.
Um im Frequenzbereich zu integrieren, muss einerseits die Phase um -90° verschoben werden. Andererseits muss jeder Punkt des Betrages durch $\omega = 2\ \pi\ f$ geteilt werden. Um zwei Mal zu integrieren ergibt sich entsprechend -180° für die Phase und $4\ \pi^2\ f^2$ als Divisor für den Betrag.
Das Problem bei diesem Verfahren ist, dass sich für kleine Frequenzen große Fehler ergeben. Da der Shaker aber gerade bei kleinen Frequenzen betrieben werden soll, erweist sich dieses Verfahren ebenfalls als ungeeignet.

Bei dem letzten Ansatz geschieht das Integrieren wieder im Zeitbereich und mittels *LabVIEW*. Anstatt jedoch die Anfangsbedingungen herauszufinden, wird nach dem Integrieren jeweils der Mittelwert des Signals abgezogen (siehe Abbildung 9.5).

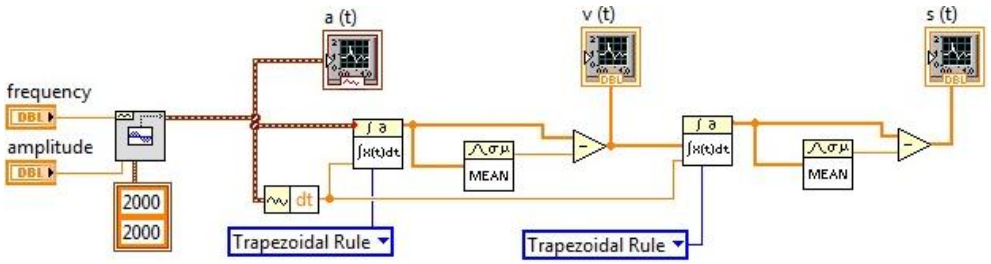


Abbildung 9.5.: Zweimaliges Integrieren bei *LabVIEW* mit Abziehen des Mittelwertes, Autorin: Melanie Schulze

Durch dieses Verfahren ergibt sich das erwartete Ergebnis (siehe Abbildung 9.6)

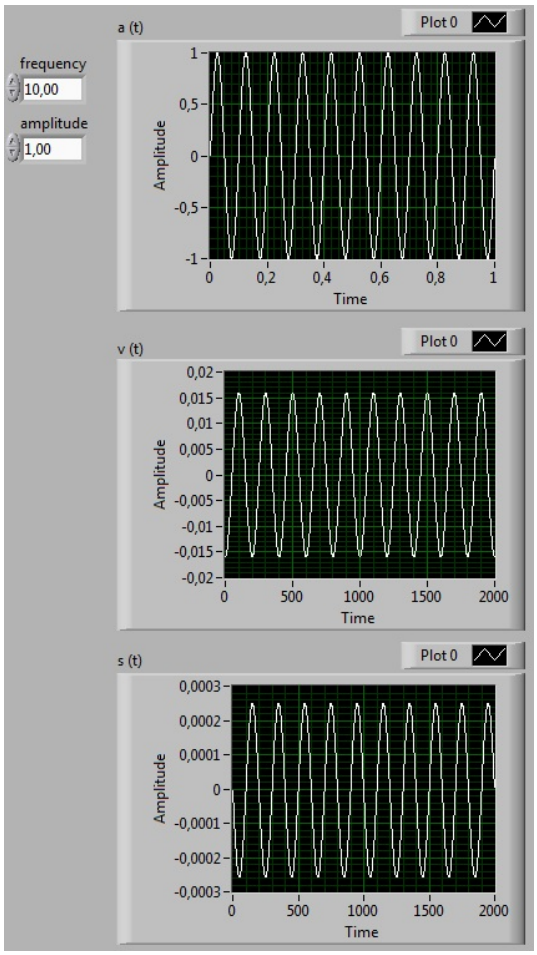


Abbildung 9.6.: Ergebnis des zwei Mal integrierten Sinus-Signals mit Abziehen des Mittelwertes, Autorin: Melanie Schulze

Bei einem reinen Sinus funktioniert dieses Verfahren. Bei den Messwerten handelt es sich aber nicht um reine Sinussignale. Näherungsweise kann man das Verfahren trotzdem

verwenden. Dieses geht jedoch nur, weil man weiß, dass sich das vom Shaker bewegte Modell zwar hin und her bewegt, jedoch nicht seinen Ort insgesamt wechselt.
Es muss außerdem darauf geachtet werden, dass bei den realen Messwerten auch schon vor der ersten Integration der Mittelwert abgezogen wird. In dem Beispiel aus Abbildung 9.5 musste dieses nicht gemacht werden, da der Sinus keinen Offset hatte. Bei den Messwerten kann man aber nicht davon ausgehen, dass diese mittelwertfrei sind.

Die Aufgabe dieser Bachelorarbeit ist es, ein Programm zu entwickeln, mit dem Resonanzfrequenzen gefunden werden können. Diese kann man auch finden, wenn man nur das Beschleunigungssignal betrachtet.
Das Integrieren um ein Wegsignal zu erhalten, ist also nicht direkt Teil der Aufgabe. Aus diesem Grund und aus Mangel an Zeit, noch tiefer in die Theorie einzusteigen, wird nun der letztgenannte Ansatz verwendet, auch wenn dieser keine exakten Ergebnisse liefert.
Da dieses Verfahren nicht genau ist, wird in der Auswertung nicht ausschließlich mit den Wegsignalen gearbeitet, sondern vorrangig mit den Beschleunigungssignalen.

Der Einfachheit halber wird der Ansatz in *DIAdem* (siehe Abschnitt 8.1) durchgeführt. Dort sind alle Messwerte nach der Messung vorhanden. Somit muss die Integration nicht kontinuierlich während der Messung in *LabVIEW* durchgeführt werden, sondern kann im Anschluss mit der Auswertung der Daten verbunden werden.

10. Messung an einem Hubschrauberwindkanalmodell

An dem fertig aufgebauten Hubschrauberwindkanalmodell sollen nun die entwickelten Programme getestet werden (siehe Abbildungen 10.1 bis 10.3).

In Abbildung 10.1 ist der Aufbau dargestellt. Einerseits ist das Hubschrauberwindkanalmodell zu sehen, andererseits kann man im Vordergrund das Messsystem erkennen.

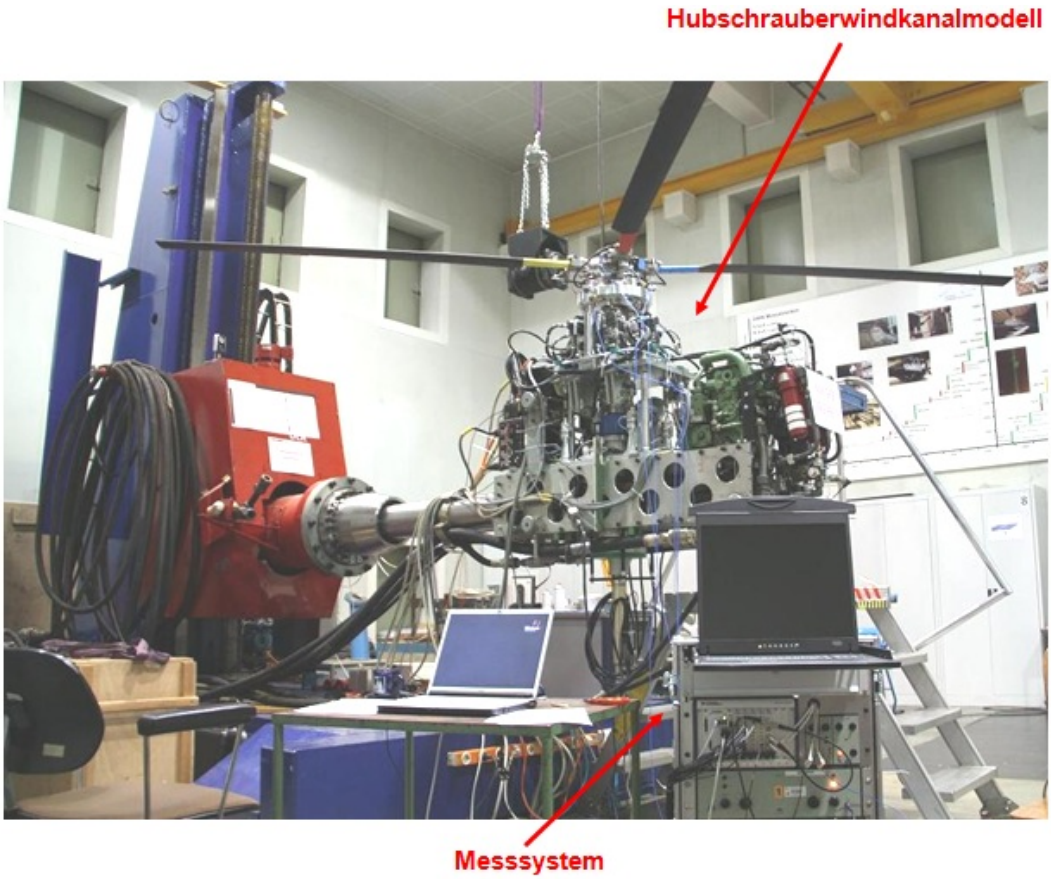


Abbildung 10.1.: Hubschrauberwindkanalmodell von der Seite mit dem Messsystem, Autor: René Pfeifer

Abbildung 10.2 zeigt, wo die Beschleunigungssensoren an dem Modell angebracht sind. Es ist zu erkennen, dass die Sensoren an verschiedenen Positionen angebracht sind, um somit von möglichst vielen Bauteilen Messdaten zu erhalten.
Beim Anbringen sollte darauf geachtet werden, dass die einzelnen Sensoren gleich ausgerichtet sind (also z.B. die Z-Richtung jedes Sensors als Vertikale), um die Sichtung der Auswertung zu erleichtern.
Weiterhin ist in dem Bild zu erkennen, dass von jedem Sensor eine Leitung abgeht. Diese verbindet die Sensoren mit den Eingangskarten des *PXI*-Messsystems.

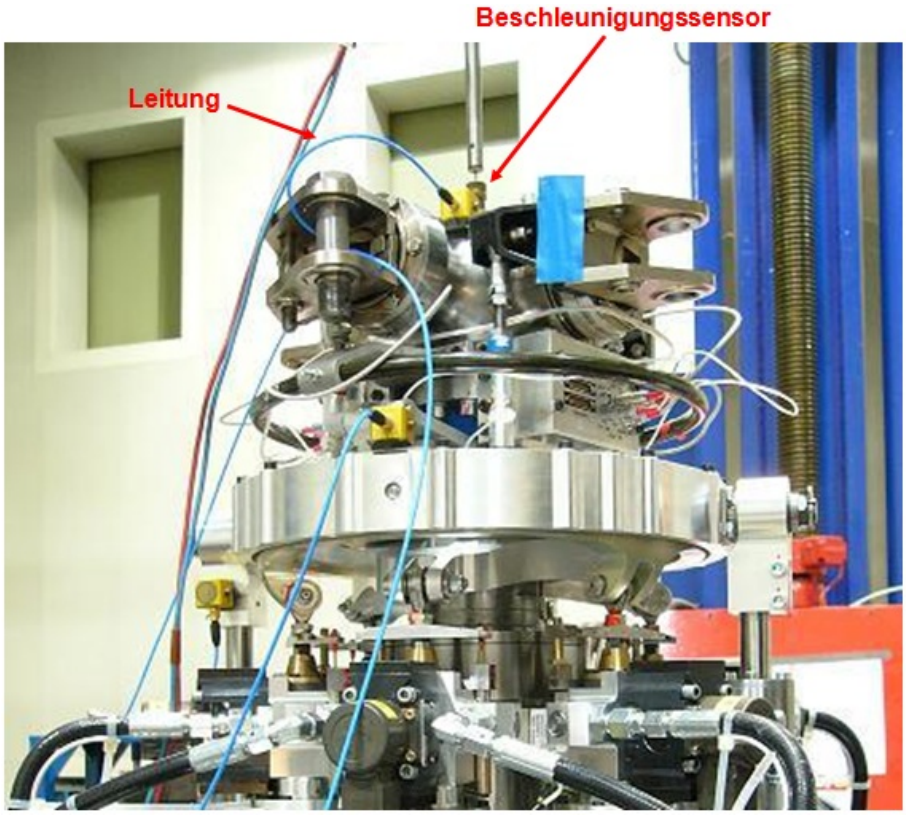


Abbildung 10.2.: Rotoraufbau mit Taumelscheibe, Autor: René Pfeifer

Schließlich ist in Abbildung 10.3 zu sehen, wie der Shaker mit dem Modell verbunden ist, um dieses anzuregen.

Abbildung 10.3.: Hubschrauberwindkanalmodell mit Shaker, Autor: René Pfeifer

Während den Messungen, die in diesem Kapitel zu sehen sind, ist ein Problem aufgetreten. Dieses wird in Kapitel 11 genau erläutert. Obwohl dieses Problem bestand, konnten die ersten wichtigen Informationen aus den Messungen gewonnen werden.

11. Fazit und Ausblick

Das vorgegebene *PXI*-Messsystem eignet sich hervorragend für messtechnische Aufgaben - in unserem Fall die Datenerfassung von sechs Beschleunigungssensoren und einem Kraftsensor. Zudem ist gleichzeitig mit dem Einlesen eine Ausgabe möglich. Dies hat den großen Vorteil, dass die gesamte Erfassung und die Ausgabe über ein einziges Messsystem realisiert werden können. Hinzu kommt, dass die gesamte Steuerung über eine Software, die in unserem Fall in *LabVIEW* entwickelt wurde, bedient werden kann. So wird dem Bediener alles auf einem Bildschirm angezeigt, für die Bedienung ist ein übersichtliches Programm vorhanden und er muss sich nur in eine Applikation einarbeiten. Da das Einlesen und Ausgeben gleichzeitig möglich ist, braucht man nicht jeweils mindestens ein Gerät, sondern hat alles in einer kompakten Bauform. Des Weiteren hat das *PXI*-Messsystem die Größe von einem 19" Rack und kann somit in konventionellen Rackschränken eingebaut werden. In unserem Fall ist das Messsystem in einem mobilen Container untergebracht. Dies steigert die Flexibilität des Gesamtsystems.

Trotz aller Vorteile gab es in der Entwicklungsphase ein großes Problem. Dieses soll im Folgenden erläutert werden.
Es war ursprünglich geplant, dass bei einem Frequenzsweep jede Frequenz eine Einschwingzeit erhält. Dieses wurde so gelöst, dass die Aufnahme der Messdaten erst startete, wenn die Einschwingzeit abgelaufen ist. Dabei musste darauf geachtet werden, dass die vorher vom Benutzer eingegebene Anzahl an Perioden eingehalten wird. Dafür wurde nach Formel (11.1) (siehe Fremdworterklärung) die Anzahl an Samples[3] berechnet, die für eine Periode der jeweiligen Frequenz nötig ist. Diese wurde anschließend mit der Periodenanzahl multipliziert. Somit erhielt man eine Gesamtanzahl an Samples[3], die für die aktuelle Frequenz aufgenommen werden musste. War diese Aufnahme der Samples[3] beendet, stoppte die Aufnahme, die Frequenz wurde erhöht und derselbe Ablauf, wie eben beschrieben, begann von Neuem für die nächste Frequenz.
Dieses beschriebene Vorgehen funktioniert auch unter bestimmten Einstellungen, nämlich bei der Wahl eines großen Ausgangspuffers. Der Nachteil an dem großen Ausgangspuffers ist, dass dadurch die Ausgabe des Signals um Sekunden verzögert wird. Wurde der Puffer klein genug gewählt, um eine zeitnahe Ausgabe des Signals zu gewährleisten, traten Sprünge in dem Ausgabesignal auf. Das liegt im Fundament der Datenerfassung, da die Daten mit endlicher Anzahl erfasst wurden. Das bedeutet, dass während der Ausgabe die Verbindung zu den Messkarten initialisiert wird und nach der Einschwingzeit die benötigten Samples[3] eingelesen und anschließend wieder beendet werden. Dieser Vorgang wiederholte sich bei jeder Frequenz. Während dieses Einlesens ist die Applikation damit beschäftigt und kann keine weiteren Programmteile ausführen. Da die Ausgabe im Regenerationsmodus betrieben wurde (siehe dazu auch Abschnitt 7.2.22), war dies kein Problem, da das Signal, welches im Puffer lag weiterhin ausgegeben wurde. Jedoch schaffte es die Applikation aus

den oben genannten Gründen nicht immer, eine komplette Periode des Signals in den Puffer zu schreiben. Dadurch wurde ein Teil der alten Daten mit den Neuen ausgegeben, wodurch Sprünge im Signal hervorgerufen wurden.
Das Problem wurde nun so umgangen, dass die gesamten Programmteile der Signalerfassung und -ausgabe geändert wurden, sodass nicht mehr eine Aufnahme einer bestimmten Anzahl an Samples[3] pro Frequenz gestartet wird, sondern eine komplett kontinuierliche Messung über den gesamten gesweepten Bereich durchgeführt wird. Der Nachteil hieran ist, dass es nicht mehr möglich ist, für jede Frequenz eine Einschwingzeit neben der eigentlichen Messzeit vorzusehen. Das bedeutet, dass nun in der Messzeit auch der Einschwingvorgang enthalten ist.
Da das *DIAdem*-Programm und das *LabVIEW*-Programm parallel entwickelt wurden, waren schon viele Teile des *DIAdem*-Programms vorhanden, die nach Behebung des Problems nicht mehr verwendet werden konnten. So war es zunächst geplant, zwei verschiedene Arten von FFTs durchzuführen. Einmal die über den gesamten Messwertkanal, die auch jetzt noch vorhanden ist. Zum anderen sollte jeweils eine FFT pro gesweepter Frequenz berechnet werden. Diese sollte in der Auswertung mit 16 Oberwellen dargestellt werden. So hatte man mit der FFT über den gesamten Messbereich eine Übersicht und mit der FFT pro Frequenz eine detailliertere Auswertung. Um diese FFT jedoch berechnen zu können, war es zunächst nötig, jeden Messwertkanal so aufzuteilen, dass hinterher Messwertkanäle entstehen, die nur Messwerte zu einer Frequenz enthalten. Hierfür wurde im *DIAdem*-Programm die jeweilige Anzahl an Samples[3] berechnet und die Kanäle entsprechend aufgeteilt. Genau diese Berechnung ist nun nicht mehr möglich. Vorher war es so, dass in *LabVIEW* die aufzunehmende Anzahl an Samples[3] berechnet und dann auch aufgenommen wurde. Da die Aufnahme nun kontinuierlich stattfindet, wird pro Frequenz immer eine Zeit berechnet und nach Ablauf dieser Zeit wird die Frequenz erhöht. Da diese Zeit aber niemals exakt ist, schon allein auf Grund der Laufzeiten des Programms und weil *LabVIEW* keine kleinere Zeiteinheit als Sekunden einzuhalten schafft, ist auch die Anzahl an Samples[3] niemals genau gleich. Da die Berechnung der detaillierten FFT aber doch einen großen Teil in der Entwicklung des Programms eingenommen hat, sind einige Programmauszüge dessen im Anhang zu sehen (siehe Anhang H).
Abschließend ist zu sagen, dass die endliche Messung zwar durch die kontinuierliche Messung ersetzt wurde, diese aber trotzdem zum gewünschten Ergebnis führt. Der Grund für diese Änderung ist die umfangreiche Messdatenerfassung in *LabVIEW* in Kombination mit der Hardware, welche nicht vermieden werden kann.

Auch wenn das eben beschriebene Problem aufgetreten ist und nicht behoben werden konnte, kann man schlussendlich sagen, dass das Gesamtpaket aus *PXI*-System, *LabVIEW* und *DIAdem* hervorragend aufeinander abgestimmt ist und daher für solche Anwendungen wie in dieser Bachelorarbeit bestens geeignet ist.

Für die Zukunft sind noch einige Erweiterungen denkbar, auf welche im Folgenden hingewiesen wird.
Es sind momentan nur sechs Beschleunigungssensoren vorhanden. Da die *PXI*-Messkarten

aber mehr Plätze bieten, sei an dieser Stelle als erstes erwähnt, dass die Versuche mit bis zu zehn Beschleunigungssensoren durchgeführt werden können (siehe hierzu auch Abbildung 7.21).
Des Weiteren besteht die Möglichkeit, statt einem Shaker zwei zu verwenden. Hierfür sind schon ein zweiter Kraftsensor inklusive Ladungsverstärker, ein zweiter Leistungsverstärker sowie die Anschlüsse an den Messkarten vorhanden. Für diesen Fall müssten allerdings noch die Programme in *LabVIEW* und *DIAdem* erweitert werden.
Eine weitere Aufgabe für spätere Arbeiten wäre die genauere Ausarbeitung der Umwandlung von den Beschleunigungssignalen in Wegsignale, da dieses in dieser Arbeit noch nicht abschließend geklärt ist (siehe Kapitel 9).
Außerdem ist es momentan so, dass mit einer konstanten Spannung angeregt wird. Vorteilhaft wäre es jedoch, mit einer konstanten Kraft anzuregen. Es wäre denkbar, einen Regler zu entwerfen, der die eingeleitete Kraft konstant hält. Der Vorteil dabei wäre, dass zum Beispiel in der Resonanz Kraft vom Regler weggenommen werden würde. Somit würde das System nicht aufschwingen und es würden eventuelle Schäden an der Mechanik verhindert werden.
Denkbar wäre auch, aus der Script-Datei Teile zu extrahieren und mehrere Programme daraus zu machen. So könnte der Anwender in dem *LabVIEW*-Programm gefragt werden, was in der Auswertung zu sehen sein soll (z.B. nur die FRF oder nur die FFT). Abhängig von dieser Auswahl müsste dann das jeweils passende Script aufgerufen werden.

Fremdworterklärung

[1] Task

„Ein Task ist ein Messprojekt, für das bestimmte virtuelle Kanäle sowie Takt-, Trigger- und andere Einstellungen festgelegt werden. In einem Task sind also alle Parameter einer Messung oder Signalerzeugung zusammengefasst. Alle Kanäle in einem Task müssen demselben Zweck dienen, z. B. Erfassung analoger Signale oder zählergestützte Signalausgabe. Ein Task kann jedoch Kanäle für verschiedene Arten von Messungen enthalten, wie z. B. einen Kanal für die analoge Erfassung einer Temperatur und einen Kanal für die analoge Erfassung einer Spannung.“ (aus [8])

[2] Samplerate

„Einer der wichtigsten Parameter eines Systems zur Erfassung oder Ausgabe analoger Signale ist die Rate, mit der das Messgerät das ankommende Signal abtastet oder das Ausgangssignal erzeugt. Die Sample-Rate (beim traditionellen NI-DAQ-Treiber auch manchmal „Scan-Rate“ genannt) gibt an, wie schnell ein Gerät auf jedem Kanal ein Sample erfasst oder ausgibt. Je höher die Sample-Rate eines Eingangssignals ist, desto mehr Teilstücke einer Signalperiode werden in einem bestimmten Zeitraum entnommen. Die Signalform ist auf diese Weise viel besser darstellbar als bei einer langsamen Abtastung. Wenn ein Signal mit einer Frequenz von 1 Hz so abgetastet wird, dass pro Periode 1000 Proben (Samples) des Signals entnommen werden (1000 $\frac{S}{s}$), kann die Signalform wesentlich besser nachgebildet werden als bei nur 10 Proben pro Periode (10 $\frac{S}{s}$). Wenn zu langsam abgetastet wird, ist das analoge Signal schlechter reproduzierbar. Bei der so genannten Unterabtastung erscheint das Signal mit einer anderen Frequenz als es eigentlich hat. Diese Fehldarstellung eines Signals wird Aliasing oder auch Treppeneffekt genannt.“ (aus [8])

[3] Samples

Anzahl der Messpunkte, die generiert oder erfasst werden sollen (vgl. (11.1))

$$Samplerate\left[\frac{S}{s}\right] = \frac{Samples\ [S]}{Zeit\ [s]} = Samples\ [S] \cdot Frequenz\ [Hz] \tag{11.1}$$

[4] Cluster

Ein „Cluster“ kann mehrere unterschiedliche Datentypen akzeptieren. Ein Beispiel dafür ist der „Error-Cluster“ von *LabVIEW* (siehe Abbildung 11.1).
Die linke grüne Leitung ist der „Error-Cluster“. Es ist zu sehen, dass der Baustein „Unbundle by name“ den Cluster aufteilt. Er beinhaltet einen booleschen sowie numerischen Wert und einen Text. Ein Cluster kann also aus einer großen Anzahl verschiedener Datentypen bestehen.

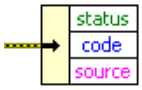


Abbildung 11.1.: Beispiel zu einem Cluster, Autor: René Pfeifer

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Advanced Test Equipment Rentals: *Bruel & Kjaer 2651 Charge Amplifier*, URL http://www.atecorp.com/equipment/bruelkjaer/2651_2805.asp - Datum des Aufrufs: 27. Juli 2011

[2] Burgemeister, Jan : *Ladungsverstärker, Prinzipschaltung*, URL http://upload.wikimedia.org/wikipedia/de/c/cb/Ladungsverstaerker.gif - Datum des Aufrufs: 14. November 2011

[3] Deichmann, N.: *Einführung in die Zeitreihenanalyse Teil 2.*, Zürich : Technische Hochschule, Institut für Geophysik, URL http://loadbalancer.ethz.ch/staff/jclinton/GDP07/skript2.pdf - Datum des Aufrufs: 05. September 2011

[4] *DIAdem* Forum : *FFT-Praxis in NI DIAdem*, URL http://www.diadem-forum.de/was-du-schon-immer-uber-fft-wissen-wolltest-t-215.html - Datum des Aufrufs: 30. August 2011

[5] *DIAdem* Hilfe der Software

[6] Van der Wall, B. G.: *Drehflügeltechnik - Rotordynamik*, Braunschweig: Institut für Luft- und Raumfahrtsysteme, Technische Universität Braunschweig - befindet sich auf der beigefügten CD -

[7] Istanbul Technical University : *Product Data Force Transducers - Types 8200 and 8201*, URL http://www.titak.itu.edu.tr/Force_transducers/BK_8200.pdf - Datum des Aufrufs: 25. Juli 2011

[8] *LabVIEW* Hilfe der Software

[9] National Instruments : *2.53 GHz Dual-Core Embedded Controller for PXI NI PXI-8108*, URL http://sine.ni.com/ds/app/doc/p/id/ds-273/lang/de - Datum des Aufrufs: 27. Juli 2011

[10] National Instruments : *24-Bit, 204.8 kS/s Dynamic Signal Acquisition and Generation NI 4461, NI 4462*, URL http://sine.ni.com/ds/app/doc/p/id/ds-337/lang/de - Datum des Aufrufs: 27. Juli 2011

[11] National Instruments : *24-Bit, 102.4 kS/s, 8- and 4-Channel Dynamic Signal Acquisition NI 4472 Series, NI PCI-4474*, URL http://www.ni.com/pdf/products/us/3sv414-416.pdf - Datum des Aufrufs: 27. Juli 2011

[12] National Instruments : *DIAdem Einführung in die interaktive Datenauswertung*, URL http://www.ni.com/pdf/manuals/370931d_0113.pdf - Datum des Aufrufs: 28. Juli 2011

[13] National Instruments : *General-Purpose 8-Slot Chassis for PXI NI PXI-1042, NI PXI-1042Q*, URL http://sine.ni.com/ds/app/doc/p/id/ds-256/lang/de - Datum des Aufrufs: 27. Juli 2011

[14] National Instruments : *LabVIEW Run-Time Engine 2010 - (32-bit Standard RTE) - Windows 7/7 64 bit/Server 2003 R2 (32-bit)/XP/Vista x86/Vista x64/Server 2008 R2 (64-bit)*, URL http://joule.ni.com/nidu/cds/view/p/id/2087/lang/de - Datum des Aufrufs: 15. Dezember 2011

[15] National Instruments : *NI PXI – Ihre Eintrittskarte in die Welt der modularen Mess- und Prüftechnik*, URL http://webcasts.de.ni.com/?elqPURLPage=118 - Datum des Aufrufs: 01. Ausgust 2011

[16] National Instruments : *NI-TDMS-Dateiformat*, URL http://zone.ni.com/devzone/cda/tut/p/id/5557 - Datum des Aufrufs: 08. August 2011

[17] National Instruments : *NI-TDMS-Dateiformat*, URL http://zone.ni.com/devzone/jsp/largeImage.jsp?imagename=/cms/images/devzone/tut/TDMS_with_Properties.png&language=de - Datum des Aufrufs: 14. Oktober 2011

[18] National Instruments : *PXI-Hard- und -Softwarearchitektur*, URL http://sine.ni.com/np/app/main/p/ap/global/lang/de/pg/1/sn/n24:PXI-FSLASH-CompactPCI/fmid/3/ - Datum des Aufrufs: 19. Juli 2011

[19] Mjs-Electronics : *Audio & Video*, URL http://www.mjs-electronics.se/images/Bruel_kj/2651.jpg - Datum des Aufrufs: 13. Oktober 2011

[20] PCB PIEZOTRONICS : *Model 356B18 ICP Accelerometer*, URL http://www.pcb.com/contentstore/docs/PCB_Corporate/Vibration/products/Manuals/356B18.pdf - Datum des Aufrufs: 26. Juli 2011

[21] Vienna University of Technology : *IIR-Filter*, URL http://ti.tuwien.ac.at/rts/teaching/courses/dspv/files/IIRFilter.pdf - Datum des Aufrufs: 24. Oktober 2011

[22] Weber, Manfred : *Metra Mess- und Frequenztechnik in Radebeul e.K.*, URL http://www.mmf.de/messtechnik.htm - Datum des Aufrufs: 14. November 2011

[23] Wikipedia : *Filter mit endlicher Impulsantwort*, URL `http://de.wikipedia.org/wiki/Filter_mit_endlicher_Impulsantwort` - Datum des Aufrufs: 24. Oktober 2011

[24] Wikipedia : *Digital Visual Interface*, URL `http://de.wikipedia.org/wiki/DVI` - Datum des Aufrufs: 20. Juli 2011

[25] Wikipedia : *Integrated Electronics Piezo-Electric*, URL `http://de.wikipedia.org/wiki/Integrated_Electronics_Piezo-Electric` - Datum des Aufrufs: 22. Juli 2011

[26] Wikipedia : *Piezoelektrizität*, URL `http://de.wikipedia.org/wiki/Piezoelektrizit%C3%A4t` - Datum des Aufrufs: 10. November 2011

[27] Wikipedia : *Ladungsverstärker*, URL `http://de.wikipedia.org/wiki/Ladungsverst%C3%A4rker` - Datum des Aufrufs: 28. Juli 2011

[28] Wikipedia : *VGA-Anschluss*, URL `http://de.wikipedia.org/wiki/VGA-Anschluss` - Datum des Aufrufs: 20. Juli 2011

[29] Wikipedia : *Federpendel*, URL `http://de.wikipedia.org/wiki/Federpendel` - Datum des Aufrufs: 18. August 2011

Inhalt der CD

- Datenblätter
- *DIAdem*-Programm
- Dokumentation der Bachelorarbeit
- Informationen zum Bedienen der Programme
- Informationen zum Einstellen der Empfindlichkeiten von Ladungsverstärker und Kraftsensor
- *LabVIEW*-Programm als .exe-Datei
- *LabVIEW*-Programm als .vi-Datei
- Vorlesungsskript: Drehflügeltechnik - Rotordynamik

Anhang

A. Informationen

Informationen zum Bedienen der Programme

Allgemein ist zu beachten, dass die Ordnerstruktur des Ordners „NI Shaker-Software“ nicht verändert werden darf. Wird einer der Unterordner dieses Ordners gelöscht oder umbenannt, werden sowohl im *LabVIEW*- als auch im *DIAdem*-Programm Fehler auftreten, da diese Ordner dann nicht mehr gefunden werden können.
Des Weiteren sollte auf dem Zielrechner der *Windows Media Player* installiert sein, um das Video im *LabVIEW*-Programm abspielen zu können.

LabVIEW:

Kurze Übersicht

Menüseite 1: Namensvergabe:

Auf dieser Seite können sämtliche Namen, Bezeichnungen und Beschreibungen festgelegt werden, welche später zu einer besseren und übersichtlicheren Auswertung beitragen. Diese können auch aus einer .txt-Datei geladen oder in eine .txt-Datei gespeichert werden.

Zudem gibt es unter dem Button „Anschlussübersicht“ eine Übersicht, wie die einzelnen Sensoren usw. angeschlossen werden müssen und unter dem Button „Tutorial“ ein kleines Video, welches den Einstieg in die Bedienung des Programms erleichtern soll.

Wenn diese Seite korrekt und vollständig ausgefüllt ist, gelangt man mit dem Button „Weiter“ zur nächsten Seite.

Menüseite 2: Messkarteneinstellungen:

Im Fenster „Anzahl der Sensoren“ muss die Anzahl der verwendeten Sensoren eingetragen werden. Danach müssen die Einstellungen der einzelnen Karten in den Slots 2, 3, 4 und 5 vorgenommen werden.

Einzustellen sind:

Maximal-/ Minimalwert:
Hiermit wird die maximale bzw. minimale Spannung für die Messkarte festgelegt.

Wenn diese Seite korrekt und vollständig ausgefüllt ist, gelangt man mit dem Button „Weiter“ zur nächsten Seite bzw. um sich Einstellungen der vorherigen Seite noch einmal anzuschauen oder zu korrigieren benutzt man den Button „Zurück“.

Menüseite 3: Anregungseinstellungen:

Auf dieser Seite müssen die „Samplerate“, „Rampenanstiegszeit“ (falls unerwünscht auf null setzen) und „Charge Amplifier“-Einstellungen (die Einstellung die an dem Charge Amplifier eingestellt ist) eingetragen werden. Danach ist eine Anregungsform zu wählen. Hier steht „Wellenform“ und „Rauschen“ zur Auswahl. Dabei stehen unter Wellenform die Signalformen Sinus („Sine Wave“), Sägezahn („Sawtooth Wave“), Rechteck („Square Wave“) und Dreieck („Triangle Wave“) zur Auswahl. Zum Schluss müssen bei der gewählten Anregungsform noch die einzelnen Parameter wie Offset, Amplitude etc. eingestellt werden.

Unten auf der Seite ist eine Fortschrittsanzeige zu finden, welche den Fortschritt der aktuellen Messung anzeigt.

Wenn diese Seite korrekt und vollständig ausgefüllt ist, startet man mit dem Button „Start“ die Messung. Um sich Einstellungen der vorherigen Seiten noch einmal anzuschauen oder zu korrigieren benutzt man den Button „Zurück“.

Anzeige vom Ausgangssignal, Bescheunigungs- und Kraftsensoren:

Die Anzeige rechts neben den Menüseiten zeigt den Verlauf des Anregungssignals und die Aufnahmen der Beschleunigungssensoren sowie des Kraftsensors an.

Über den beiden Fenstern ist eine Signallampe sowie ein „Stop“- Button. Die Lampe dient lediglich als Anzeige, ob die gesamte Messung abgeschlossen ist. Der Button dient dem vorzeitigen Stoppen der aktuellen Messreihe.

Wichtige Zusatzinformationen

- Für eine sinnvolle Auswertung sollten mindestens drei bis fünf Signalperioden aufgenommen werden.
- Für eine Messung ohne Anregung ist die Signalart „Rauschen“ zu wählen und das RMS-Level auf null zu setzen.
- Sinnvollerweise sollte *DIAdem* vor Beginn der Messung gestartet werden, um unnötige Ladezeiten sowie das automatische Schließen von *DIAdem* nach Beenden von *LabVIEW* zu vermeiden. Sollte vergessen werden, *DIAdem* zu öffnen, übernimmt *LabVIEW* das Starten des Programms. Jedoch wird *DIAdem* mit dem Beenden von *LabVIEW* auch wieder geschlossen.

- Auf Menüseite 1 ist die Eingabe von Informationen über den Ort des Sensors auf jeweils 120 Zeichen begrenzt.
- Die Einteilung des Drehknopfs „Amplifier Gain“ des Ladungsverstärkers ist logarithmisch. Um ein Verhältnis von Eingangssignal zu Ausgangssignal von 1:1 zu bekommen, muss der Knopf auf 7 gestellt werden.
- Zu beachten ist, dass der Verstärker das Signal, welches von der Messkarte „PXI-4461“ ausgegeben wird, invertiert ausgibt.

DIAdem:

Es werden von einer Messung immer zwei .tdms-Dateien gespeichert. Zum einen gibt es die Datei, die die reinen Messdaten enthält. Diese ist in dem Ordner „Messdaten“ zu finden. Zum anderen werden alle zu einer Messung durchgeführten Berechnungen in einer Datei gespeichert. Diese ist in dem Ordner „Berechnungen“ des Ordners „Auswertung“ zu finden. So können sowohl die Rohdaten als auch die Berechnungen später noch einmal nachvollzogen werden. Um die Inhalte dieser Dateien begutachten zu können, wird das Modul „NAVIGATOR“ aufgerufen und dort die jeweilige Datei geöffnet. Anschließend können alle Daten im Modul „VIEW“ gesichtet werden.

Im Modul „REPORT“ ist es möglich, die dargestellten Daten zu verändern, wie zum Beispiel die x- oder y-Achse neu zu skalieren oder Daten aus einem Koordinatensystem zu entfernen. Wichtig zu wissen ist hierbei, dass es zwar möglich ist, einen Report zu speichern, jedoch werden damit nicht die veränderten Daten gespeichert. Ein Report ist lediglich eine Art „Formatvorlage“. Es werden die Einstellungen von Koordinatensystem gespeichert, nicht jedoch die Daten in ihnen. Ruft man einen Report auf, enthält dieser also generell keinerlei Daten.

B. Datenblatt Beschleunigungssensor 356B18

Model 356B18

ICP® Accelerometer

Installation and Operating Manual

For assistance with the operation of this product,
contact the Vibration Division of PCB Piezotronics, Inc.

Division toll-free: 888-684-0013
24-hour SensorLine: 716-684-0001
Fax: 716-685-3886
E-mail: vibration@pcb.com

PCB PIEZOTRONICS
VIBRATION DIVISION

PCB PIEZOTRONICS
A PCB GROUP COMPANY

Warranty, Service, Repair, and Return Policies and Instructions

The information contained in this document supersedes all similar information that may be found elsewhere in this manual.

Total Customer Satisfaction – PCB Piezotronics guarantees Total Customer Satisfaction. If, at any time, for any reason, you are not completely satisfied with any PCB product, PCB will repair, replace, or exchange it at no charge. You may also choose to have your purchase price refunded in lieu of the repair, replacement, or exchange of the product.

Service – Due to the sophisticated nature of the sensors and associated instrumentation provided by PCB Piezotronics, user servicing or repair is not recommended and, if attempted, may void the factory warranty. Routine maintenance, such as the cleaning of electrical connectors, housings, and mounting surfaces with solutions and techniques that will not harm the physical material of construction, is acceptable. Caution should be observed to insure that liquids are not permitted to migrate into devices that are not hermetically sealed. Such devices should only be wiped with a dampened cloth and never submerged or have liquids poured upon them.

Repair – In the event that equipment becomes damaged or ceases to operate, arrangements should be made to return the equipment to PCB Piezotronics for repair. User servicing or repair is not recommended and, if attempted, may void the factory warranty.

Calibration – Routine calibration of sensors and associated instrumentation is recommended as this helps build confidence in measurement accuracy and acquired data. Equipment calibration cycles are typically established by the users own quality regimen. When in doubt about a calibration cycle, a good "rule of thumb" is to recalibrate on an annual basis. It is also good practice to recalibrate after exposure to any severe temperature extreme, shock, load, or other environmental influence, or prior to any critical test.

PCB Piezotronics maintains an ISO-9001 certified metrology laboratory and offers calibration services, which are accredited by A2LA to ISO/IEC 17025, with full traceablility to N.I.S.T. In addition to the normally supplied calibration, special testing is also available, such as: sensitivity at elevated or cryogenic temperatures, phase response, extended high or low frequency response, extended range, leak testing, hydrostatic pressure testing, and others. For information on standard recalibration services or special testing, contact your local PCB Piezotronics distributor, sales representative, or factory customer service representative.

Returning Equipment – *Following these procedures will insure that your returned materials are handled in the most expedient manner.* Before returning any equipment to PCB Piezotronics, contact your local distributor, sales representative, or factory customer service representative to obtain a Return

Materials Authorization (RMA) Number. This RMA number should be clearly marked on the outside of all package(s) and on the packing list(s) accompanying the shipment. A detailed account of the nature of the problem(s) being experienced with the equipment should also be included inside the package(s) containing any returned materials.

A Purchase Order, included with the returned materials, will expedite the turn-around of serviced equipment. It is recommended to include authorization on the Purchase Order for PCB to proceed with any repairs, as long as they do not exceed 50% of the replacement cost of the returned item(s). PCB will provide a price quotation or replacement recommendation for any item whose repair costs would exceed 50% of replacement cost, or any item that is not economically feasible to repair. For routine calibration services, the Purchase Order should include authorization to proceed and return at current pricing, which can be obtained from a factory customer service representative.

Warranty – All equipment and repair services provided by PCB Piezotronics, Inc. are covered by a limited warranty against defective material and workmanship for a period of one year from date of original purchase. Contact PCB for a complete statement of our warranty. Expendable items, such as batteries and mounting hardware, are not covered by warranty. Mechanical damage to equipment due to improper use is not covered by warranty. Electronic circuitry failure caused by the introduction of unregulated or improper excitation power or electrostatic discharge is not covered by warranty.

Contact Information – International customers should direct all inquiries to their local distributor or sales office. A complete list of distributors and offices can be found at www.pcb.com. Customers within the United States may contact their local sales representative or a factory customer service representative. A complete list of sales representatives can be found at www.pcb.com. Toll-free telephone numbers for a factory customer service representative, in the division responsible for this product, can be found on the title page at the front of this manual. Our ship to address and general contact numbers are:

PCB Piezotronics, Inc.
3425 Walden Ave.
Depew, NY 14043 USA
Toll-free: (800) 828-8840
24-hour SensorLine℠: (716) 684-0001
Website: www.pcb.com
E-mail: info@pcb.com

DOCUMENT NUMBER: 21354
DOCUMENT REVISION: B
ECN: 17900

General

OPERATING GUIDE

for use with

PIEZOELECTRIC ICP® ACCELEROMETERS

SPECIFICATION SHEET, INSTALLATION DRAWING
AND CALIBRATION INFORMATION ENCLOSED

PCB ASSUMES NO RESPONSIBILITY FOR DAMAGE CAUSED TO THIS PRODUCT AS A RESULT OF PROCEDURES THAT ARE INCONSISTENT WITH THIS OPERATING GUIDE.

1.0 INTRODUCTION

Congratulations on the purchase of a quality, ICP® acceleration sensor. In order to ensure the highest level of performance for this product, it is imperative that you properly familiarize yourself with the correct mounting and installation techniques before attempting to operate this device. If, after reading this manual, you have any additional questions concerning this sensor or its application, feel free to call a factory Application Engineer at 716-684-0001 or your nearest PCB sales representative.

2.0 ICP® ACCELEROMETERS

Powered by simple, inexpensive, constant-current signal conditioners, these sensors are easy to operate and interface with signal analysis, data acquisition and recording instruments. The following features further characterize ICP® sensors:

- Fixed voltage sensitivity, regardless of cable type or length.
- Low-impedance output signal, which can be transmitted over long cables in harsh environments with virtually no loss in signal quality.
- Two-wire operation with low cost coaxial cable, two-conductor ribbon wire or twisted-pair cabling.
- Low-noise, voltage-output signal compatible with standard readout, signal analysis, recording, and data acquisition equipment.
- Low cost per-channel - ICP® accelerometers require only an inexpensive, constant-current signal conditioner to operate.
- Intrinsic self-test feature – monitoring the sensor's output bias voltage provides an indication of proper operation, faulty condition, and bad cables.

In the rear of this manual you will find a **Specification Sheet**, which provides the complete performance characteristics of your particular sensor.

3.0 OPTIONAL FEATURES

Many sensors are supplied with standard, optional features. When listed before the model number, the following prefix letters indicate that the sensor is manufactured or supplied with a particular optional feature: **"A"** option: adhesive mount; **"HT"** option: extended high temperature range; **"J"** option: electrically ground isolated; **"M"** option: metric mounting thread; **"Q"** option: extended discharge time constant; **"T"** option: built-in transducer electronic data sheet (TEDS); and **"W"** option: attached, water-resistant cabling. Other prefix letters, such as **"K"**, **"KR"**, **"GK"**, **"GKR"**, **"KL"**, and **"GKL"**, indicate that the sensor is ordered in kit form, including interconnect cabling and signal conditioner. If you have any questions or concerns regarding optional features, consult the Vibration Division's product catalog or contact a PCB factory representative.

4.0 INSTALLATION OVERVIEW

When choosing a mounting method, consider closely both the advantages and disadvantages of each technique. Characteristics like location, ruggedness, amplitude range, accessibility, temperature, and portability are extremely critical. However, the most important and often overlooked consideration is the effect the mounting technique has on the high-frequency performance of the accelerometer.

® ICP is a registered trademark of PCB Group, Inc., which uniquely identifies PCB sensors that incorporate built-in microelectronics.

Shown in figure 1 are six possible mounting techniques and their effects on the performance of a typical piezoelectric accelerometer. (Note that not all of the mounting methods may apply to your particular sensor). The mounting configurations and corresponding graph demonstrate how the high-frequency response of the accelerometer may be compromised as mass is added to the system and/or the mounting stiffness is reduced.

NOTE: *The low-frequency response is unaffected by the mounting technique. This roll-off behavior is typically fixed by the sensor's built-in electronics. However, when operating AC-coupled signal conditioners with readout devices having an input impedance of less than one megohm, the low frequency range may be affected. If necessary, contact a factory representative for further assistance.*

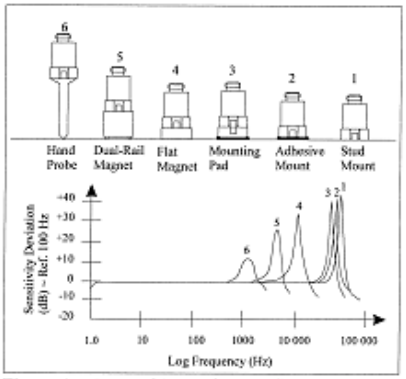


Figure 1. Assorted Mounting Configurations and Their Effects on High Frequency

4.1 STUD MOUNT

This mounting technique requires smooth, flat contact surfaces for proper operation and is recommended for permanent and/or secure installations. Stud mounting is also recommended when testing at high frequencies.

NOTE: *Do NOT attempt mounting on curved, rough, or uneven surfaces, as the potential for misalignment and limited contact surface may significantly reduce the sensor's upper operating frequency range.*

STEP 1: First, prepare a smooth, flat mounting surface, then drill and tap a mounting hole in the center of this area as shown in Figure 2 and in accordance with the enclosed **Installation Drawing**.

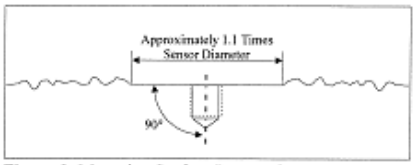


Figure 2. Mounting Surface Preparation

A precision-machined mounting surface with a minimum finish of 63 µin (0.00016 mm) is recommended. (If it is not possible to properly prepare the test structure mounting surface, consider adhesive mounting as a possible alternative). Inspect the area, checking that there are no burrs or other foreign particles interfering with the contact surface.

STEP 2: Wipe clean the mounting surface and spread on a light film of grease, oil, or similar coupling fluid prior to installation.

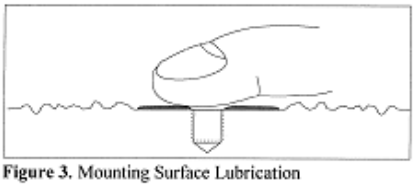


Figure 3. Mounting Surface Lubrication

Adding a coupling fluid improves vibration transmissibility by filling small voids in the mounting surface and increasing the mounting stiffness. For semi-permanent mounting, substitute epoxy or another type of adhesive.

STEP 3: Screw the mounting stud into the base of accelerometer and hand-tighten (this step is unnecessary for units having an integral mounting stud). Then, screw the sensor into the tapped hole that was prepared in the test object. Tighten the unit in place by applying, with a torque wrench, the recommended mounting torque, as listed on the enclosed **Installation Drawing**.

NOTE: *It is important to use a torque wrench during this step. Under-torquing the sensor may not adequately couple the device; over-torquing may result in stud failure.*

4.2 ADHESIVE MOUNT

Adhesive mounting is often used for temporary installation or when the test object surface cannot be adequately prepared for stud mounting. Adhesives like hot glue and wax perform well for temporary installations whereas two-part epoxies and quick-bonding gels (super glue) provide a more permanent installation. Two

techniques are used for adhesive mounting; they are via an adhesive mounting base (method 1 below) or direct adhesive mounting (method 2 below).

NOTE: *Adhesively mounted sensors often exhibit a reduction in high-frequency range. Generally, smooth surfaces and stiff adhesives provide the best high frequency response.*

METHOD 1 - Adhesive Mounting Base

This method involves attaching a base to the test structure, then securing the sensor to the base. This allows for easy removal of the accelerometer. Also, since many bases are manufactured of "hard-coated" aluminum, they provide electrical isolation to eliminate ground loops and reduce electrical interference that may propagate from the surface of the test object.

STEP 1: Prepare a smooth, flat mounting surface. A minimum surface finish of 63 µin (0.00016 mm) generally works best.

STEP 2: Stud-mount the sensor to the flat side of the appropriate adhesive mounting base according to the guidelines set forth in **STEPS 2** and **3** of the Stud Mount Procedure presented above.

STEP 3: Place a small portion of adhesive on the underside of the mounting base (the underside is discernable by the concentric grooves which are designed to accept the adhesive). Firmly press down on the assembly to displace any extra adhesive remaining under the base.

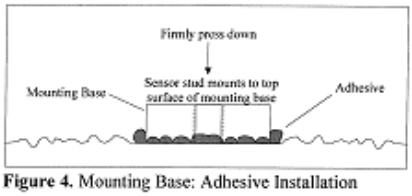


Figure 4. Mounting Base: Adhesive Installation

METHOD 2 - Direct Adhesive Mount

For restrictions of space or for convenience, most sensors can be adhesive-mounted directly to the test structure (an exception being units having integral mounting studs).

STEP 1: Prepare a smooth, flat mounting surface. A minimum surface finish of 63 µin (0.00016 mm) generally works best.

STEP 2: Place a small portion of adhesive on the underside of the sensor. Firmly press down on the top of the assembly to displace any adhesive. Be aware that excessive amounts of adhesive can make sensor removal difficult. Also, adhesive that may invade the tapped mounting hole in the base of the sensor will compromise future ability to stud mount the unit.

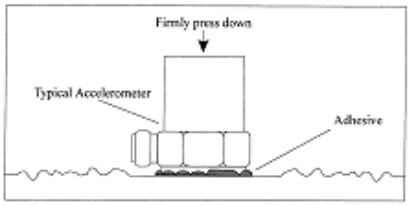


Figure 5. Direct Adhesive Mounting

4.2-1 ADHESIVE MOUNT REMOVAL (other than wax)

NOTE: *A debonder should always be used to avoid sensor damage.*

To avoid damaging the accelerometer, a debonding agent must be applied to the adhesive prior to sensor removal. With so many adhesives in use (everything from super glues, dental cement, epoxies, etc), there is no universal debonding agent available. The debonder for the Loctite 454 adhesive that PCB offers is Acetone. If you are using anything other than Loctite 454, you will have to check with the individual manufacturers for their debonding recommendations. The debonding agent must be allowed to penetrate the surface in order to properly react with the adhesive, so it is advisable to wait a few minutes before removing the sensor.

After the debonding agent has set, you can use an ordinary open-end wrench if the accelerometer has a hex base or square base, or the supplied removal tool for teardrop accelerometers. After attaching either, use a gentle shearing (or twisting) motion (by hand only) to remove the sensor from the test structure.

4.3 MAGNETIC MOUNT

Magnetic mounting provides a convenient means for making quick, portable measurements and is commonly used for machinery condition monitoring, predictive maintenance, spot checks, and vibration trending applications.

NOTE: *The correct magnet choice and an adequately prepared mounting surface are critical for obtaining reliable measurements, especially at high frequencies. Poor installations can cause as much as a 50% drop in the sensor frequency range.*

Not every magnet is suitable for all applications. For example, rare earth magnets are commonly used because

of their high strength. Flat magnets work well on smooth, flat surfaces, while dual-rail magnets are required for curved surfaces such as motor housings and pipes. In the case of non-magnetic or rough surfaces, it is recommended that the user first weld, epoxy, or otherwise adhere a steel mounting pad to the test surface. This provides a smooth location for mounting and a target to insure that subsequent measurements for trending purposes are taken at the same location.

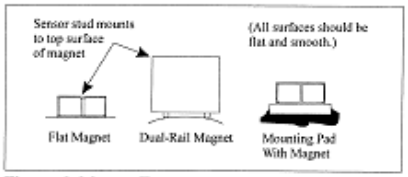


Figure 6. Magnet Types

STEP 1: Prepare a smooth, flat mounting surface. A minimum surface finish of 63 µin (0.00016 mm) generally works best. After cleaning the surface and checking for burrs, apply a light film of silicone grease, machine oil, or similar-type coupling fluid.

STEP 2: After choosing the correct magnet type, inspect the magnet, verifying that its mounting surfaces are flat and smooth.

STEP 3: Stud-mount the accelerometer to the appropriate magnet according to the guidelines set forth in **STEP 3** of the above Stud Mount Procedure.

STEP 4: To avoid damage to the sensor, install the magnet/sensor assembly to the prepared test surface by gently "rocking" or "sliding" it into place.

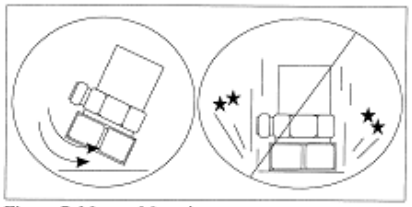


Figure 7. Magnet Mounting

CAUTION: *Magnetically mounting of an accelerometer has the potential to generate very high (and very damaging) acceleration (g) levels. To prevent such damage, exercise caution and install the assembly gently by rocking it into place. If shock is expected to be a particular concern, use a sensor with built-in shock protection. For further assistance, contact a factory representative.*

4.4 HANDHELD OR PROBE TIP MOUNT

This method is NOT recommended for most applications. Both the accuracy and repeatability at low (<5 Hz) and high frequency (>1 kHz) ranges are questionable. It is generally used only for machinery condition monitoring, when installation space is restricted, or other portable trending applications. The technique, however, can be useful for initially determining locations of greatest vibration to establish a permanent sensor installation point.

5.0 CABLING

Care and attention to cable installation and cable condition is essential as the reliability and accuracy of any measurement system is no better than that of its weakest link. Do to the nature of vibration measurements, all sensor cables will ultimately fatigue and fail. Good installation practice will extend the life of a cable, however, it is highly recommended to keep spare cables on hand to enable continuation of the test in the event of a cable failure.

STEP 1: Ascertain that you have the correct cable type.

One cable type cannot satisfy all applications. ICP® sensors can be operated with any ordinary two-wire or coaxial cable. Special, low-noise cables that are typically recommended for use with high-impedance, charge-output sensors can also be used. For applications requiring conformity to CE, low noise cables are essential. Industrial applications often require shielded, twisted-pair cables to reduce the effects of EMI and RFI that is present near electrical motors and machinery. Teflon-jacketed cabling may be necessary to withstand corrosive environments and higher temperatures. Consult the Vibration Division's product catalog for more information about cables or feel free to contact a factory representative for a specific recommendation on cables that are best suited for your application.

STEP 2: Connect the cable to the accelerometer.

A small amount of thread-locking compound placed on the connector threads prior to attachment helps secure the cable during testing. In wet, oily, or dirty environments, the connection can be sealed with silicone rubber sealant, O-rings, and flexible, heat-shrink tubing.

Coaxial Cables: Make connection by inserting the cable's connector pin into the sensor's mating socket. Then thread the connector into place by turning the cable connector's outer shell onto the accelerometer's electrical connector.

NOTE: *Do not spin the accelerometer while holding the cable connector stationary, as this will cause undue*

friction on the center pin of the cable connector and lead to premature fatigue.

Multi-pin connectors: Make connection by inserting the sensor's mating pins onto the cable connector's mating sockets. Then thread the connector into place by turning the cable connector's outer shell onto the accelerometer's electrical connector.

Pigtail Connections: Certain miniature accelerometers and shock sensors are provided with lightweight cables attached to "Pigtail" connections. This type of connection reduces overall weight and incidence of connection intermittency under shock conditions. In the event of a cable or connection failure, the cables may be repaired in the field simply by re-soldering the stripped leads to the exposed pins on the sensor. (Check the **Installation Drawing** to determine signal and ground pins). In many cases, it is also helpful to protect the solder joint with heat-shrink tubing or epoxy.

NOTE: *If you do not have the experience or resources to attach pigtail leads, consult PCB to discuss factory attachment. Damage to internal electronics may be caused by excessive heat during soldering and such failure is not covered by warranty.*

STEP 3: Route the cable to the signal conditioner, making certain to relieve stress on the sensor/cable connection. Also, minimize cable motion by securing it with tape, clamps or ties at regular intervals.

Common sense should be used to avoid physical damage and minimize electrical noise. For instance, avoid routing cables near high-voltage wires. Do not route cables along floors or walkways where they may be stepped on or become contaminated. To avoid ground loops, shielded cables should have the shield grounded at one end only, typically at the signal conditioner.

STEP 4: Finally, connect the remaining cable end to the signal conditioner. It is good practice to dissipate any electrical charge that may have accumulated in the cable by shorting the signal pin to the ground pin or shell prior to attachment.

6.0 POWERING

All ICP® sensors require constant current excitation for proper operation. For this reason, use only PCB constant-current signal conditioners or other approved constant-current sources. A typical system schematic is shown in Figure 8.

NOTE: *Damage to the built-in electronics resulting from the application of incorrect power, or the use of an unapproved power source, is NOT covered by warranty.*

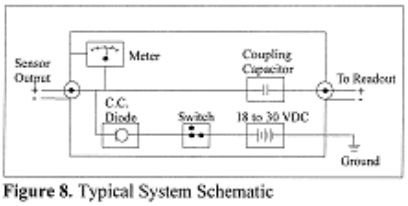


Figure 8. Typical System Schematic

The power supply consists of a current-regulated, 18 to 30 VDC source. This power is regulated by a current-limiting circuit, which provides the constant-current excitation required for proper operation of ICP® sensors. In general, battery-powered devices offer versatility for portable, low-noise measurements, whereas line-powered units provide the capability for continuous monitoring. Consult the Vibration Division's product catalog for more information about signal conditioners.

NOTE: *Under no circumstances should a voltage be supplied to an ICP® accelerometer without a current-regulating diode or equivalent electrical circuit. This may include ohmmeters, multi-meters and continuity testers.*

Meters or LEDs are used on PCB signal conditioners to monitor the bias voltage on the sensor output signal, to check sensor operation, and detect cable faults. Normally, a "yellow" reading indicates an open circuit; "green" indicates normal operation; and "red" indicates either a short or overload condition. Finally, a capacitor at the output stage of the device removes the sensor output bias voltage from the measurement signal. This provides a zero-based, AC-coupled output signal that is compatible with most standard readout devices.

NOTE: *Units having a low bias voltage may be in the "red," when actually they are working properly. If suspect, the bias voltage can be checked with a voltmeter attached to a "T" connector installed on the input connector to the signal conditioner.*

Note: *For readout devices having an input impedance near one gigohm (as encountered with some A to D converters), it may be necessary to place a one megohm resistor in parallel to the readout input to eliminate slow turn-on and signal drift.*

Today, many FFT analyzers, data acquisition modules, and data collectors have the proper constant-current excitation built-in for direct use with ICP® sensors. Before using this feature, however, check that the supply voltage and constant current are within acceptable limits for use with your particular sensor. (Check enclosed **Specification Sheet**). Please contact the respective signal

conditioner manufacturer or check the product manual for more information.

7.0 OPERATING

After completing the system setup, switch on the signal conditioner and allow 1 to 2 minutes for the system to stabilize. The meter (or LED) on the signal conditioner should be reading "green." This indicates proper operation and you may begin taking measurements. If a faulty condition is indicated (red or yellow reading), first check all system connections, then check the functionality of the cable and signal conditioner. If the system still does not operate properly, consult a PCB factory representative.

NOTE: *Always operate the accelerometer within the limitations listed on the enclosed* ***Specification Sheet****. Operating the device outside these parameters can cause temporary or permanent damage to the sensor.*

8.0 ACCELEROMETER CALIBRATION

Accelerometer calibration provides, with a definable degree of accuracy, the necessary link between the physical quantity being measured and the electrical signal generated by the sensor. In addition, other useful information concerning operational limits, physical parameters, electrical characteristics, or environmental influences may also be determined. Without this link, analyzing data becomes a nearly impossible task. Fortunately, most sensor manufacturers provide a calibration record that documents the exact characteristics of each sensor. (The type and amount of data varies depending on the manufacturer, sensor type, contractual regulations, and other special requirements).

Under normal conditions, piezoelectric sensors are extremely stable, and their calibrated performance characteristics do not change over time. However, the sensor may be temporarily or permanently affected by harsh environments influences or other unusual conditions that may cause the sensor to experience dynamic phenomena outside of its specified operating range. This change manifests itself in a variety of ways, including: a shift of the sensor resonance due to a cracked crystal; a temporary loss of low-frequency measuring capability due to a drop in insulation resistance; or total failure of the built-in microelectronic circuit due to a high mechanical shock.

For these reasons, it is recommended that a recalibration cycle be established for each accelerometer. This schedule is unique and is based on a variety of factors, such as: extent of use, environmental conditions, accuracy requirements, trend information obtained from previous calibration records, contractual regulations, frequency of "cross-checking" against other equipment, manufacturer recommendation, and any risk associated with incorrect readings. International standards, such as ISO 10012-1, provide insight and suggest methods for determining recalibration intervals for most measuring equipment. With the above information in mind and under "normal" circumstances, PCB conservatively suggests a 12- to 24-month recalibration cycle for most piezoelectric accelerometers.

NOTE: *It is good measurement practice to verify the performance of each accelerometer with a Handheld Shaker or other calibration device before and after each measurement. The PCB Handheld Shaker operates at a fixed frequency and known amplitude (1.0 g) to provide a quick check of sensor sensitivity.*

8.1 RECALIBRATION SERVICE

PCB offers recalibration services for our piezoelectric accelerometers, as well as units produced by other manufacturers. Our internal metrology laboratory is certified to ISO 9001, accredited by A2LA to ANSI/IEC 17025 and ANSI/NCSL Z540-1, complies with ISO 10012-1 (and former MIL-STD-45662A), and uses equipment directly traceable to NIST. Our investment in equipment, traceability and conformance to industry standards ensures accurate calibration against relevant specifications, in a timely fashion.

8.2 BACK-TO-BACK CALIBRATION THEORY

Many companies choose to purchase the equipment necessary to perform the recalibration procedure in house. While this may result in both a savings of time and money, it has also been attributed to incorrect readings and costly errors. Therefore, in an effort to prevent the common mistakes associated with customer-performed calibration, this document includes a broad overview of the Back-to-Back Calibration technique. This technique provides a quick and easy method for determining the sensitivity of a test accelerometer over a wide frequency range.

Back-to-Back Calibration is perhaps the most common method for determining the sensitivity of piezoelectric accelerometers. This method relies on a simple comparison to a previously calibrated accelerometer, typically referred to as a reference standard.

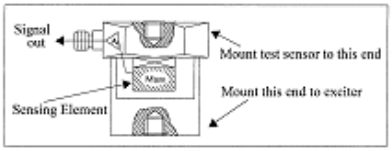


Figure 9. Reference Standard Accelerometer

These high-accuracy devices, which are directly traceable to a recognized standards laboratory, are designed for stability, as well as configured to accept a test accelerometer. By mounting a test accelerometer to the reference standard and then connecting this combination to a suitable vibration source, it is possible to vibrate both devices and compare the data as shown in Figure 10. (Test set-ups may be automated and vary, depending on the type and number of accelerometers being calibrated).

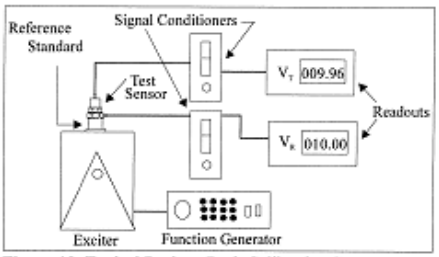


Figure 10. Typical Back-to-Back Calibration System

Because the acceleration is the same on both sensors, the ratio of their outputs (V_T/V_R) must also be the ratio of their sensitivities. With the sensitivity of the reference standard (S_R) known, the exact sensitivity of the test sensor (S_T) is easily calculated by using the following equation:

$$S_T = S_R\,(V_T/V_R)$$

By varying the frequency of the vibration, the sensor may be calibrated over its entire operating frequency range. The typical response of an unfiltered accelerometer is shown in Figure 11.

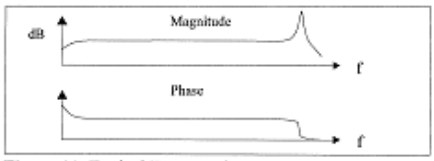


Figure 11. Typical Test Accelerometer Response

8.3 PCB CALIBRATION PROCEDURE

Numerous precautions are taken at PCB to insure accurate and repeatable results. This section provides a brief overview of the primary areas of concern.

Since the Back-to-Back Calibration technique relies on each sensor experiencing an identical acceleration level, proper mounting of the test sensor to the reference standard is imperative. Sensors with mounting holes are attached directly to the reference standard with a stud tightened to the recommended mounting torque. A shouldered mounting stud is typically used to prevent the stud from "bottoming out" in the hole. Both mounting surfaces are precision-machined and lapped to provide a smooth, flat interface according to the manufacturer's specification. A thin layer of silicone grease is placed between the mating surfaces to fill any imperfections and increase the mounting stiffness. The cables are stress-relieved by first routing them to the shaker head, then to a nearby stationary location. This reduces cable motion, which is especially important when testing charge output sensors, and helps to prevent extraneous motion or stresses from being imparted into the system. A typical set-up is shown in Figure 12.

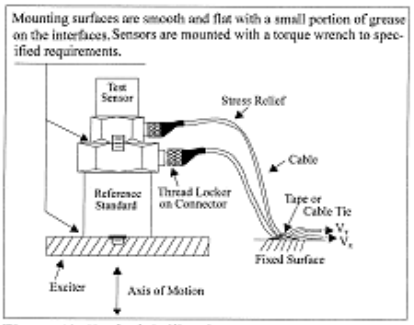


Figure 12. Typical Calibration Set-Up

Adhesively mounted sensors use similar practices. However, in this case, a small portion of quick-bonding gel, or similar temporary adhesive, is used to attach the test sensor to a reference standard designed with a smooth, flat mounting surface.

In addition to mounting, the selection of the proper equipment is critical. Some of the more important considerations include: 1) the reference standard must be specified and previously calibrated over the frequency and/or amplitude range of interest; 2) the shaker should be selected to provide minimal transverse (lateral) motion and minimal distortion; and 3) the quality of the meters, signal generator, and other devices should be selected so as to operate within the limits of permissible error.

8.4 COMMON MISTAKES

Most calibration errors are caused by simply overlooking some of the fundamental principals of dynamics. This section attempts to address some of the more common concerns.

For stud-mount sensors, always mount the accelerometer directly to the reference standard. Ensure that the mounting surfaces are smooth, flat, and free of any burrs. Always use a

coupling fluid, such as silicone grease, in the mounting interface to maintain a high mounting stiffness. Mount the sensor according to the manufacturer's recommended mounting torque. DO NOT use any intermediate mounting adaptors, as the mounted resonant frequency may be reduced, and thereby compromise the high-frequency performance. If necessary, use adaptor studs.

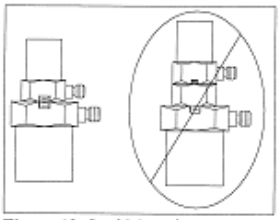


Figure 13. Stud Mounting

For adhesive mount sensors, use a thin, stiff layer of temporary adhesive such as quick-bonding gel or superglue. DO NOT use excessive amounts of glue or epoxy, as the mounting stiffness may be reduced and compromise high-frequency performance. It may also damage the sensor during removal.

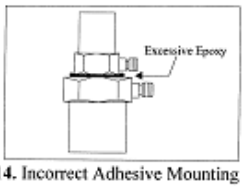


Figure 14. Incorrect Adhesive Mounting

Triaxial accelerometers should always be mounted directly to the reference standard. Unless absolutely required, DO NOT use adaptors to re-orient the sensor along the axis of motion, as the mounting stiffness may be altered. The vibration at the test sensor's sensing element may differ from the vibration at the reference standard due to a "cantilever" effect, seen in Figure 15.

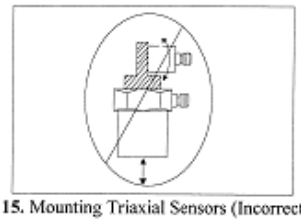


Figure 15. Mounting Triaxial Sensors (Incorrect)

Understand Back-to-Back Calibration limitations. Do not expect the uncertainty of calibration to be any better than ±2%. (In fact, the uncertainty may be as high as ±3% or ±4% for frequencies <10 Hz or >2 kHz.) Since large sensors may affect high-frequency accuracy, verify that the test sensor does not mass load the reference standard. Validate your calibration system with another accelerometer prior to each calibration session. Check with the manufacturer for exact system specifications.

8.5 CONCLUSIONS

Without an adequate understanding of dynamics, determining what, when, and how to test a sensor is a difficult task. Therefore, each user must weigh the cost, time, and risk associated with self-calibration versus utilizing the services of an accredited laboratory.

9.0 SERVICE

See the supplement sheet, contained in this manual, for information on our warranty, service, repair, and return policies and instructions.

When unexpected measurement problems arise, call our 24-hour SensorLine℠ to discuss your immediate dynamic instrumentation needs with a factory representative.
Dial 716-684-0001.

PCB PIEZOTRONICS INC
VIBRATION DIVISION

3425 Walden Avenue, Depew, NY 14043-2495 USA **Vibration Division toll-free** 888-684-0013
24-hour SensorLine[SM] 716-684-0001 **FAX** 716-685-3886 **E-mail** vibration@pcb.com **Website** www.pcb.com

A PCB GROUP COMPANY ISO 9001 CERTIFIED A2LA ACCREDITED to ISO 17025

Manual Number: 18292
Manual Revision: B
ECN Number: 19829

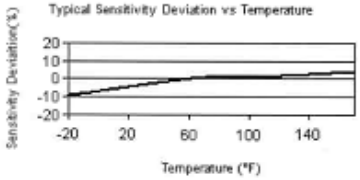
VIB-ICPMANUAL-0
Printed in U.S

9

Model Number	TRIAXIAL ICP® ACCELEROMETER	Revision: G
356B18		ECN #: 28468

Performance	ENGLISH	SI	
Sensitivity(± 10 %)	1000 mV/g	102 mV/(m/s²)	
Measurement Range	± 5 g pk	± 49 m/s² pk	
Frequency Range(± 5 %)	0.5 to 3000 Hz	0.5 to 3000 Hz	
Frequency Range(± 10 %)	0.3 to 5000 Hz	0.3 to 5000 Hz	
Resonant Frequency	≥ 20 kHz	≥ 20 kHz	
Phase Response(± 5 °)(at 70°F [21°C])	2 to 2000 Hz	2 to 2000 Hz	
Broadband Resolution(1 to 10,000 Hz)	0.00005 g rms	0.0005 m/s² rms	[1]
Non-Linearity	≤ 1 %	≤ 1 %	[2]
Transverse Sensitivity	≤ 5 %	≤ 5 %	
Environmental			
Overload Limit(Shock)	± 5000 g pk	± 5000 g pk	
Temperature Range(Operating)	-20 to +170 °F	-29 to +77 °C	
Temperature Response	See Graph	See Graph	
Base Strain Sensitivity	0.0007 g/με	0.007 (m/s²)/με	[1]
Electrical			
Excitation Voltage	20 to 30 VDC	20 to 30 VDC	
Constant Current Excitation	2 to 20 mA	2 to 20 mA	
Output Impedance	≤ 600 ohm	≤ 600 ohm	
Output Bias Voltage	8 to 12 VDC	8 to 12 VDC	
Discharge Time Constant	0.8 to 3.0 sec	0.8 to 3.0 sec	
Settling Time(within 10% of bias)	<12 sec	<12 sec	
Spectral Noise(1 Hz)	11.4 μg/√Hz	112 (μm/s²)/√Hz	[1]
Spectral Noise(10 Hz)	4.0 μg/√Hz	39 (μm/s²)/√Hz	[1]
Spectral Noise(100 Hz)	1.2 μg/√Hz	12 (μm/s²)/√Hz	[1]
Spectral Noise(1 kHz)	0.4 μg/√Hz	4.4 (μm/s²)/√Hz	[1]
Physical			
Sensing Element	Ceramic	Ceramic	
Sensing Geometry	Shear	Shear	
Housing Material	Anodized Aluminum	Anodized Aluminum	
Sealing	Epoxy	Epoxy	
Size (Height x Length x Width)	0.80 in x 1.03 in x 0.80 in	20.3 mm x 26.1 mm x 20.3 mm	
Weight	0.88 oz	25 gm	[1]
Electrical Connector	1/4-28 4-Pin	1/4-28 4-Pin	
Electrical Connection Position	Side	Side	
Mounting Thread	10-32 Female	10-32 Female	

CE [3]



All specifications are at room temperature unless otherwise specified.
In the interest of constant product improvement, we reserve the right to change specifications without notice.

ICP® is a registered trademark of PCB Group, Inc.

OPTIONAL VERSIONS

Optional versions have identical specifications and accessories as listed for the standard model except where noted below. More than one option may be used.

A - Adhesive Mount

Mounting Thread	None - Adhesive Mount Only	None - Adhesive Mount Only
Supplied Accessory : Model 080A109 Petro Wax (1)		
Supplied Accessory : Model 080A90 Quick Bonding Gel (1)		

J - Ground Isolated

Electrical Isolation(Base)	$>10^8$ ohm	$>10^8$ ohm
Size - Height x Length x Width	0.85 in x 1.03 in x 0.85 in	21.6 mm x 26.1 mm x 21.6 mm

T - TEDS Capable of Digital Memory and Communication Compliant with IEEE P1451.4
TLA - TEDS LMS International - Free Format
TLB - TEDS LMS International - Automotive Format
TLC - TEDS LMS International - Aeronautical Format
TLD - TEDS Capable of Digital Memory and Communication Compliant with IEEE 1451.4

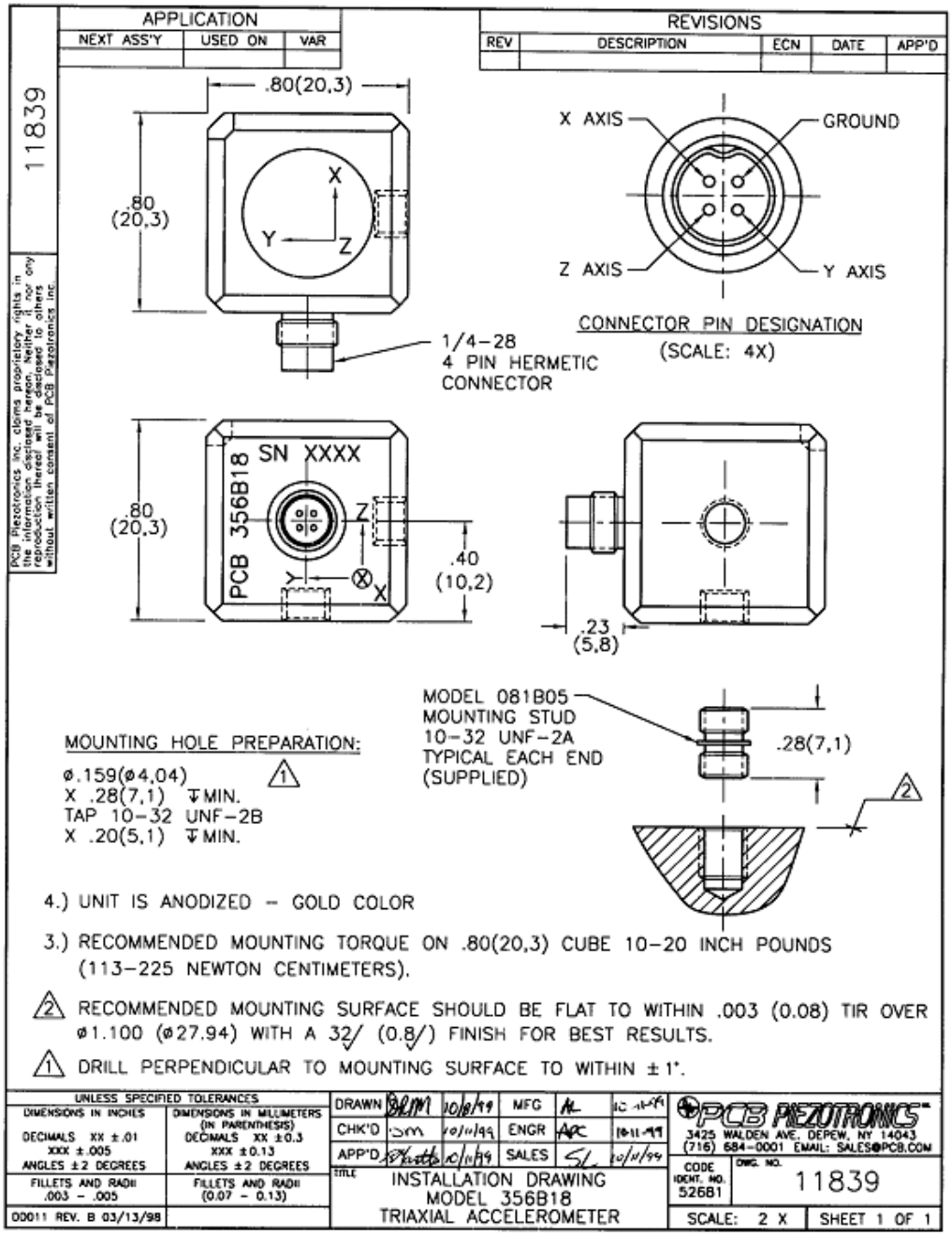
Output Bias Voltage	8.5 to 13 VDC	8.5 to 13 VDC

NOTES:
[1] Typical.
[2] Zero-based, least-squares, straight line method.
[3] See PCB Declaration of Conformance PS023 for details.

SUPPLIED ACCESSORIES:
Model 080A109 Petro Wax (1)
Model 080A68 Adhesive mounting base (for Models 356B07 and 356B08) (1)
Model 081B05 Mounting Stud (10-32 to 10-32) (1)
Model ACS-1T NIST traceable triaxial amplitude response, 10 Hz to upper 5% frequency. (1)
Model M081B05 Mounting Stud 10-32 to M6 X 0.75 (1)

Entered: BLS	Engineer: [illegible]	Sales: [illegible]	Approved: [illegible]	Spec Number:
Date: 5-16-07	Date: 5/16/07	Date: 5/16/07	Date: 5/16/07	11615

PCB PIEZOTRONICS™
VIBRATION DIVISION
3425 Walden Avenue, Depew, NY 14043
Phone: 716-684-0001
Fax: 716-685-3886
E-Mail: vibration@pcb.com


APPLICATION
NEXT ASS'Y
USED ON
VAR
REVISIONS
REV
DESCRIPTION
ECN
DATE
APP'D
11839
.80(20,3)
.80
(20,3)
X
Y
Z
1/4-28
4 PIN HERMETIC
CONNECTOR
X AXIS
GROUND
Z AXIS
Y AXIS
CONNECTOR PIN DESIGNATION
(SCALE: 4X)
SN XXXX
PCB 356B18
.40
(10,2)
.23
(5,8)
MODEL 081B05
MOUNTING STUD
10-32 UNF-2A
TYPICAL EACH END
(SUPPLIED)
.28(7,1)
MOUNTING HOLE PREPARATION:
ø.159(ø4,04)
X .28(7,1) ↧MIN.
TAP 10-32 UNF-2B
X .20(5,1) ↧MIN.
4.) UNIT IS ANODIZED - GOLD COLOR
3.) RECOMMENDED MOUNTING TORQUE ON .80(20,3) CUBE 10-20 INCH POUNDS
(113-225 NEWTON CENTIMETERS).
RECOMMENDED MOUNTING SURFACE SHOULD BE FLAT TO WITHIN .003 (0.08) TIR OVER
ø1.100 (ø27.94) WITH A 32/ (0.8/) FINISH FOR BEST RESULTS.
DRILL PERPENDICULAR TO MOUNTING SURFACE TO WITHIN ± 1°.
UNLESS SPECIFIED TOLERANCES
DIMENSIONS IN INCHES
DECIMALS XX ±.01
XXX ±.005
ANGLES ±2 DEGREES
FILLETS AND RADII
.003 - .005
DIMENSIONS IN MILLIMETERS
(IN PARENTHESIS)
DECIMALS XX ±0.3
XXX ±0.13
ANGLES ±2 DEGREES
FILLETS AND RADII
(0.07 - 0.13)
00011 REV. B 03/13/98
DRAWN
CHK'D
APP'D
MFG
ENGR
SALES
TITLE
INSTALLATION DRAWING
MODEL 356B18
TRIAXIAL ACCELEROMETER
PCB PIEZOTRONICS
3425 WALDEN AVE. DEPEW, NY 14043
(716) 684-0001 EMAIL: SALES@PCB.COM
CODE IDENT. NO. 52681
DWG. NO. 11839
SCALE: 2 X
SHEET 1 OF 1
PCB Piezotronics inc. claims proprietary rights in the information disclosed hereon. Neither it nor any reproduction thereof will be disclosed to others without written consent of PCB Piezotronics inc.

C. Datenblatt Kraftsensor

Product Data

Force Transducers — Types 8200 and 8201

USES:

- ❍ Dynamic, short duration static and impact force measurements in machinery, buildings etc.
- ❍ Measurement of Frequency Response Functions when used together with an accelerometer

FEATURES:

- ❍ Wide force ranges
- ❍ Extremely good linearity
- ❍ High long term stability
- ❍ All welded, hermetically sealed construction
- ❍ Small size
- ❍ Wide operating temperature range
- ❍ Individually calibrated
- ❍ High resolution
- ❍ Easily mounted

These transducers are designed to measure dynamic, short duration static and impact, tensile and compressive forces in machinery and other constructions. They are mounted so that the force to be measured is transmitted through the transducer. Used with vibration exciters they can measure and control the applied force, and can be used for the measurement of Frequency Response Functions in conjunction with an accelerometer.

Type 8200 is a small, permanently pre-loaded transducer usable in the force range 1000 N tensile to 5000 N compressive (−225 lbf to + 1225 lbf). Its top seismic mass is low (3 grams), this feature being particularly important when measuring on low impedance structures. The transducer is mounted using the threaded spigot and tapped hole in the body.

Type 8201 is a general purpose transducer with high force measuring capability. It is supplied with pre-loading nuts for the measurement of tensile and compressive forces in the range −4000 N to +16000 N (−900 lbf to +3600 lbf). With the pre-loading nuts removed, compressive forces up to 20000 N (+4500 lbf) are measurable. Its calibration is still valid after remounting the pre-loading nuts. The 8201 is dimensioned so that it is easily connected or built into the measuring object without the need for altering or removing existing mechanical elements.

8200
-1000 N to +5000 N
(-225 lbf to +1225 lbf)

8201
-4000 N to +20000 N
(-900 lbf to +4500 lbf)

The transducers have a rugged, all welded, hermetically sealed construction with a ceramic insulated microplug connector sealed with moulded glass, allowing them to be used under very severe environmental conditions. The allowable temperature operating range is −196°C to + 200°C for the Type 8200 and −196°C to + 150°C for the Type 8201.

Sectional drawings of the two force transducers are shown in Fig. 2. Both transducers work on the piezoelectric effect of quartz which, when stressed, produces an electrical charge which is proportional to the stressing force.

Brüel & Kjær

This charge is measured electronically to accurately determine the force to which the transducer is subjected. The quartz piezoelectric element has a very low sensitivity to temperature changes and temperature transients.

The high overall stiffness of the transducers ensures that they have a high resonance frequency and that when a transducer is introduced into a mechanical system it has a minimal disturbing effect due to deformation. As the internal capacitance of the transducers is very low, use of charge amplifiers such as the Brüel & Kjær Types 2626, 2634, 2635 and 2651 is recommended.

The use of charge amplifiers allows the use of long or varying lengths of input cables without disturbing the sensitivity calibration of the set-up. Low internal noise levels in these preamplifiers enable forces as low as 0.001 N (0.00022 lbf) to be resolved. The Force Transducers have a very high leakage resistance which allows the measurement of short duration static loads when used with the Brüel & Kjær Low Frequency Charge Amplifier Type 2651.

Each individual Force Transducer is calibrated before it leaves the factory and a calibration chart is included with the instrument, see Fig. 3. Since the piezoelectric discs are artificially aged during manufacture they are extremely stable, and under normal use the transducers should not need to be re-calibrated.

Type 8201 is calibrated both with and without the pre-loading nuts and

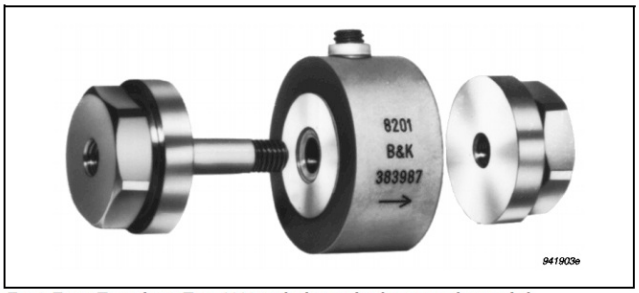


Fig. 1 Force Transducer Type 8201 with the pre-loading nuts dismantled

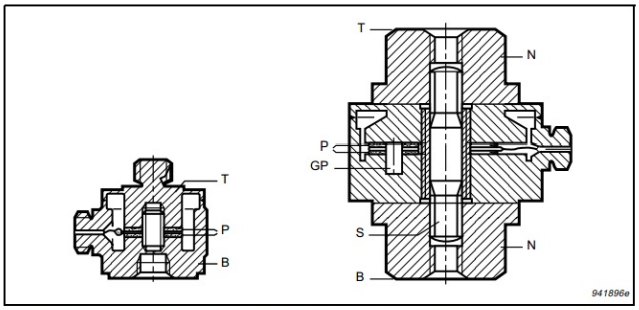


Fig. 2 Sectional drawing of the force transducers. T = Top, P = piezoelectric discs. GP = guide pin, S = pre-loading screw, N = pre-loading nut, B = base

provided that the stated pre-loading force is applied when remounting the load nuts, the original calibration for the pre-loaded 8201 is valid.

When used without the pre-loading nuts, the additional stiffness of any retaining screw through the centre hole will reduce the calibrated sensitivity. In this situation the set-up should be calibrated as a whole by applying a test force and noting the force transducer output. Alternatively the new sensitivity is calculated from the ratio of the screw and trans-

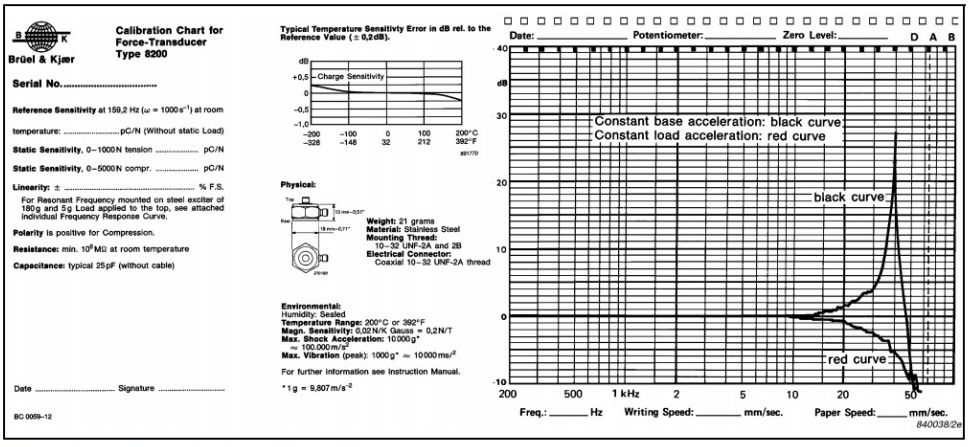


Fig. 3 Typical calibration chart supplied with the force transducers

Fig. 4 A Force Transducer Set

Fig. 5 Transducer Package (8200 only)

ducer stiffness being equal to the ratio of forces transmitted by them.

Force Transducer Type 8200 is available in two forms designated by the suffix "S" and "P" which follow the Type number. Force Transducer Sets (suffix "S") contain a single transducer in a mahogany case complete with cable and various accessories to aid mounting (accessory set UA 0330), see Fig. 4. The Force Transducer Packages (suffix "P") contain five individually packaged transducers complete with fixing studs and cables, see Fig. 5. Type 8201 is supplied only as a set, in a mahogany case, with accessories.

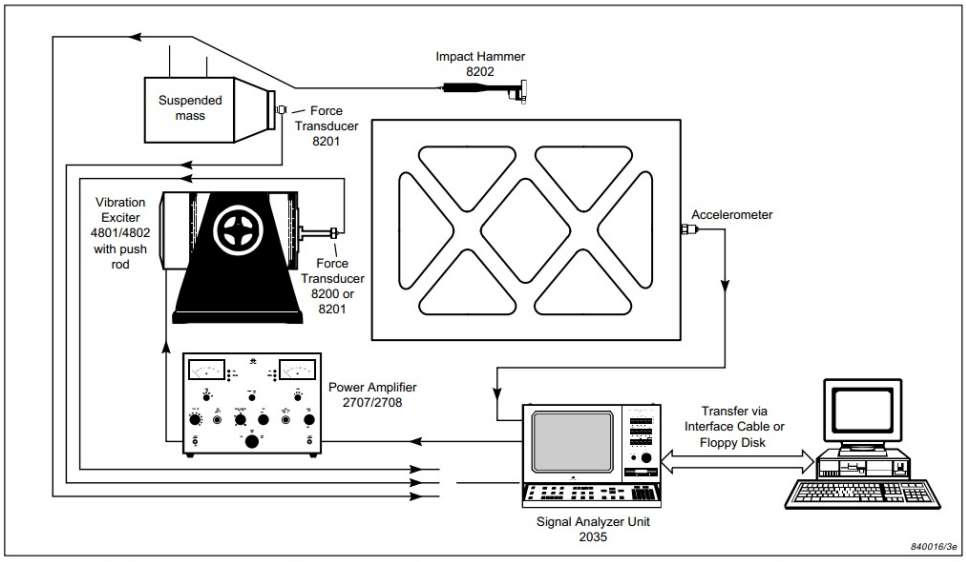
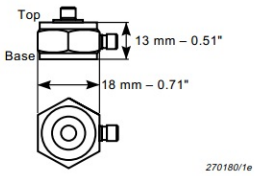
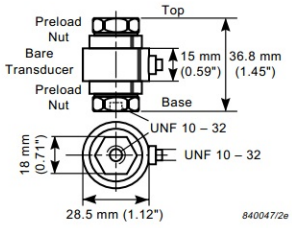


Fig. 6 Instrumentation for Structural Analysis, using Vibration Exciter, Suspended Mass, or Impact Hammer excitation techniques

Specifications 8200 and 8201

	8200	8201
Force range	1000 N tensile to 5000 N compressive (–225 lbf to +1125 lbf)	4000 N tensile to 16000 N compressive (–900 lbf to +3600 lbf) with pre-loading nuts. 20000 N compressive (+4500 lbf) without pre-loading nuts
Linearity[1]	< ±1% of max. force	< ±2% of max. force
Reproducibility of original sensitivity with re-mounting of pre-loaded nuts	—	< ±2% of max. force
Charge Sensitivity[1] (typical)	4 pC/N (17.8 pC/lbf)	4 pC/N (17.8 pC/lbf)
Capacitance (typical)	25 pF	70 pF
Leakage Resistance (at 25°C)	> 10^6 MΩ	> 10^6 MΩ
Stiffness	5×10^8 N/m (2.9×10^6 lbf/in)	7×10^8 N/m (4×10^6 lbf/in)
Deformation at Maximum Force	0.01 mm (0.0004 in)	0.03 mm (0.0012 in)
Resonance Frequency[1] **with 5 grams load mounted on top** (typical)	35 kHz	20 kHz
Effective Seismic Mass: **Above Piezoelectric Element (Top)**	3 grams	With pre-loading nuts: 43 grams Without pre-loading nuts: 16 grams
Below Piezoelectric Element (Base)	18 grams	With pre-loading nuts: 69 grams Without pre-loading nuts: 43 grams
Temperature Range	–196 to +200°C (–321 to +392°F)	–196 to +150°C (–321 to +302°F)
Temperature Transient Sensitivity (typical)	0.5 N/°C (0.06 lbf/°F)	0.4 N/°C (0.05 lbf/°F)
Transverse Sensitivity[2] (typical)	5%	4% (with pre-loading nuts mounted)
Maximum Transverse Force for stated Transverse Sensitivity	100 N (22.5 lbf)	500 N (112 lbf)
Bending Moment Sensitivity[3] (typical)	100 pC/Nm (136 pC/lbf ft)	25 pC/Nm (34 pC/lbf ft) (with pre-loading nuts mounted)
Maximum Bending Moment for stated Bending Moment Sensitivity	1 Nm (0.74 lbf ft)	10 Nm (7.4 lbf ft)
Strain Sensitivity[4] (top and base)	< 0.04 N (0.009 lbf) per µStrain	< 0.004 N (0.0009 lbf) per µStrain (with pre-loading nuts mounted)
Magnetic Sensitivity[4] **at 50 Hz** (typical)	0.2 N/T[5] (0.05 lbf/T)	0.9 N/T[5] (0.2 lbf/T)
Material	Stainless steel AISI 316	Stainless steel AISI 316
Weight	21 grams (0.046 lb.)	112 grams (0.25 lb.) with pre-loading nuts 58 grams (0.13 lb.) without pre-loading nuts
Dimensions		

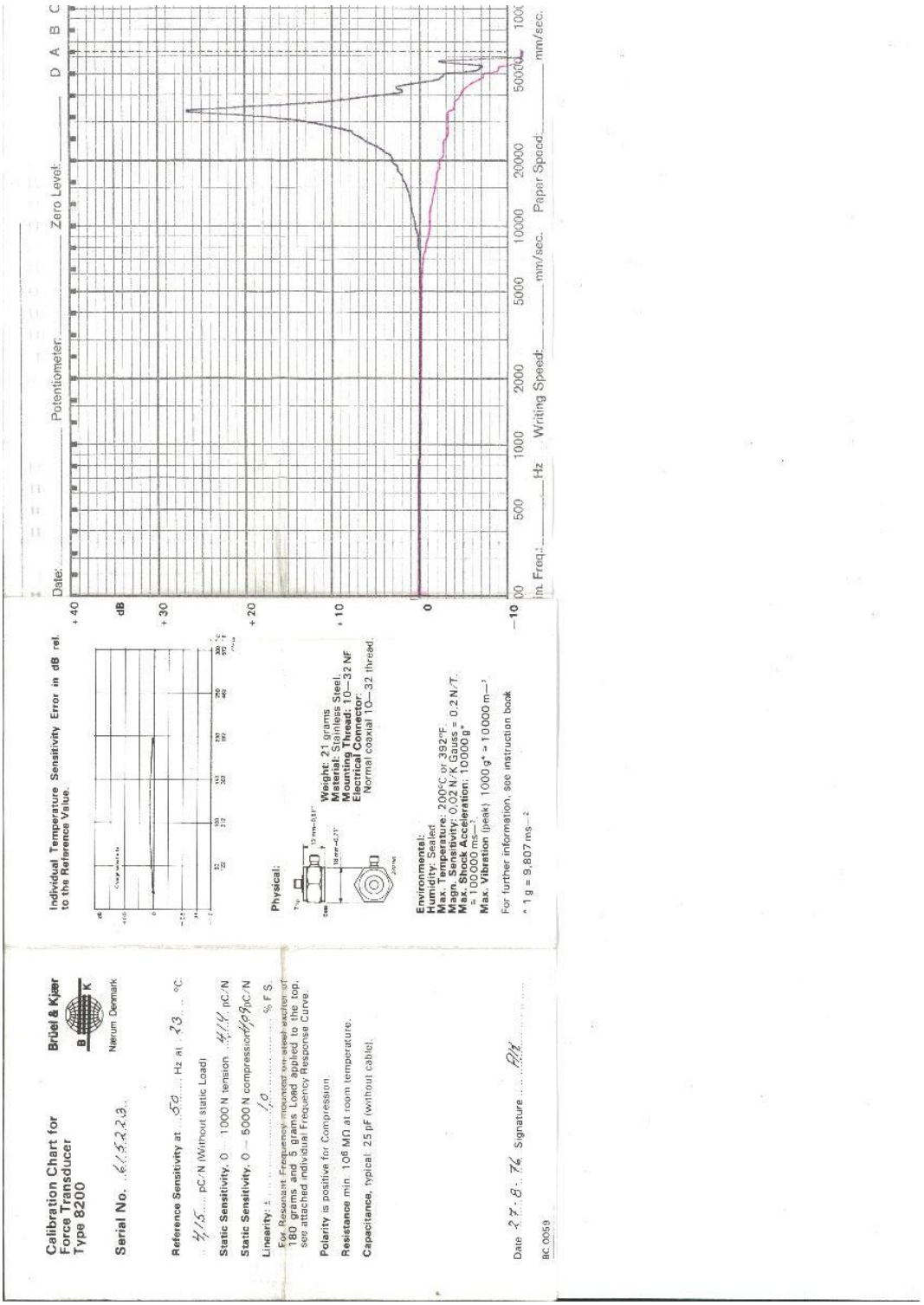
[1] Individual values given on the calibration chart
[2] For forces transverse to the main axis of the transducer
[3] The total bending moment is the sum of the resultant bending moments due to axial and transverse forces taken about the mid-point of the transducer
[4] Ref. ANSI S2.11–1969
[5] 1 Tesla = 10 kGauss

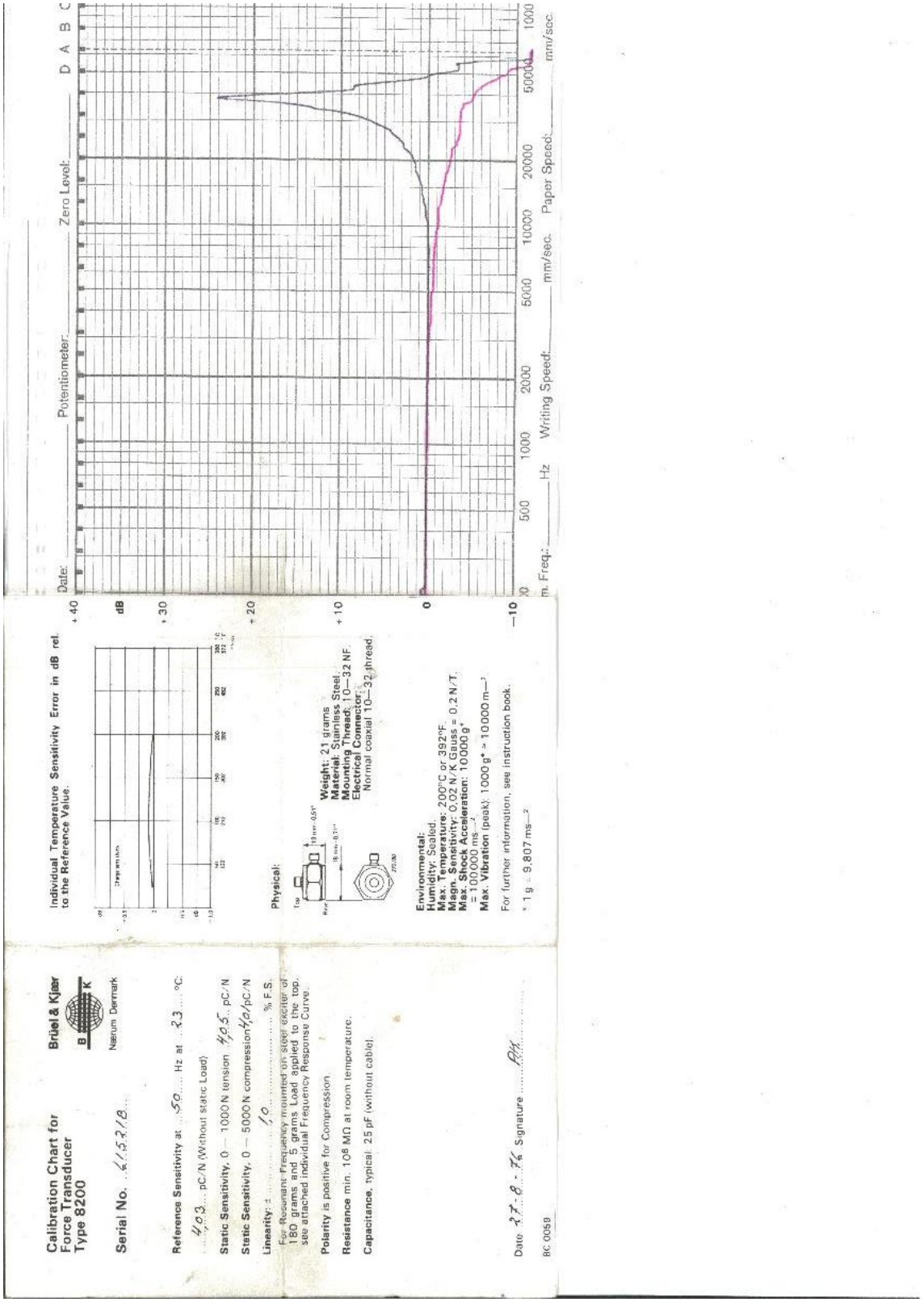
Brüel & Kjær reserves the right to change specifications and accessories without notice

Brüel & Kjær

WORLD HEADQUARTERS:
DK-2850 Naerum · Denmark · Telephone: +45 45 80 05 00 · Fax: +45 45 80 14 05 · Internet: http://www.bk.dk · e-mail: info@bk.dk
Australia (02) 9450-2066 · Austria 00 43-1-865 74 00 · Belgium 016/44 92 25 · Brazil (011) 246-8166 · Canada: (514) 695-8225 · China 10 6841 9625 / 10 6843 7426
Czech Republic 02-67 021100 · Finland 90-229 3021 · France (01) 69 90 69 00 · Germany 0610 3/908-5 · Holland (0)30 6039994 · Hong Kong 254 8 7486
Hungary (1) 215 83 05 · Italy (02) 57 60 4141 · Japan 03-3779-8671 · Republic of Korea (02) 3473-0605 · Norway 66 90 4410 · Poland (0-22) 40 93 92 · Portugal (1) 47114 53
Singapore (65) 275-8816 · Slovak Republic 07-37 6181 · Spain (91) 36810 00 · Sweden (08) 71127 30 · Switzerland 01/94 0 09 09 · Taiwan (02) 713 9303
United Kingdom and Ireland (0181) 954-236 6 · USA 1 - 800 - 332 - 2040
Local representatives and service organisations worldwide
BP0443–15

D. Kalibrierdaten Kraftsensor




Calibration Chart for
Force Transducer
Type 8200
Brüel & Kjær
Nærum Denmark
Serial No. 41532/8
Reference Sensitivity at 50 Hz at 23 °C:
403 pC/N (Without static Load)
Static Sensitivity, 0 — 1000 N tension 405 pC/N
Static Sensitivity, 0 — 5000 N compression 40a/pC/N
Linearity: ± 10 % F.S.
For Resonant Frequency mounted on steel anchor of 180 grams and 5 grams Load applied to the top, see attached individual Frequency Response Curve.
Polarity is positive for Compression
Resistance min. 10⁸ MΩ at room temperature.
Capacitance, typical 25 pF (without cable).
Date 27-8-76 Signature
BK 0059
Individual Temperature Sensitivity Error in dB rel. to the Reference Value.
Physical:
Weight: 21 grams
Material: Stainless Steel.
Mounting Thread: 10—32 NF.
Electrical Connector: Normal coaxial 10—32 thread.
Environmental:
Humidity: Sealed.
Max. Temperature: 200°C or 392°F
Max. Sensitivity: 0.02 N/K Gauss = 0.2 N/T.
Max. Shock Acceleration: 10000 g* = 100000 ms⁻²
Max. Vibration (peak): 1000 g* = 10000 m—²
For further information, see instruction book.
* 1 g = 9.807 ms⁻²
Date:
Potentiometer:
Zero Level:
dB
+40
+30
+20
+10
0
—10
m. Freq.: Hz
Writing Speed: mm/sec.
Paper Speed: mm/sec.
A B C D
10 500 1000 2000 5000 10000 20000 50000 1000

E. Datenblatt Ladungsverstärker

Request Quote

visib & Kjaer 2651 Charge Amplifier

Features

- General purpose vibration and shock measurements particularly with Uni-Gain® Accelerometers
- Multichannel measuring systems and vibration test control loops
- Preamplifier for piezoelectric impedance heads and force transducers
- Wide frequency range 3 Hz to 200 kHz
- Output sensitivity settings of 0,1 - 1 - 10 mV/pC
- Built-in integrator converts acceleration signal to velocity
- Grounded, plus Floating input mode isolated from output ground
- Switchable lower frequency limits

Specifications

Charge Input	via 10-32 UNF coaxial and multi-pin screened sockets	
Input Modes	"Grounded"	
	"Floating"	
Amplifier Sensitivity	0,1 mV/pC, 1 mV/pC and 10 mV/pC	
Signal Output	Via 10-32 UNF coaxial and multi-pin screened socket	
	Max Output:	10 V (10 mA) peak
	DC Offset:	< ± 0 mV with dual polarity supply or half single polarity supply voltage
Frequency Range	3 dB Lower Limit:	1 Hz for all three sensitivity settings or 0,003 Hz; 0,03 Hz and 0,3 Hz for 0,1; 1 and 10 mV/pC respectively
	3 dB Upper Limit:	>200 kHz for 0,1 and 1 mV/pC sensitivity and > 150 kHz for 10 mV/pC

Integrator	Converts acceleration to velocity proportional signal	
	Frequency Range:	10 Hz to 10 kHz (-10% limits)
Power Requirements	Dual Polarity:	±6 to ±18 VDC
	Single Polarity:	+12 to +35 VDC
	Supply Current:	15 mA nominal increasing to 30 mA with "Overload" LED lit

Bruel & Kjaer 2805 - Power Supply

- Power supply for preamplifiers in vibration measurement set-ups
- Power supply for instruments operating on 14 V and 28 V DC
- Used for supplying the following instruments with the direct current necessary for their operation:
  - Charge Amplifiers types 2634, 2635 and 2651
  - Accelerometer Calibrator Type 4291
  - Vibration Meter Type 2511
  - Tunable Band Pass Filter Type 1621
- 12 outputs, 2 channels
- ±14 V or 28 V or DC output

Specifications

Output	2 channels short circuit protected
Output Connections	For each channel 5 x 10 - 32 UNF sockets, also 1 x 6 pin Bulgin plug JP 4705
Output Voltage	28 Volts or ± 14 Volts
Output Current	Max. 200 mA per channel
Power Requirement	100, 115, 127, 220, 240 Volts (50 to 400 Hz)

Please contact us for more information on the Bruel & Kjaer 2651 Charge Amplifier

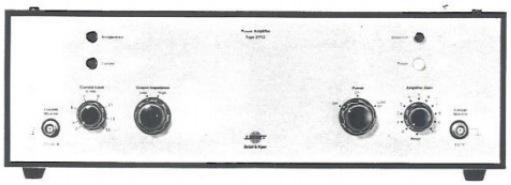
Featured Products:

F. Datenblatt Leistungsverstärker

Instruction Manual 2712

Power Amplifier Type 2712

HRKIV



A 180 VA transistorized Power Amplifier for driving small vibration exciters having a permanent magnet field, particularly the Vibration Exciters Types 4808 and 4809 which have a force rating of 112 N (25 lbf) and 45 N (10 lbf) respectively. Its useful frequency range extends from DC up to 100 kHz with full power output available from 40 Hz up to 10 kHz. Extensive protective functions with adjustable output current limit from 2 to 15 A RMS, safeguard against overtesting and system component failures.

033-0198

Brüel & Kjær

type 2712

180 VA Power Amplifier

FEATURES:

- Direct coupled solid state
- 180 VA power output
- Adjustable RMS output current limit
- Front panel control for Low or High output impedance
- Low distortion over wide frequency range
- Internally protected against current overload
- Extensive built-in protection with three indicator lights
- Front panel voltage and current monitor points

USES:

To drive

- Vibration Exciter Type 4808
- Vibration Exciter Type 4809 safely to full rating
- Vibration Exciter Type 4805 with associated heads at reduced rating

The Power Amplifier Type 2712 has been designed to drive small vibration exciters, particularly the 25 lbf (112 N) Vibration Exciter Type 4808. The RMS output current limit is adjustable, by a front panel control, and therefore this Power Amplifier will also drive the 10 lbf (45 N) Vibration Exciter Type 4809 safely to full rating. (The 2712 can also be used to drive the Vibration Exciter Type 4805 with associated heads at reduced rating.)

The Power Amplifier has a useable frequency range from DC to 100 kHz. The full AC output capability is 180 VA into a 0,8 Ω exciter or resistor load and is available in the frequency range 40 Hz to 10 kHz. The maximum voltage gain is 14 dB. Harmonic content of the output is very small as heavy negative feedback is used. A balanced preamplifier and the use of silicon transistors results in an instrument which can tolerate temperature and supply line variations while maintaining good stability.

Type 2712 can be used as a voltage generator with low output impedance and a flat voltage to frequency response, or as a current generator with high output impedance and a flat current to frequency response.

Description

A simplified block diagram of the Power Amplifier is shown in Fig.1. The instrument consists of an input stage, a preamplifier, a power amplifier and various warning and safety circuits with indication lamps.

Input

Both a capacitively coupled AC input and a direct coupled DC input are provided. Under normal working conditions the signal passes through a FET gate to the preamplifier stage. When the built-in protective circuitry is activated, however, the gate is triggered and disconnects the input signal from the preamplifier.

Preamplifier Section

The type of feedback from the output to the preamplifier stage is selected by the output impedance switch. Voltage feedback is used in the low impedance mode giving constant output voltage and very low output impedance. Feedback proportional to the current flowing in the load is used in the high impedance mode resulting in a constant output current and high output impedance.

Excessive signal levels will saturate the preamplifier and cause distortion of the output waveform. This will trigger the clipping detector which then lights the yellow DISTORTION warning lamp on the front panel. The instrument remains operative in this condition.

Power Output Section

From the preamplifier, the signal is fed to the power output stage. This is directly coupled to the output, and hence to a connected vibration exciter, to eliminate the need for a bulky output transformer. A current limiting circuit prevents instantaneous excessive positive and negative output current peaks.

As well as power amplification the 2712 provides system control and protection functions. During operation the voltage and current levels and waveforms can be inspected at the monitor points provided.

Protection

The Power Amplifier Type 2712 contains protection functions for the Power Amplifier itself and the connected vibration exciter. When triggered, they turn off the FET gate at the input thus disconnecting the input signal. Each triggered protective circuit also lights a red lamp which gives an indication of the reason for equipment shut-down.

17—177

Overload protection is provided for excessive coil drive current. This feature enables the 2712 to safely drive vibration exciters with different maximum current ratings. A front panel control is used to preset the true RMS output current at which the circuitry trips. The limit can be set anywhere between 2 A and 15 A RMS. The signal to the exciter is switched off if the preset driving coil current is exceeded, and the red CURRENT lamp will light.

The power output stage is protected by a temperature sensing safety device. Abnormal load conditions, high ambient temperatures or short circuited output can result in output transistor temperatures that exceed design limits and lead to transistor failure. To prevent such damage the temperature protective circuitry blocks the amplifier input signal, and the red TEMPERATURE lamp will light. Also excessive temperature of a power transistor triggers the protective circuitry and lights the lamp. Resetting after both current and temperature shutdown is made simply by turning the amplifier gain control fully counter-clockwise.

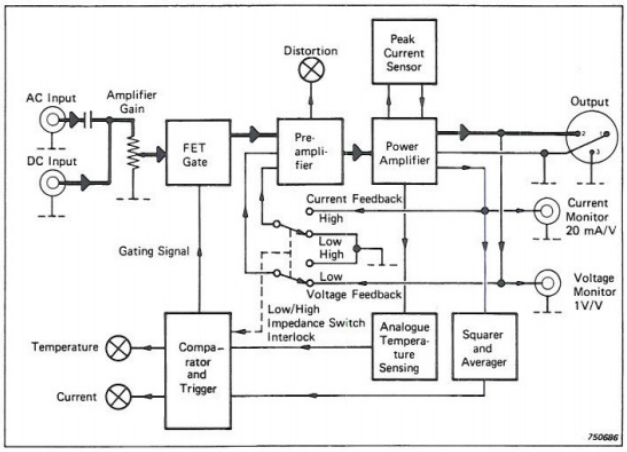


Fig.1. Simplified block diagram of the Power Amplifier

Specifications 2712

Power Output Capacity:
180 VA into a 0,8 Ω exciter or resistor load at 25°C and nominal mains voltage
144 VA into a 1 Ω exciter or resistor load at 40°C or at 10% above nominal mains voltage
(3-pin Cannon socket at rear panel)

Output Voltage Capacity:
12 V RMS, DC to 15 kHz

Output Current Capacity:
7,5 A RMS at or below 5 Hz
15 A RMS, 40 Hz to 10 kHz
12 A RMS at 15 kHz

Frequency Range:
Full capacity: 40 Hz to 10 kHz
Reduced capacity: DC to 100 kHz

Frequency Response:
Typical small signal response in low impedance mode:
DC Input: DC to 15 kHz ± 0,5 dB
DC to 100 kHz ± 3 dB
AC Input: 15 Hz to 15 kHz ± 0,5 dB
(2 separate BNC sockets at rear panel)

Input Impedance:
> 10 kΩ

DC Stability:
Less than 50 mV drift from 0 V for ± 10% variation of mains supply from nominal, and for 10 to 40°C (50 to 104°F) variation in ambient temperature

Protection:
Input signal is removed and an indicator lamp is lit when the following parameters exceed preset limits:
Driver Coil Current — true RMS adjustable limit 2 to 15 A
Power Transistor Temperature
Heat Sink Temperature
Front panel indication is provided for Output Signal Distortion — no shut-down

Low and High Impedance:

	Low Impedance	High Impedance
Gain at 1 kHz	5 V/V ± 2 dB	8 A/V ± 2 dB
Output Impedance	< 0,02 Ω 5 Hz to 1 kHz < 0,05 Ω DC to 15 kHz	> 20 Ω 5 Hz to 1 kHz > 50 Ω 20 Hz to 300 Hz > 80 Ω 40 Hz to 100 Hz
Harmonic Distortion (full capacity)	< 0,2% 5 Hz to 5 kHz < 0,5% 5 kHz to 15 kHz	< 0,4% 5 Hz to 2 kHz < 1% 2 kHz to 15 kHz
Noise and Hum (below full output)	at least 80 dB	at least 70 dB

Other Features:
Electronic peak current limiting
Voltage and Current monitor points (front panel BNC sockets)

Temperature Range:
5 to 40°C (41 to 104°F)

Power Requirements:
Single phase 100, 115, 127, 150, 220, 240, V RMS, ± 10%
Complies with safety class I of IEC 348

Cabinet:
Supplied as model A (light-weight metal cabinet), B (model A in a mahogany cabinet) or C (as A but with flanges for standard 19" racks)

Dimensions: (model A)
(excluding feet, knobs etc.):
Height: 133 mm (5,2 in)
Width: 430 mm (16,9 in)
Depth: 200 mm (7,9 in)

Weight: (model A)
14,5 kg (32 lb)

Accessories Included:
1 3-pin Cannon Plug JP 0308
3 BNC Plugs JP 0035
1 Mains Cable AN 0010
Various fuses

2. CONTROLS

2.1. FRONT PANEL

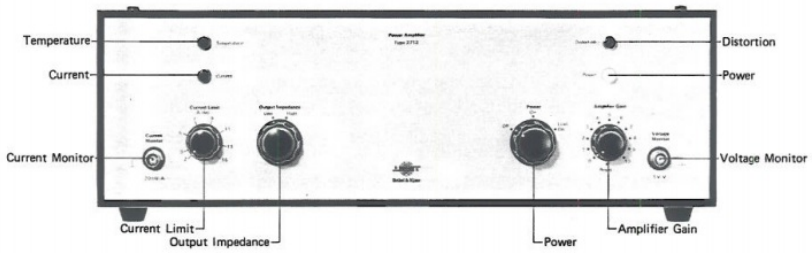


Fig.2.1. Front Panel of 2712

POWER: Three position switch for connection of power to the Amplifier and Vibration Exciter. The positions are:

"Off". Power off position. Mains supply and Exciter power lines are disconnected internally.

"Power On". Stand by position. Connects the mains supply leaving Exciter power lines disconnected. The white POWER lamp and Amplifier cooling fan should function.

"Load On". Exciter power lines are connected ready for operation.

Note: Never switch directly from "Power On" to "Load On". First wait a few seconds and ensure that the AMPLIFIER GAIN control is set to its "Reset" position. This will prevent a power surge which can cause automatic shut down of the Amplifier as well as a severe mechanical transient with the Exciter.

AMPLIFIER GAIN: Single turn potentiometer for adjustment of the Amplifier power output level to the Exciter. It has a click-stop "Reset" position for restoring operation of the Amplifier after automatic shut down when the red TEMPERATURE and CURRENT warning lamps are lit.

CURRENT LIMIT: Single turn potentiometer for adjustment of the Amplifier output current limit between 2 and 15 A RMS. Should be set to the maximum

drive current of the particular Exciter employed. Above the selected limit the Amplifier will be automatically shut down to protect the Exciter.

The CURRENT LIMIT protection circuitry employs a 60 s time constant matching the thermal-time constant of B & K Exciter moving coils.

OUTPUT IMPEDANCE: Two position switch for selection of feedback and output impedance mode. The positions are:

"Low". Provides constant voltage characteristics independent of test object changes on the Exciter. Gives best acceleration waveform and is therefore preferable for most vibration tests.

"High". Provides constant current characteristics, keeping generated force independent of test object changes.

VOLTAGE MONITOR: A BNC socket providing an output of the Amplifier voltage waveform (including DC component) for display on an Oscilloscope. It is direct coupled to the POWER OUTPUT socket on the rear panel and has an output sensitivity of 1 V/V.

CURRENT MONITOR: A BNC socket providing a phase inverted output of the Amplifier output current waveform (including DC component) for display on an oscilloscope. The output sensitivity is 20 mV/A.

Aside from the white POWER "On" lamp, there are three other indicator lamps on the front panel. These are:

TEMPERATURE: Red warning lamp indicating automatic shut down of the Amplifier when maximum operating temperature of power output transistors exceeded. To resume operation, see section 3.4.

CURRENT: Red warning lamp indicating automatic shut down of Amplifier due to excessive drive current. To resume operation, see section 3.4.

DISTORTION: Amber lamp indicating clipping of current and voltage output waveform to the Exciter. The Amplifier will continue to operate, but the input drive level must be reduced to resume normal operation.

2.2. REAR PANEL

SIGNAL INPUT DC: BNC socket providing a direct coupled input to the Amplifier. Enables a DC offset voltage to be applied for centring the vibration table of an Exciter when this is statically offset by heavy test specimens. The input impedance is 10 KΩ. Full output is produced by an input signal of 3,4 V peak.

SIGNAL INPUT AC: BNC socket providing a capacitive coupled input to the Amplifier. The —0,5 dB lower limiting frequency is between 10 and 15 Hz. Full output is produced by an input signal of 2,4 V RMS.

POWER OUTPUT: Power output socket accepting the 3 pin Cannon plug WK-C3-32C (B & K order no. JP 0308) provided, for connection of an Exciter as dis-

cussed in section 3.2.2. For full output power of 180 VA the moving coil of the Exciter should have a nominal load impedance of 0,8 Ω.

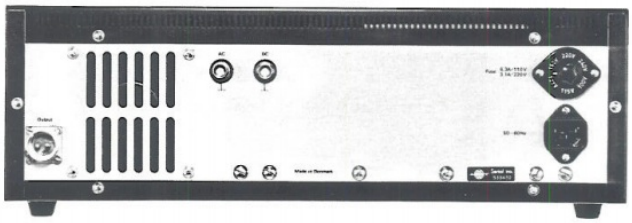


Fig.2.2. Rear Panel of 2712

MAINS VOLTAGE SELECTOR AND FUSE: Voltage selector for operation of the Amplifier from a 100, 115, 127, 150, 220, 240 V (50 to 60 Hz) single phase AC mains supply. To select the correct voltage setting or change the fuse, see section 3.2.3.

MAINS INPUT: Input socket accepting the power cable AN 0010 provided for connection of a mains supply as discussed in section 3.2.3.

5

3. OPERATION

3.1. PRELIMINARY

3.1.1. Rack Mounting

Power Amplifier Type 2712 may be used free standing on its four rubber feet or, with the addition of two Mounting Brackets KS 0023, may be mounted in a 19 inch instrumentation rack. The brackets are available on separate order and bolt into the slots at the front of the Amplifier side panels. The slots are hidden by a plastic cover strip which may be slid out after removing the bottom panel of the Amplifier and levering off the plastic clips at the bottom of each strip.

3.1.2. Ventilation

Forced air cooling of the 2712 enables it to be operated at ambient temperatures up to 40°C (104°F). At higher temperatures protective circuitry automatically shuts down the Amplifier to prevent overheating of its power output transistors.

With approximately 225 VA being dissipated as heat with the 2712 it is important that the flow of cooling air reaching the power output transistors is not interrupted. The ventilation grills on the side and rear panels of the Amplifier should therefore be kept free of obstructions.

3.2. SYSTEM CHECKS AND CONNECTIONS

Before connecting a mains supply the following system checks and connections should be carried out to ensure the correct function and safe operation of the apparatus.

3.2.1. Internal Supply Connections

The 2712 is delivered with the 21 V secondary taps of its mains transformer connected as shown in Fig.3.1. This is suitable for operation of the Amplifier with vibration exciters having a nominal load impedance greater than 0,76 Ω — namely the B & K Vibration Exciter Types 4808 and 4809. For operation with vibration exciters having a nominal load impedance equal to or less than 0,76 Ω — namely the B & K Exciter Body Type 4805 with interchangeable Exciter Heads Type 4811, 4812, 4813 and 4814 — the 19 V secondary tap connections shown in Fig.3.2 should be employed.

The secondary tap connections are a push fit type and are accessible on removing the Amplifier top panel which is fastened by two screws on the rear panel. The wrong secondary voltage will not harm the Amplifier, but will limit its maximum power output capability which depends on exciter load impedance as discussed in section 4.2.

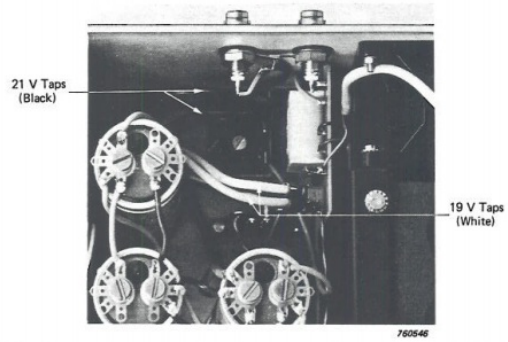


Fig.3.1. Internal connections for operation of the 2712 from its 21 V secondary taps

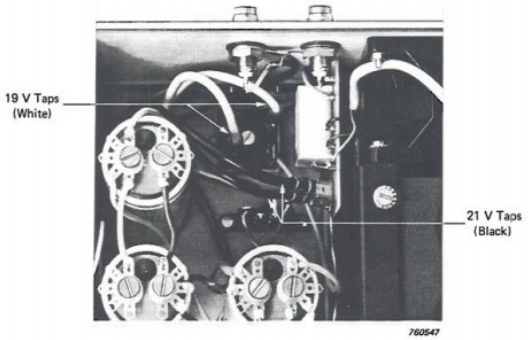


Fig.3.2. Internal connections for operation of the 2712 from its 19 V secondary taps

3.2.2. Exciter Connections

The POWER OUTPUT socket of the 2712 accepts the 3 pin Cannon plug WK-C3-32C (B & K no. JP 0308) provided, and has the pin identities shown in Fig.3.3.

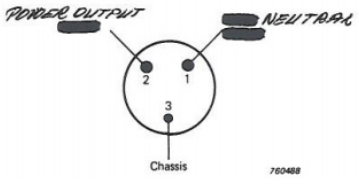


Fig.3.3. 2712 POWER OUTPUT socket (external view)

For connection to Vibration Exciter Type 4808 and to the interchangeable Exciter Heads Type 4811, 4812, 4813 and 4814 of Exciter Body Type 4805, the respective Drive Cable AQ 0095 and AQ 0026 should be used. These are supplied with the Exciters and have the plug connections shown in Figs.3.4 and 3.5 respectively.

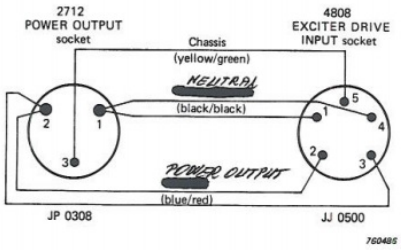


Fig.3.4. Drive Cable AQ 0095. Soldering side of plugs shown

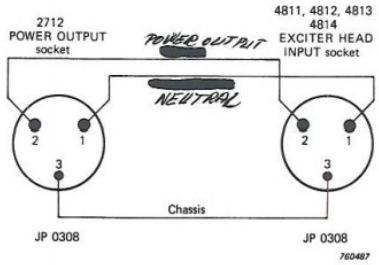


Fig.3.5. Drive Cable AQ 0026. Soldering side of plugs shown

For connection of the 4809 or other types of vibration exciter a suitable drive cable will have to be made up individually. For this purpose pin 1 (live) and pin 2 (neutral) of the POWER OUTPUT socket should be connected to the Exciter using the 3 pin Cannon plug provided.

3.2.3. Mains Supply Connections

The 2712 may be powered from a 100, 115, 127, 150, 220 or 240 V (50 to 60 Hz) single phase AC mains supply. Before connecting the supply the following checks and adjustments should be made.

Voltage setting

The mains voltage setting is displayed in the window of the VOLTAGE SELECTOR on the rear panel of the Amplifier. To select the correct mains voltage setting, press in the knob at the centre of the selector and turn it counter clockwise to release it. Behind the knob are some slots which with the aid of wide blade screwdriver may be used to turn the selector so that its white line points to the correct line voltage setting (± 10%).

Fuse check and replacement

The mains fuse is contained in the knob at the centre of the VOLTAGE SELECTOR. For 100 to 150 V mains supplies the fuse should be a 6,3 A slow blow (B & K order no. VF 0044), whilst for 220 and 240 V supplies it should be a 3,1 A slow blow (B & K order no. VF 0019). Both types of fuse are provided.

Note: Make sure that only fuses with the required rated current and of specified type are used for replacement. The use of mended fuses and of short circuiting of fuse holders should be avoided.

Supply connections

Once the voltage setting and fuse have been checked the mains supply may be connected to the MAINS INPUT socket of the Amplifier using the Power Cable AN 0010 provided. To fit a suitable plug to the cable, see Fig.3.6.

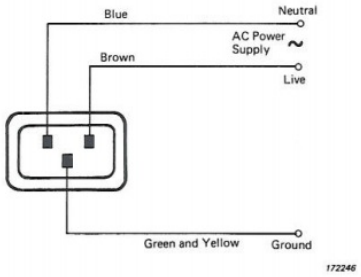


Fig.3.6. Connection of mains supply to MAINS INPUT socket of the 2712

For maximum operating safety it is recommended that the protective (green/yellow) conductor of the cable be connected to a suitable earth, such as the earth contact of a mains outlet socket. The use of an extension cable without protective conductor should be avoided.

3.2.4. Grounding Considerations

When using the 2712 together with other mains operated equipment in complex measurement set-ups, hum pick-up by ground loops may be produced. To prevent this it is necessary to ensure that the instrument set-up is properly grounded. This can be done as follows:

1. Connect the signal ground lines of all the instruments together. This is automatically done through the screens of the cables used to interconnect their input and output sockets.

2. Connect the signal ground line and chassis of the 2712 to the earth of a mains supply. This can be done using the MAINS INPUT socket connections shown in Fig.3.6.

3. Make the necessary adjustments such that the chassis of the other instruments are connected to one and only one of the following points a) mains ground, b) signal ground or c) chassis ground of an instrument which must eventually be returned to mains earth.

4. If a vibration exciter is employed, check that its housing is not grounded by the surface on which it is resting. Also isolate grounded test specimens and measurement transducers from the vibration table of the exciter.

3.3. OPERATING PROCEDURE

After making the system checks and connections given in section 3.2, apply the following setting up procedure to commence operation.

1. Set Power Amplifier controls:

POWER SWITCH	"Off"
AMPLIFIER GAIN:	"Reset" click-stop position
CURRENT LIMIT:	Maximum current limit of Exciter moving coil or of Amplifier, whichever is the smaller. Consult manufacturers data for particular Exciter employed and refer to Figs.4.3 and 4.4 of this Manual.
OUTPUT IMPEDANCE:	"Low" for the best acceleration waveform. "High" for force related tests. See section 4.2.

2. Connect the output of a Vibration Control Generator, such as B & K Type 1023, 1026, 1027 or 1047, to the AC or "DC" INPUT socket on the rear panel of the Power Amplifier. Adjust the output voltage controls of the generator for zero output.

3. On the POWER AMPLIFIER set:

POWER SWITCH:	"Power On" and then wait a few seconds before selecting "Load On" to connect the Amplifier output to the Exciter.
GAIN CONTROL:	Fully clockwise position "10".

4. Set the Vibration Control Generator to the required vibration test frequency and slowly turn up its output voltage level until the required vibration level is obtained on vibration table of the Exciter.

   If the amber DISTORTION lamp lights or the maximum displacement limit of the Exciter is exceeded causing the vibration table to knock against its end stops, then adjust Control Generator output voltage to a lower level in order to resume normal operation.

   For swept frequency vibration testing tune the Exciter Control Generator to the lowest frequency of interest so as to check that the Exciters low frequency displacement limit is not exceeded.

5. To set the Amplifier to Stand-by during the course of a test, return the GAIN CONTROL to "Reset" and the POWER SWITCH to "Power On". At the end of a test set the POWER SWITCH to "Off".

3.4. WARNING LAMPS AND FAULT DETECTION

If one of the red warning lamps light then a fault will have occurred in the system. Under such circumstances a vibration test will be automatically stopped in order to protect the Amplifier and Exciter. To help establish the cause of shut down a list of probable faults is given in Table 3.1.

Warning Lamp	Probable Fault
CURRENT	Input drive level too high for CURRENT LIMIT setting. OUTPUT IMPEDANCE switched before turning GAIN CONTROL to "Reset". Wrong output connections to Exciter.
TEMPERATURE	Overdrive at low frequencies. Wrong output connections to Exciter. Vibration laboratory temperature too high. Force cooling system of Amplifier blocked. Damaged power output transistor.

760054

Table 3.1. Fault detection

If incorrect Amplifier control settings or Exciter connection cause shut-down, turn the AMPLIFIER GAIN control to its "Reset" position and make the necessary adjustments. Normal operation may then be resumed by returning the AMPLIFIER GAIN control to the position used for test.

Should the shut-down occur as a result of an internal fault within the Amplifier or Exciter then the test must be discontinued by setting the AMPLIFIER GAIN control to "Reset" and the POWER switch to "Off". To remedy an internal fault the relevant service Instruction Manual for the Amplifier and Exciter should be consulted.

4. CHARACTERISTICS

4.1. SIGNAL INPUTS

The SIGNAL INPUTS of the 2712 have a minimum input impedance of 10 kΩ. The AC SIGNAL INPUT is for connection of an Exciter Control generator and is capacitive coupled giving a —0,5 dB lower limiting frequency of approximately 15 Hz. The DC SIGNAL INPUT is direct coupled enabling a DC offset voltage to be applied for centring Exciter tables which are statically displaced by heavy vibration test specimens.

The maximum input voltage with the AC and DC SIGNAL INPUTS is 3,4 V peak. With higher input levels the drive signal is clipped causing the DISTORTION lamp of the Amplifier to light.

4.2. POWER OUTPUT

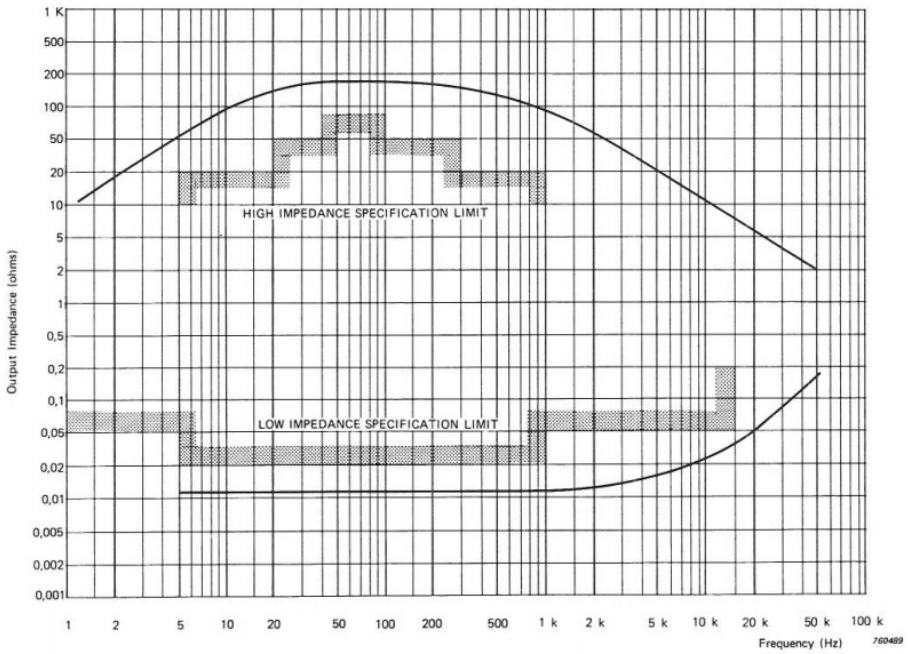


Fig. 4.1. Output impedance of the 2712 as a function frequency and OUTPUT IMPEDANCE switch mode

The POWER OUTPUT of the 2712 is direct coupled. Its output impedance depends on the type of feedback selected with the OUTPUT IMPEDANCE switch and is as shown in Fig.4.1.

With the "Low" impedance mode a fraction of the voltage developed across the moving coil of the Exciter is used as feedback. This gives the Amplifier voltage source characterstics — very low output impedance, constant output voltage with frequency — producing the best acceleration waveform. It is therefore suited for most single Exciter applications as well as for multiple Exciter applications at low frequencies where it is important that Exciters have the same motion.

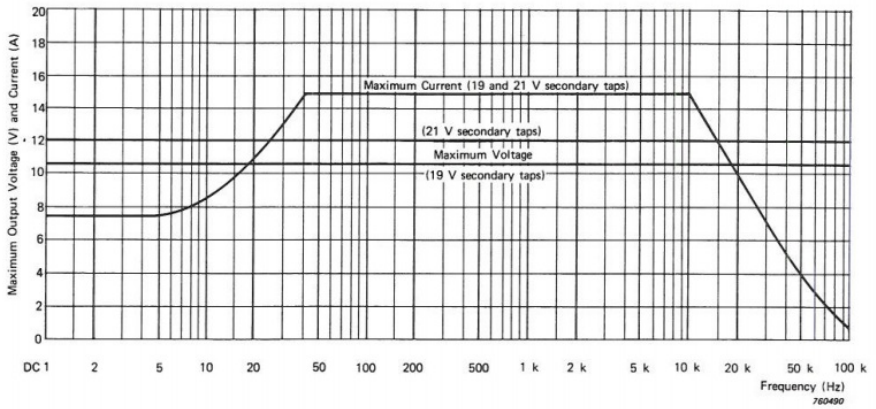


Fig.4.2. Maximum output current and voltage ratings of the 2712 as a function of frequency and mains transformer secondary tap connections

With the "High" impedance mode feedback is proportional to the current flowing in the Exciter moving coil. This gives the Amplifier constant current characteristics — high output impedance, constant output current with frequency — necessary to maintain a constant force with the Exciter despite changes in the vibration test specimen. This is useful for single Exciter fatigue tests and multiple Exciter resonant mode studies on vibration test specimens.

As shown in Fig.4.2 the maximum output voltage rating of the Amplifier depends on the connection of its mains transformer secondary taps (see section 3.2.1), whilst its maximum output current rating depends on frequency. With the 21 V secondary taps the maximum power output is 180 VA which is obtained with Exciters having a nominal load impedance of 0,8 Ω, whilst with the 19 V taps maximum power output is 165 VA which is obtained with Exciters having a nominal load impedance of 0,75 Ω. With other Exciter load impedances the maximum output rating is as shown in Figs.4.3 and 4.4. These are valid at frequencies ranging from 40 Hz up to 10 kHz. At other frequencies the Amplifier's output rating must be derated in accordance with Fig.4.2.

4.3. FREQUENCY RESPONSE

Full power output of 180 VA is available at frequencies beween 40 Hz and 10 kHz. At lower power levels the Amplifier has a useful frequency range extending from DC up to

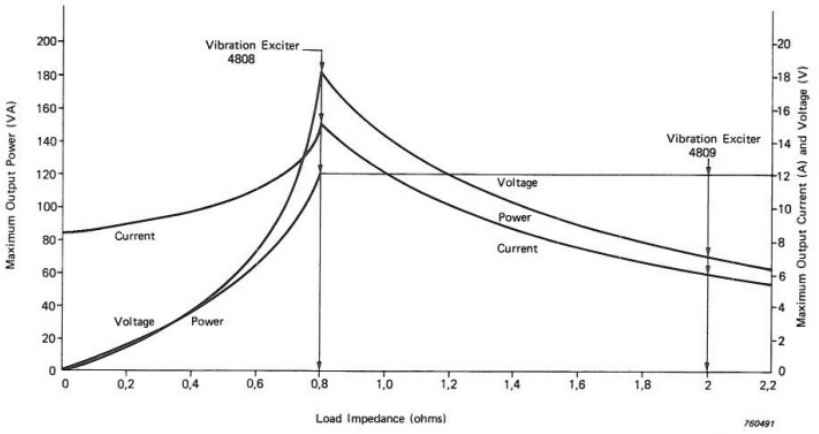


Fig.4.3. Maximum voltage, current and power output ratings of the 2712 as a function of Exciter load impedance, with the 21 V secondary taps of the Amplifiers mains transformer connected. Valid at frequencies ranging from 40 Hz up to 10 kHz

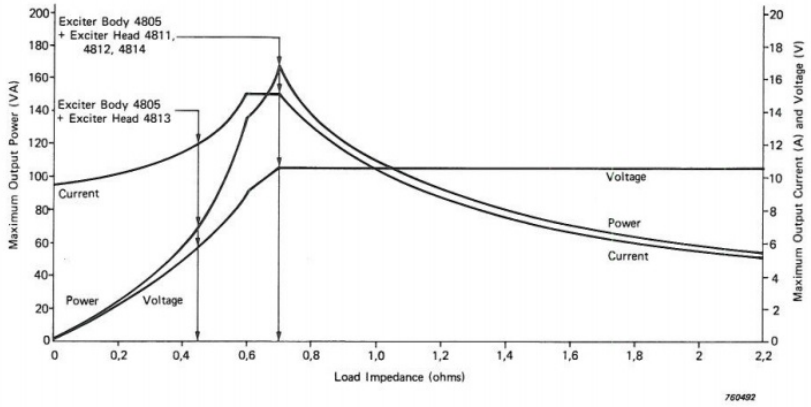


Fig.4.4. Maximum voltage, current and power output ratings of the 2712 as a function of Exciter load impedance with the 19 V secondary taps of the Amplifiers mains transformer connected. Valid of frequencies ranging from 40 Hz up to 10 kHz

100 kHz. This depends on the SIGNAL INPUT socket and OUTPUT IMPEDANCE switch setting as shown by the small signal response curves given in Fig.4.5.

4.4. DISTORTION

The percentage harmonic distortion produced by the 2712 is shown in Fig.4.6. Considering the 180 VA power output rating of the Amplifier the amount of distortion produced is very low. This can be attributed to the generous amount of feedback applied and the use of a direct coupled output.

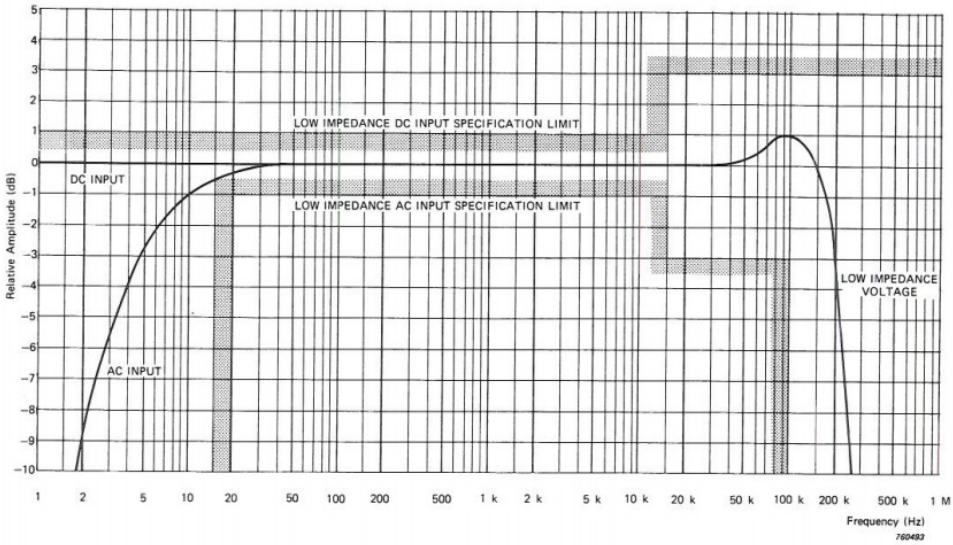


Fig.4.5. Small signal frequency response of the 2712 for power output levels up to 20 VA

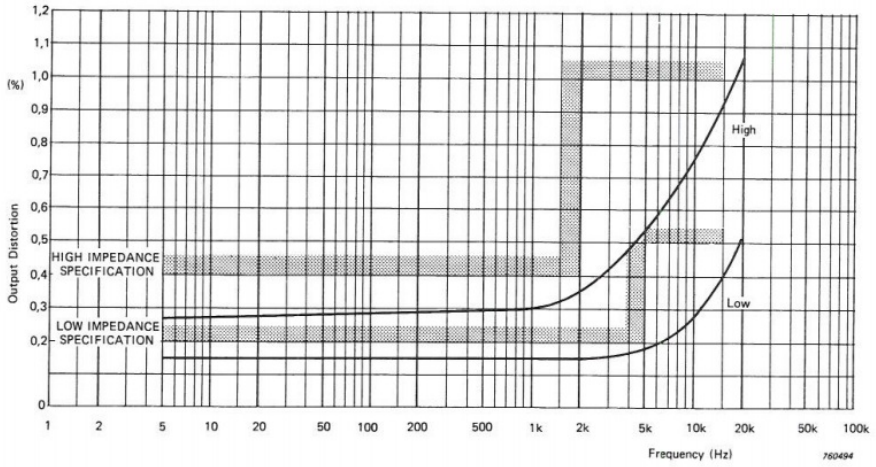


Fig.4.6. Typical percentage harmonic distortion curves for the "Low" and "High" OUTPUT IMPEDANCE modes of the 2712 with 180 VA into a 0,8 Ω load

5. ACCESSORIES

The range of B & K Exciter Control Generators and Vibration Exciters which may be used with the 2712 is shown in Fig.5.1. Full details on the equipment concerned can be obtained from the B & K Short and Main Catalogues which are available on request.

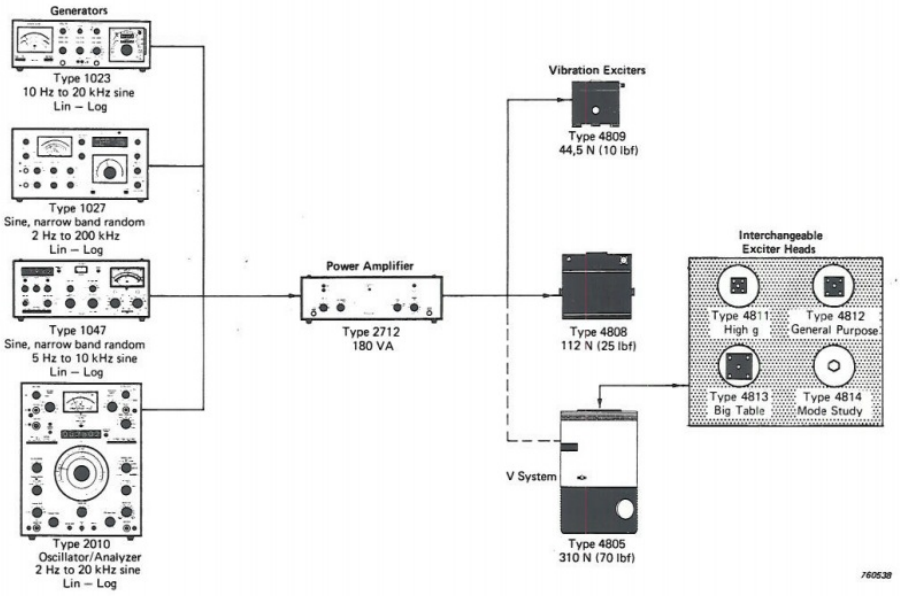
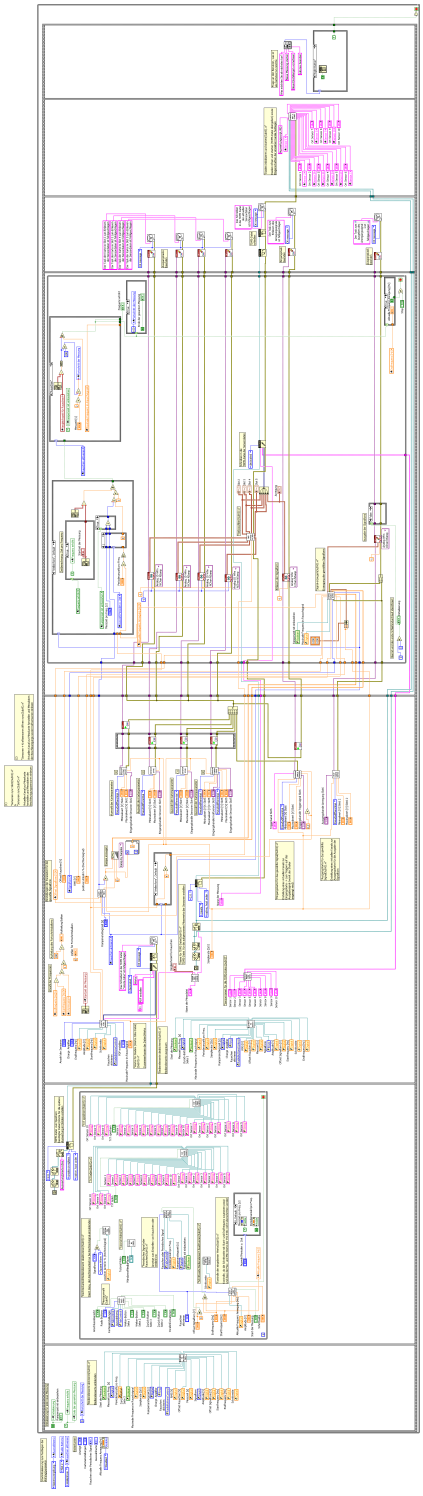


Fig.5.1. The range of B & K vibration test equipment for use with the 2712 Power Amplifier

G. *LabVIEW*-Programm im Blockdiagramm



LabVIEW Programm im Blockdiagramm, Autor: René Pfeifer

H. Ursprünglich verwendete *DIAdem*-Programmteile

Aufteilen der Kanäle nach Frequenzen:

```
'Sensor-Channels nach Frequenzen aufteilen:
'////////////////////////////////////////////////////////////////////////////
DIM frequenz
DIM samples(1000)  'Array für die Samples anlegen.

frequenz = startfrequenz  'Frequenz mit der Startfrequenz initialisieren

for i = 1 to anz_freq  'for-Schleife (4)

  samples(i) = samplerate / frequenz * (anz_period+1)  'Samples zur jeweiligen
                                                       'Frequenz berechnen
  frequenz = frequenz + schrittweite  'Frequenz um die Schrittweite erhöhen

next  'Ende der for-Schleife (4)
```

Berechnung der Samples[3] pro Frequenz, Autorin: Melanie Schulze

```
l = 1  'l: Index, an den die kopierten Channels geschoben werden sollen
m = 0  'm: Zähler, ab wo in den kopierten Channels bis zum Ende alles gelöscht wird
n = 1  'n: Zähler für die Sensor-Channels
o = samples(1)  'o: Zähler, bis wo in den kopierten Channels von Anfang an alles gelöscht wird
p = startfrequenz  'p: Zähler für die Frequenz im Channelnamen
channel_laenge = ChnLength("[1]/[1]")  'Länge der Sensor-Channels bestimmen
```

Variablen und Indizes zur Aufteilung der Messwert-Kanäle, Autorin: Melanie Schulze

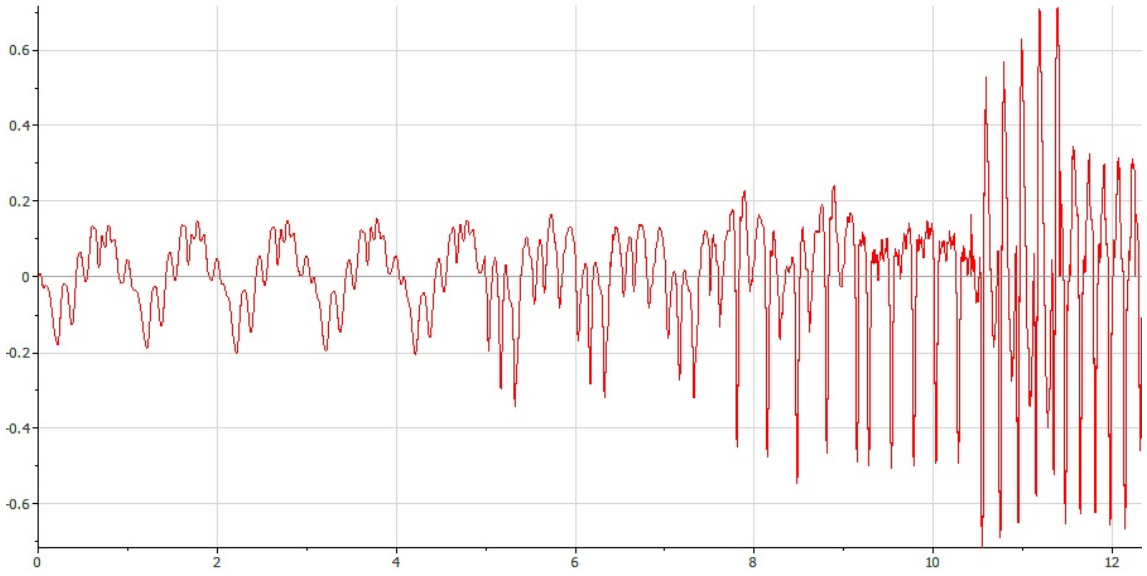
```
for i = 1 to anz_sensoren  'for-Schleife (5)

  for j = 1 to 3  'for-Schleife (6)

    p = startfrequenz

    for k = 1 to anz_freq  'for-Schleife (7)

      Call ChnCopy("[1]/[" + Str(n) + "]", "[2]/" + name_sensoren(i) + ", " + A

      if (k = 1) then  'Wenn k=1 ist, müssen nur am Ende des Channels Einträge gelöscht werden:

        Call DataAreaDel(A,samples(k)+1,channel_laenge,A)

      elseif (k = anz_freq) then  'Wenn k=anz_freq ist, müssen nur am Anfang des Channels Einträge gelöscht werden:

        Call DataAreaDel(A,1,channel_laenge-samples(anz_freq),A)

      else  'In jedem anderen Fall müssen Einträge am Anfang und am Ende des Channels gelöscht werden:

       m = channel_laenge - samples(k) - o
       Call DataAreaDel(A ,channel_laenge-m,m,A)
       Call DataAreaDel(A ,1,o,A)
       o = o + samples(k)

      end if

      p = p + schrittweite
      l = l + 1

    next  'Ende der for-Schleife (7)

    m = channel_laenge - samples(anz_freq) - samples(anz_freq-1) + 1
    o = samples(1)
    n = n + 1

  next  'Ende der for-Schleife (6)

next  'Ende der for-Schleife (5)
'////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Aufteilen der Kanäle nach Frequenzen, Autorin: Melanie Schulze

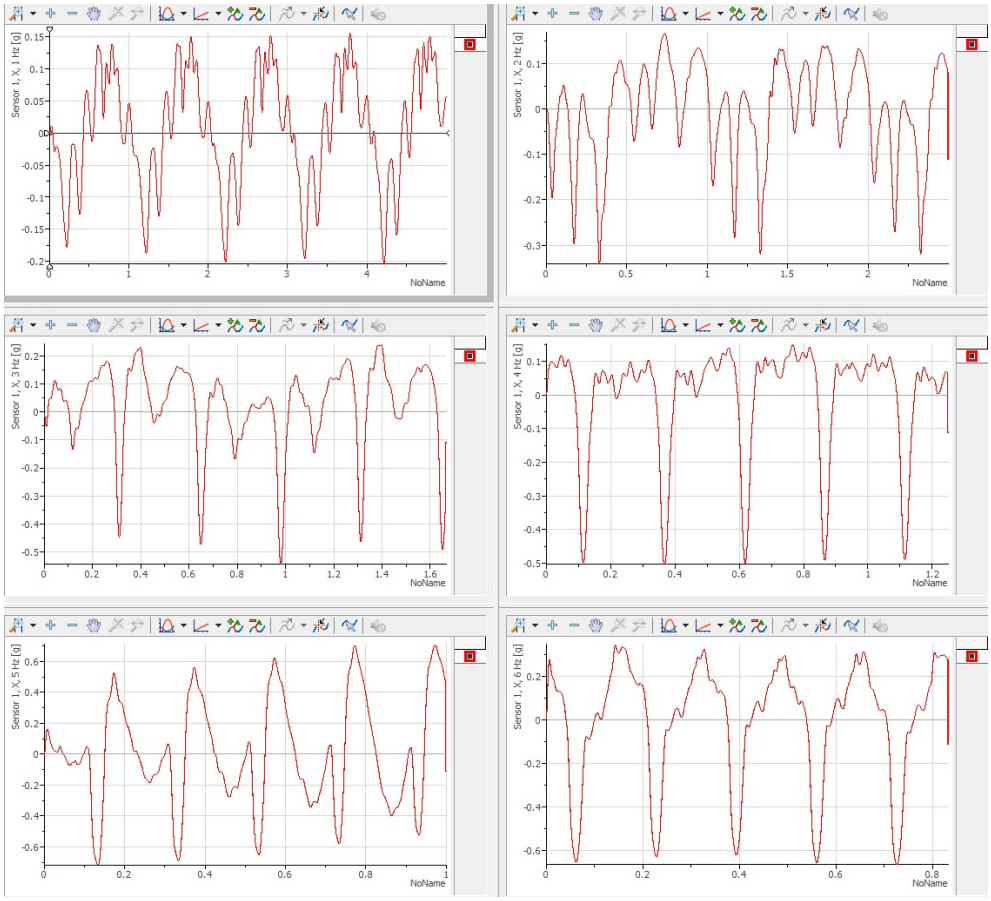
```
name_sensoren(i) + ", " + sensorrichtung(j) + ", " + Str(p) + " Hz"
```

Bedeutung der Abkürzung „**A**“, Autorin: Melanie Schulze

Ergebnis der Aufteilung (erstes Bild: Messwertkanal mit allen Messwerten, zweites Bild: Messwertkanäle mit Messwerten zu jeweils einer Frequenz):



Ursprünglicher Messwertkanal (Beispielmesswerte, 1 Hz – 6 Hz), Autorin: Melanie Schulze



Neu entstandene Kanäle (Messwerte nach Frequenzen aufgeteilt), Autorin: Melanie Schulze

Zusammensetzen der entstandenen Kanäle (werden für die FRF benötigt):

```
'Aufgeteilte Channels wieder zusammensetzen:
'----------------------------------------------------------------
n = 1  'n: Zähler für die zu kopierenden Channels
m = 1  'm: Zähler für die zu löschenden Channels bzw. Index, an den die neuen Channels geschoben werden sollen
l = 2  'l: Zähler für die anzuhängenden Channels

for i = 1 to anz_sensoren  'for-Schleife (12)

  for j = 1 to 3  'for-Schleife (13)

    Call ChnCopy("[2]/[" + Str(n) + "]", name_sensoren(i) + "," + sensorrichtung(j))  'Jeweils ersten Channel kopieren

    for k = 2 to anz_freq  'for-Schleife (14)

      Call ChnConcat("[2]/[" + Str(l) + "]", name_sensoren(i) + "," + sensorrichtung(j))  'Anhängen der jeweiligen Channel
                                                                                          'an die Kopie des ersten
      l = l + 1

    next  'Ende der for-Schleife (14)

    Call Data.Root.ChannelGroups(1).Channels.Remove(m)  'ursprüngliche Messwerte löschen
    'Neue Channels verschieben:
    Call Data.Move(Data.Root.ChannelGroups(3).Channels(name_sensoren(i) + "," + sensorrichtung(j)),Data.Root.ChannelGroups(1).Channels,m)

    n = n + anz_freq
    m = m + 1
    l = l + 1

  next  'Ende der for-Schleife (13)

next  'Ende der for-Schleife (12)
'----------------------------------------------------------------
```

Zusammensetzen der vorher nach Frequenzen aufgeteilten Kanäle, Autorin: Melanie Schulze