Dokumentation

zum Labview-Programm

**„Doktorarbeit Cornelius“**

Programmierer:

Daniel Schuster

Inhalt

[1. Verwendung 2](#_Toc291674768)

[2. Erläuterungen zum Blockdiagramm 3](#_Toc291674769)

[2.1 äußere Sequenz 3](#_Toc291674770)

[2.2 while-Schleife und innere Sequenz 3](#_Toc291674771)

[2.2.1 „Event Structure“ 3](#_Toc291674772)

[2.2.2 „Timed Structures“ 5](#_Toc291674773)

[2.2.3 restlichen Frames 6](#_Toc291674774)

[2.3 SubVIs 7](#_Toc291674775)

[2.3.1 Allgemein 7](#_Toc291674776)

[2.3.2 „Header“ 7](#_Toc291674777)

[2.3.3 „I3252 Comm Interface“ 7](#_Toc291674778)

[2.3.4 „I3252 Edit Setpoint“ 7](#_Toc291674779)

[2.3.5 „I3252 Edit Temperature“ 7](#_Toc291674780)

[2.3.6 „Resistivity Calculation“ 7](#_Toc291674781)

[2.3.7 „System Time in Seconds“ 7](#_Toc291674782)

# 1. Verwendung

Das Programm dient der Steuerung und Abfrage des Thermocontrollers I3252 und dem Auslesen von zwei VC 840 Messgeräten und einem Druckmessgerät Pfeiffer Single Gauge.

Die Messgeräte werden Standardmäßig ausgelesen und erst nach Betätigung des Schalters „Write to File“ in einer Textdatei abgespeichert. Das Eingabefeld „Intervall“ legt den Abstand der Messpunkte fest.

Der Setpoint für den Thermocontroller wird erst nach Betätigung des Tasters „Setpoint START“ übernommen und angesteuert. Er wird ind den EEPROM des Gerätes geschrieben. Die Regelung kann mit Hilfe des Schalters „Standby“ unterbrochen werden. Durch den Taster „Reset STOPP“ kann der Setpoint aus dem EEPROM gelöscht werden.

Die Eingabeflächen „Constant“ und „Thickness“ dienen der Berechnung des spezifischen Widerstands.

Die Eingabeflächen „User“ und „Sample“ werden in den Header der Textdatei übernommen.

Der Indikator „File Path“ zeigt den aktuellen Speicherort der Textdatei an. Dieser wird nach ausschalten des Schalters „Write to File“ gelöscht. Bei erneuten Schreiben muss ein neuer Speicherort gewählt werden. Bei Wahl desselben Speicherorts werden sämtliche bereits gespeicherte Daten gelöscht.

# 2. Erläuterungen zum Blockdiagramm

## 2.1 äußere Sequenz

Die äußerte Sequenz besteht aus fünf Frames. Der erste dient der Initialisierung der Ports. Der zweite dient als Verzögerung um dem Programm genug Zeit zu geben die Ports zu öffnen. Jedoch senden die beiden VC 840 bereits Informationen an den Computer die in einem Puffer zwischengespeichert werden. Im dritten Frame wird dieser Puffer geleert. Somit kommt es zu keinem „Framing Error“. Des Weiteren wird in diesem Frame die Startzeit der Messungen genommen. Im vierten Frame wird die eigentliche Messung und Verarbeitung der Daten durchgeführt. Die while-Schleife wird nur durch Auftreten eines Fehlers oder durch Betätigung des Stopbuttons auf dem Front Panel in der oberen linken Ecke unterbrochen. Im fünften und letzten Frame werden die Ports wieder geschlossen und beim Auftreten eines Fehlers wird das Programm automatisch gestoppt. Falls kein Fehler auftritt wird die Sequenz von vorne abgearbeitet.

## 2.2 while-Schleife und innere Sequenz

### 2.2.1 „Event Structure“

#### 2.2.1.1 Allgemein

Der erste Frame beinhaltet ein „Event Structure“. Dieser reagiert auf insgesamt elf verschiedene Ereignisse. Falls keines dieser Ereignisse auftreten sollte, warte die Struktur eine Millisekunde bevor die „Event Structure“ verlassen wird. Die übrigen Ereignisse treten ein, wenn der entsprechende Kontroller eine Werteänderung erfährt.

#### 2.2.1.2 Event: Setpoint

Dieser Case besteht von außen nach innen aus einer „Stacked Sequence“, einer „for-Loop“ und einer „Flat Sequence“ die wiederum zwei „Case Structures“ enthält. Als erstes wird bei dem Schalter „Setpoint START“ der Blinkmodus aktiviert und bei dem Schalter „Read Temperature“ deaktiviert. Danach folgt das eigentliche setzen des Temperatursollwerts. In der ersten Iteration wird dem Controller I3252 ein Schreibbefehl gesendet. Theoretisch kann der Setpoint in den RAM oder in den EEPROM geschrieben werden. Diese Auswahl erfolgt mit Hilfe des Schalters „RAM“. Da das Schreiben in den RAM –Speicher aber aus diversen Gründen nicht erwünscht ist, wurde es nicht implementiert. Eine nachträgliche Implementierung ist aber leicht möglich, in dem die untere String-Konstante „\*P012“ durch „\*W012“ ersetzt wird. An diesem Befehl wird eine fünfstellige Hexadezimalzahl angehängt, die den Sollwert wiedergibt. Der Sollwert wird vorher mit zehn multipliziert, da das Komma standardmäßig nach der ersten Ziffer festgesetzt wurde (Bsp.: 800 wird gesendet um 80 einzustellen). Dieser Befehlsstrang wird an das Gerät mittels des SubVIs „Comm Interface“, auf welches später noch näher eingegangen wird, weitergegeben. Der durch das Gerät zurückgegebene String wird nicht verarbeitet.

Die zweite Iteration dient dem Auslesen des eingestellten Sollwertes. Wahlweise kann der RAM- oder der EEPROM-Speicher ausgelesen werden. Die eigentliche Verarbeitung des Ausgabestrings erfolgt durch das SubVI „Edit Setpoint“.

Die Implementierung zur Einstellung des zweiten Setpoints erfolgt durch kopieren der ersten beiden Fälle und ersetzen der „1“ im Befehl durch eine „2“ (Bsp.: „\*P012“ zu „\*P022“). Es darf aber nicht vergessen werden, dass dann ebenfalls die Anzahl der Iterationen angepasst werden muss.

Als letztes wird der Blinkmodus des Schalters „Setpoint START“ wieder deaktiviert.

#### 2.2.1.3 Event: Standby

Dieses Ereignis aktiviert bzw. deaktiviert als erstes den Blinkmodus des entsprechenden Schalters und sendet danach den Befehl zum aktivieren („\*D03“) oder zum deaktivieren („\*E03“) des Standby-Modus an den Thermocontroller.

#### 2.2.1.4 Event: Set Command

Der in das Feld „Command shall be” eingegebene Befehl wird an den Thermocontroller übermittelt. Die korrekte Form des Befehls ist im „Communication Manuel“ nachzuschlagen. Das „\*“-Zeichen wurde allerdings schon implementiert und muss somit nicht mit eingegeben werden.

#### 2.2.1.5 Event: Reset

In der ersten Iteration wird der Reset-Befehl an den Thermocontroller gesendet. In Folge wird der gesamte EEPROM-Speicher gelöscht. Deshalb wird in der zweiten Iteration der Setpoint aktualisiert. Dieser Befehl hat keinen Einfluss auf den RAM-Speicher. Die anschließende Wartezeit ist auf vier Sekunden festgesetzt und wurde empirisch ermittelt. Diese Zeit benötigt das Gerät um den Reset durchzuführen und für die Kommunikation wieder zur Verfügung zu stehen.

#### 2.2.1.6 Event: Auto Tune

Es ist festgeschrieben, dass der Setpoint für die „Auto Tune“-Funktion wenigstens 20°C über der aktuellen Temperatur liegt. Deshalb wird zunächst der aktuelle Temperaturwert ausgelesen. Im zweiten Schritt werden zu diesem Wert zwanzig addiert und der Setpoint wird in den RAM-Speicher geschrieben. Das Schreiben in diesen Speicher ist nötig, da die „Auto Tune“-Funktion“ erst nach einem Reset startet. Im dritten Schritt wird der eigentliche Befehl für den „Auto Tune“ gesendet (siehe dazu „Communication Manuel“ S.27). Als vierte und letzte Iteration der for-Schleife folgt der bereits erwähnte Reset-Befehl. Der Temperatursollwert, der im RAM-Speicher eingetragen ist, muss zum Schluss wieder auf 0°C gesetzt werden. Die Zeit für die Ausführung des „Auto Tune“-Befehls wurde empirisch ermittelt und 45 s festgesetzt.

#### 2.2.1.7 Event: Write LOW/HI

Dieses Ereignis setzt die am Analogausgang anliegende Spannung fest. Normalerweise beträgt dieser Spannungswert 0-10 V. Das angeschlossene Netzteil verträgt aber maximal nur 2,3 V. Aus diesem Grund wird der „%HIGH“ auf 23 gesetzt. Somit beträgt die maximal anliegende Spannung nur noch 23% von der maximal möglichen, d.h 2,3V. Ein niedrigerer Wert ist jederzeit möglich, genauso wie ein höherer für den Parameter „%LOW“. Diese werden zunächst nacheinander in den RAM-Speicher geschrieben und anschließend wieder ausgelesen.

#### 2.2.1.8 Event: Set PID

Die Werte für den PID-Regler können sowohl mit Hilfe der „Auto Tune“-Funktion, als auch manuell eingestellt werden. Es wird wiederum zwischen dem Schreiben in den RAM- und dem EEPROM-Speicher unterschieden. Zunächst werden die Werte für „P“, „I“ und „D“, in dieser Reihenfolge, gesetzt und danach die aktuellen Werte ausgelesen.

#### 2.2.1.9 Event: Get Values

Dieses Ereignis hat zur Folge, dass die Werte für „P“, „I“, „D“, „%LOW“, „%HIGH“ und des Setpoints wahlweise aus dem RAM- oder EEPROM-Speicher gelesen werden. Die aktuellen Werte werden nicht nur in die entsprechenden Anzeigefelder übernommen, sondern auch in die entsprechenden Eingabefelder, mit Ausnahme des Temperatursollwerts.

#### 2.2.1.10 Event: RAM

Prinzipiell kann zwischen dem Schreiben in den RAM- oder EEPROM-Speicher des Thermocontrollers gewählt werden. Die entsprechende Schalterposition auf dem Frontpanel zeigt den aktuell verwendeten Speicher an. Die Anzeigewerte werden bei jedem Umschalten aktualisiert und sind im entsprechenden Speicher eingetragen. Die Implementierung ist identisch mit der des Ereignisses „Get Values“. Es gilt zu beachten, dass die Werte für „%LOW“ und „%HIGH“ nur in den RAM-Speicher geschrieben werden können und somit auch nur aus diesem ausgelesen werden können.

#### 2.2.1.11 Event: Set Gauge to Zero

Mit Hilfe der SubVIs „Get Pressure” und „Offset Display” werden die Werte für den alten und den neuen Offset und der durch den zweiten Drucksensor ermittelt Druck ermittelt.

#### 2.2.1.12 Event: Write to File

Sollte der Schalter „Write to File“ aktiviert werden, d.h. Schalterposition oben, so wird zunächst durch das SubVI „Header“ ein Header für die unter „File Path“ definierte Datei angelegt. Alle Werte, die bereits in der entsprechenden Datei stehen, werden dabei gelöscht.

### 2.2.2 „Timed Structures“

Im dritten Frame der inneren Sequenz werden die Werte für Zeitpunkt, Temperatur, Spannung, Strom und Druck ermittelt. Damit die Werte zum gleichen Zeitpunkt aufgenommen werden, werden „Timed Sequences“ eingesetzt. Mit dem entsprechenden Schalter auf dem Front Panel unter dem Tab „Settings“ kann gewählt werden, ob das entsprechende Gerät ausgelesen werden soll oder nicht. Die einzelnen Sequenzen werden parallel abgearbeitet, womit die Prozesszeit von diesem Frame durch der zeitintensivsten Sequenz abhängt. Dies ist in aller Regel das Auslesen der Temperatur, da ein „Delay“ von 500 ms oder größer benötigt wird, um den Controller ordentlich ansprechen zu können.

Die Systemzeit wird durch das SubVI „System Time in Seconds“ in Sekunden umgerechnet und der in dem Parameter „Starttime“ gespeicherte Wert wird abgezogen. Somit werden die Sekunden seit dem Start des Programms gezählt.

### 2.2.3 restlichen Frames

Im zweiten Frame wird das Intervall zwischen den einzelnen Messwerten festgelegt. Dabei wird der Wert des Parameters „Delay“ abgezogen. Dieser Wert beträgt im Normalfall 500 ms und ist somit nicht vernachlääsigbar.

Der Vierte Frame beinhaltet die Umrechnung des Temperatur-Strings in einen Temperaturwert durch das SubVI „Edit Tmperature“, die Berechnung des spezifischen Widerstands durch das SubVI „Resistivity Calculation“ und die Überprüfung des Druckwerts. Auch werden die neu gewonnen Werte in das entsprechende Shift-Register übernommen. Falls der Druckwert einen in dem Parameter „Pressure Limit (mbar)“ festgelegten Grenzwert überschreiten sollte, wird der Thermocontroller automatisch in den Standby-Modus versetzt. Des Weiteren wird der Schalter „Standby“ solange gesperrt, wie der Druck oberhalb des Grenzwertes liegt. Wenn der Druck wieder unter den Grenzwert gesunken ist, wird der Standby-Befehl nicht automatisch aufgehoben, sondern muss vom Benutzer durch Betätigung des entsprechenden Schalters gegeben werden.

Im fünften Frame werden die Diagramme erstellt. Dabei wird jedes Mal von neuem das gesamte Shift-Register als Graph dargestellt. Weiterhin werden die Messwerte in ein 1D-Array geschrieben.

Der sechste und letzte Frame beinhaltet den eigentlich Schreibvorgang der Messwerte in eine Textdatei und die Überprüfung ob ein Fehler vorliegt.

## 2.3 SubVIs

### 2.3.1 Allgemein

Die SubVIs für die Messgeräte VC840 und das Druckmessgerät wurden aus anderen Quellen übernommen.

### 2.3.2 „Header“

In den Header wird zunächst das aktuelle Datum und die aktuelle Zeit übernommen. Die ausgewählte Datei wird dabei überschrieben. Danach wird der Name des Benutzers und der Probe übernommen. Zum Schluss werden die Einheiten in einer Reihe, durch einen Tabulator getrennt, in den Header geschrieben.

### 2.3.3 „I3252 Comm Interface“

Dieses VI stellt die Kommunikation mit dem Thermocontroller her und gibt den Ausgabe-String zurück.

### 2.3.4 „I3252 Edit Setpoint“

Im Ausgabe-String vom Thermocontroller ist an der vierten Stelle die Stelle des Dezimalpunktes verschlüsselt. Dieses Hexadezimalzahl wird herausgefiltert und an ein „ENUM Controller“ übergeben. Dieser weist der Hexadezimalzahl eine Nummer zu, mit welcher ein Wert aus einem Array entnommen wird. Dieser Wert dient als Quotient. Der tatsächliche Wert kann somit errechnet werden.

### 2.3.5 „I3252 Edit Temperature“

Der Temperatur-String, der vom Thermocontroller zurückgegeben wird, wird in eine Dezimalzahl umgewandelt. Manchmal kommt es aus zu Dropouts, die durch dieses Programm herausgefiltert werden. Der letzte Wert wird dabei in einer „Feedback Node“ gehalten unter der Annahme, dass sich der Temperaturwert nicht sprunghaft ändert.

### 2.3.6 „Resistivity Calculation“

Der spezifische Widerstand wird nach folgender Formel ausgerechnet. Dabei entspricht t der Dicke der Probe und C ist eine Konstante.

### 2.3.7 „System Time in Seconds“

Zunächst wird die aktuelle Systemzeit ausgelesen. Danach wird diese in Stunden, Minuten und Sekunden aufgeteilt, alles in Sekunden umgerechnet und letztendlich addiert.