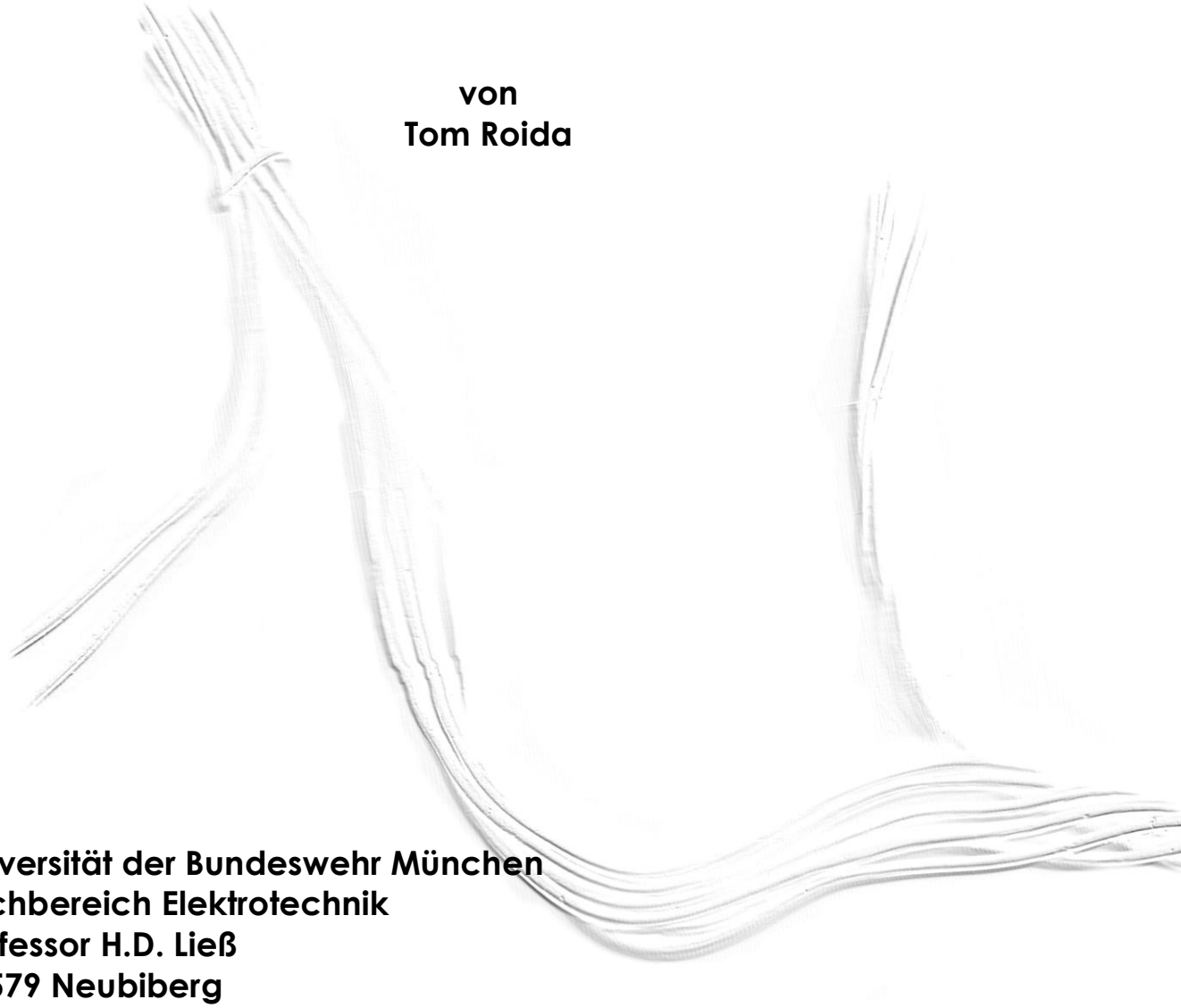
Automatisierte Messung der Wärmeableitung von elektrisch beheizten Leitern

1. Studienarbeit



von
Tom Roida

Universität der Bundeswehr München
Fachbereich Elektrotechnik
Professor H.D. Ließ
85579 Neubiberg

2. Mai 2001

Zusammenfassung

Diese Studienarbeit befasst sich mit der Messung der Wärmeableitung von (Flach-)Kabeln, wie sie z. B. in der Automobilindustrie verwendet werden. Die Wärmeableitung wird einerseits theoretisch betrachtet, andererseits wird eine Methode zur automatisierten Messung dieser Ableitung unter Laborbedingungen beschrieben. Die hierbei verwendeten Ströme liegen bei 10 bis 200 A, die Oberflächentemperaturen der Kabel liegen bei bis zu 100°C. Ein PC steuert mit einer einfachen Software die verschiedenen Laborgeräte an, um ein zu untersuchendes Kabel mit definierten Strömen zu belasten und die Messwerte zu erfassen.

Die Arbeit beginnt mit einer theoretischen Betrachtung, die im Wesentlichen eine Zusammenfassung des Manuskriptes „Bestimmung der Wärmeableitung von elektrisch beheizten Flachleitern“ [1] von H.-D. Ließ darstellt. Sie befaßt sich dabei jedoch nicht speziell mit Flachleitern, sondern mit isolierten Kabeln allgemein.

Im Hauptteil der Arbeit wird die Messung von Leistung und Widerstand eines stromdurchflossenen Leiters beschrieben. Der Leiter befindet sich bei allen Betrachtungen in einem stationären und über seine gesamte Länge gleichen Zustand. Es wird detailliert auf den Meßaufbau sowie auf das zur Messung verwendete Computerprogramm eingegangen. Im Anhang finden sich ein Manual dieses Programms mit Sourcecode-Auszügen und Beispiel-Ausgabedateien.

In einem abschließenden Kapitel wird exemplarisch auf die Auswertung der Meßergebnisse eingegangen. Die Weiterverarbeitung dieser Daten wird anhand von Tabellen und Grafiken demonstriert. Zur vollständigen Auswertung werden genaue Temperatur-Widerstands-Beziehungen der zu untersuchenden Leiter benötigt [2] (deren Ermittlung ist nicht Teil dieser Studienarbeit).

Es ist nicht Ziel dieser Arbeit, vollständige Ergebnisse über bestimmte Leiter zu liefern – statt dessen soll eine Methode geliefert werden, um diese Ergebnisse zu bekommen. Außerdem wird durch die Dokumentation des Computerprogramms die Möglichkeit zur Weiterentwicklung offen gehalten.

Inhalt

Anhang

1. Theorie

1.1 Problemstellung und Ziel der Studienarbeit

„Zur Erstellung eines Berechnungswerkzeuges für die elektrische Belastbarkeit von Leitern ist die Kenntnis des Wärmeableitverhaltens in dem für den Einsatz zulässigen Temperaturbereich notwendig. Aus diesem Grunde ist die Wärmeableitung P(T) von elektrisch beheizten Leitern in Abhängigkeit von der Leitertemperatur T durch praktische Messungen zu bestimmen. Die Ergebnisse werden für einen Vergleich mit den theoretischen Berechnungen benötigt, um gegebenenfalls die ermittelten Zusammenhänge entsprechend korrigieren zu können.“ [1]

Zur Bestimmung einer Beziehung P(T) sind mehrere Schritte nötig:

1. Bestimmung der Temperatur T:

   1.1 Experimentelle Bestimmung einer Beziehung zwischen Temperatur und Widerstand des Leiters, also einer Funktion R(T), wozu folgender Ansatz verwendet werden kann [1]:

$$R(T) = R_0[1 + \alpha_0(T - T_0) + \beta_0(T - T_0)^2] = R_0[1 + \alpha_0\Delta T + \beta_0(\Delta T)^2]$$

   Wobei:

α_0 in 1/K	der lineare Temperaturkoeffizient des Leitermaterials bezogen auf die Referenztemperatur T_0
$ß_0$ in 1/K²	der quadratische Temperaturkoeffizient des Leitermaterials bezogen auf die Referenztemperatur T_0
T_0 in K	die Referenztemperatur, üblicherweise 20°C
ΔT in K	die Temperaturdifferenz $(T - T_0)$

   Es müssen also zwei Materialkonstanten bekannt sein (α_0, β_0) und zusätzlich der Leitungswiderstand R_0. Dabei beziehen sich R_0, α_0, β_0 auf die Temperatur T_0,typischerweise 20°C. Die Konstanten α_0, β_0 können vom verwendeten Leitertyp abhängen und müssen in gesonderten Meßreihen [2] bestimmt werden. Im VDI-Wärmeatlas finden sich folgende (Anhalts-)Werte für Kupfer:

$$\alpha_{0,Cu} = 3{,}81 \cdot 10^{-3} K^{-1}$$

$$\beta_{0,Cu} = 0{,}6 \cdot 10^{-6} K^{-2}$$

$$T_0 = 20°C \quad (= 293{,}15K)$$

Für die Bestimmung der Leitertemperatur über den elektrischen Widerstand ist es unbedingt erforderlich, daß diese konstant über die gesamte Länge ist. Die obige Gleichung läßt sich nach der Leitertemperatur T auflösen:

$$\beta_0(\Delta T)^2 + \alpha_0 \Delta T + (1 - \frac{R(T)}{R_0}) = 0$$

Die Auflösung dieser quadratischen Gleichung liefert:

$$T - T_0 = -\frac{\alpha_0}{2\beta_0} + \sqrt{\frac{\alpha_0^2}{4\beta_0^2} + \frac{1}{\beta_0}(\frac{R}{R_0} - 1)}$$

Da der Wert $ß_0$ sehr klein ist (siehe oben) und typischerweise nicht sehr genau bestimmt werden kann, wird diese Gleichung sehr fehleranfällig. Eine Reihenentwicklung[1] des Wurzelterms führt zu folgendem stabileren Ausdruck [1]:

$$T - T_0 = \frac{1}{\alpha_0}(\frac{R}{R_0} - 1)(1 - \frac{\beta_0}{\alpha_0{}^2}(\frac{R}{R_0} - 1))$$

1.2 Bestimmung des Leiterwiderstands R:

Durch Messung des Spannungsabfalls am Leiter bei einem Strom I kann der Leiterwiderstand nach dem ohmschen Gesetz bestimmt werden:

$$R = \frac{U}{I}.$$

Nun kann bei bekanntem T_0 und R_0 der Widerstand in eine Leitertemperatur umgerechnet werden. Sind diese Werte jedoch nicht bekannt (und davon ist auszugehen, wenn nicht die erste Messung mit exakt dem gleichen Leiter durchgeführt wurde), müssen sie aus einer gegebenen Reihe von anderen Meßwerten extrapoliert werden. (siehe dazu Kapitel 3.2 „Weiterverarbeitung der Daten“)

2. Bestimmung der vom Leiter abgegebenen Leistung P:
Durch Messung des Spannungsabfalls am Leiter bei einem Strom I kann die Leistung des Leiters als $P = U \cdot I$ bestimmt werden.

Mit diesen Ergebnissen kann eine Beziehung zwischen Leitertemperatur und seiner elektrischen Leistung hergestellt werden, und es läßt sich ein Wärmeübertragungskoeffizient des Leiters bestimmen[1]. Dieser beschreibt das Verhältnis zwischen der Wärmeverlustleistung und der zugehörigen Leitertemperatur.

[1] Abbruch der Reihenentwicklung nach dem zweiten Glied: $\sqrt{1+x} = 1 + \frac{x}{2} - \frac{1}{8}x^2$

Die in den folgenden Kapiteln beschriebene Meßmethode mit dem Computerprogramm CPOW befaßt sich mit der Leistungs- und Widerstandsmessung eines Leiters bei einem gemessenen Strom, also den unter (2.) aufgeführten Teil.

Dabei wird im folgenden immer angenommen, daß sich der Leiter in einem stationären Zustand befindet und daß der Leiter frei in der Luft (mit konstanter Umgebungstemperatur) hängt.

Die Leiterströme I werden in etwa gleich großen Intervallen vom Wert 0 bis zu einem für den Leiter üblichen Maximalwert angehoben und im Anschluß daran in den gleichen Intervallen wieder bis auf 0 zurückgebracht.

Der Meßstrom bewirkt die Eigenerwärmung des Leiters. Die Messung erfolgt bei Raumtemperatur[1].

1.2 Thermodynamische Betrachtung des Leiters

Die von einem (isolierten) Leiter an die Umgebung abgegebene Leistung setzt sich zusammen aus:

- Thermischer Strahlung;
- Wärmeübergang, beispielsweise
  - konvektive Wärmeübertragung an Luft bei einem „hängenden“ Leiter;
  - Wärmeübertragung an den Leiterenden (Befestigungen, Anschlüsse);
  - Wärmeübertragung an Kontaktstellen zu anderen Materialien wie z. B. beim Aufliegen des Leiters am Boden.

Für die Aussagekraft der Meßergebnisse ist es unbedingt erforderlich, daß sich der Leiter vor jeder Messung in einem stationären Zustand befindet, d.h. daß die Erwärmungs- bzw. Abkühlungsphase größtenteils abgeschlossen ist und sich ein zeitlich konstanter Wärmestrom einstellt.

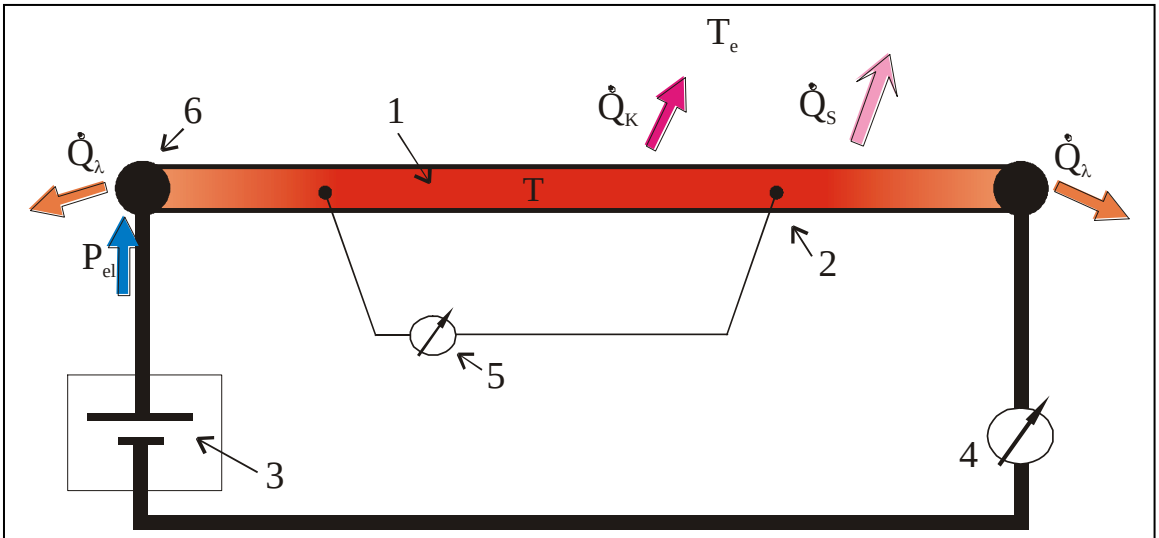


Abb. 1.1 Thermodynamische Einflüsse

1) Kabel, erhitzt auf Temperatur T, am Rand kälter
2) Auf das Kabel angelötete Meßkabel, im temperaturkonstanten Bereich
3) Stromquelle
4) Amperemeter, Messung des Stromes I
5) Voltmeter, Messung der Spannung U an der Leitung
6) Kontaktstelle zum Stromkabel der Stromquelle

$\dot{Q}_\lambda$ Wärmeleitung an der Kontaktstelle

$\dot{Q}_S$ Durch Wärmestrahlung an die Umgebung abgegebene Leistung

$\dot{Q}_K$ Konvektiver Wärmestrom Luft

P_{el} An das Kabel abgegebene elektrische Leistung

T Kabeltemperatur im mittleren Bereich, konstant

T_e Umgebungstemperatur

Die thermodynamische Bilanz am Leiter lautet:

$$\frac{\partial U}{\partial t} = \sum \dot{Q} + \dot{W}$$

wobei $\frac{\partial U}{\partial t} = 0$ sein soll, da nur stationäre Zustände untersucht werden sollen. Außerdem ist in diesem Fall $\dot{W} = P_{el}$, also die dem System „Leiter“ zugeführte Leistung. Es bleibt also:

$$P_{el} - \dot{Q}_S - \dot{Q}_K - \sum \dot{Q}_\lambda = 0$$

Für jede Leitertemperatur T existiert ein solches Gleichgewicht mit einer zugeführten Leistung $P_{el}(T)$.

Gemessen wird, wie bereits in Kapitel 1.1 beschrieben, der Spannungsabfall zwischen den Meßkabeln. Anfang und Ende des Leiters werden wegen der niedrigeren Temperatur nicht in die Messung miteinbezogen.

Nachdem mit der Stromquelle ein bestimmter Stromfluß durch den Leiter erzeugt wird, muß das Einstellen eines stationären Zustandes abgewartet werden. Erreicht wäre dieser, sobald sich die Leitertemperatur nicht mehr ändert. Da sich diese jedoch asymptotisch an eine Endtemperatur angleicht und somit nie echt erreicht wird, muß unter Beachtung des Meßfehlers eine Abschätzung vorgenommen werden. (siehe Kapitel 2.5 Fehlerbetrachtung)

1.3 Elektrische Betrachtung des Leiters

Die elektrische Schaltung des Meßaufbaus läßt sich mit einfachsten Grundgleichungen beschreiben:

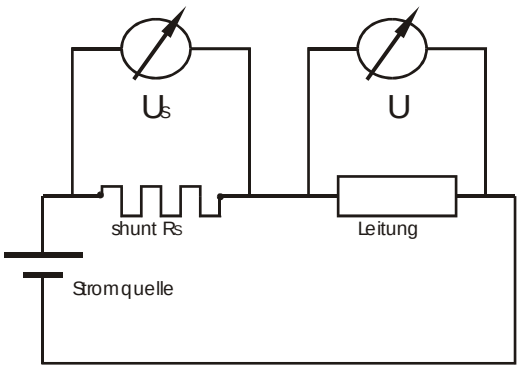


Abb 1.2 Schaltplan der Meßvorrichtung

Spannungsabfall am Shunt: $U_S = I \cdot R_S$

Eingespeiste Leistung: $P_{el} = U \cdot I = \frac{U \cdot U_S}{R_S}$

2. Praktische Durchführung der Messungen mit dem Computerprogramm CPOW

2.1 Schematische Beschreibung des Meßaufbaus

2.1.1 Verwendete Geräte

1. Stromquelle: EA-PS 9018-300 Laboratory Power Supply, Hersteller: Firma „Elektro-Automatik“, Viersen.
   - Optionale Erweiterung: IEEE488.2 Schnittstelle
   - Ausgangsspannung: 0..18V DC
   - Ausgangsstrom: 0..300A
   - Strom-Stabilität bei 0..100% Last: ≤0,15%
   - Restwelligkeit: ≤ 160mA p-p
   - Ausregelzeit: 1ms
2. Digitalmultimeter: DMM 6001, Hersteller: Firma PREMA GmbH, Mainz. (Im weiteren „DMM“ genannt.)
   - IEEE488.2 Schnittstelle
   - Temperaturmessungen mit PT100 möglich, Anzeige in °C
   - Optionale Erweiterung: Meßstellenumschalter 600/01 mit 10 Kanälen
   - Meßprinzip Gleichspannung: Vollintegrierendes Rampenverfahren
   - Offsetkorrektur zur Kompensation von Thermospannungen möglich
   - Meßbereiche (unter anderem): ±0,2V / ±2V / ±20V Gleichspannung
   - Eingangswiderstand: ≥ 10MΩ
   - Fehlergrenzen Gleichspannung:

BEREICH	%Az.	%max. Az
200mV	0,002	0,0007
2V	0,002	0,0002
20V	0,002	0,0002

3. Computer zur Meßdatenaufnahme
   - PC mit MS-DOS 6.2
   - IEC-Bus-Schnittstellenkarte: GPIB Interface card "CIO-PC2A"[1] von der Firma „Measurement Computing"; IEEE 488.2 Standard
   - Programm CPOW („Cable-Power") von Patrick Mack und Tom Roida, Version 1.65
   - Ausgabe der Daten durch das Programm CPOW als ASCII-Datei
4. Shunt-Widerstand: Je nach Einsatzbereich werden unterschiedliche Widerstände gewählt, um einen möglichst großen Spannungsabfall zu erhalten.

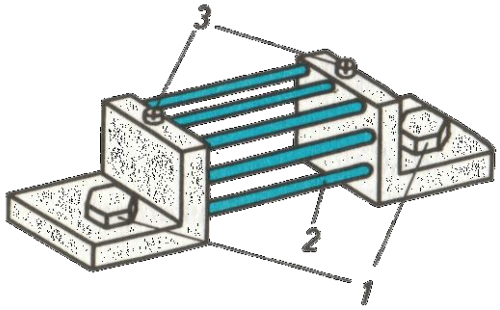


Abb. 1.3 Shunt
1 Stromanschlußschrauben
2 Temperaturunabhängiger, geeichter Widerstand
3 Spannungsabgriffe (Anschluß für Millivoltmeter
Quelle: [1]

2.1.2 Anschlüsse und Verdrahtung

Am Leistungsausgang der Stromquelle sind der Shunt-Widerstand und der zu vermessende Leiter in Reihe verbunden (Siehe Abb. 1.2). Am Shunt-Widerstand sind zwei Meßleitungen mit Kanal 1 des DMM verbunden.

Der zu vermessende Leiter ist an den Enden an eine Kupferplatte angelötet, um einen optimalen Stromfluß zu gewährleisten.

Mindestens 15 cm von den Leiterenden entfernt sind die Meßleitungen für den Spannungsabfall am Leiter angelötet. Dadurch wird nur im konstanten Temperaturbereich gemessen [1] (siehe auch Abb. 1.1 Thermodynamik). Die Länge des Leiters zwischen diesen Lötstellen beträgt ca. 10m und ist auf ±10mm genau bestimmt. Der Leiter hängt horizontal im Raum und liegt an keiner Stelle auf. Wird ein Flachleiter vermessen, so wird darauf geachtet,

[1] Es ist auch möglich, eine andere Karte der Firma „Measurement Computing" zu benutzen, die kompatibel zum CBCONF-Programm ist. In diesem Fall muß diese nur richtig konfiguriert werden (siehe Handbuch der Karte), das Programm CPOW muß nicht modifiziert werden. Bei der Verwendung einer Interfacekarte eines anderen Herstellers, zum Beispiel von „National Instruments", muß das Programm jedoch neu compiliert und gegebenenfalls angepaßt werden.

daß er nicht verdreht ist, um überall eine gleichmäßige Luftkonvektion zu ermöglichen. Der Flachleiter wird horizontal zum Erdboden aufgehängt.

Die Meßleitungen des Leiters sind mit Eingangskanal 3 des DMM verbunden.

Am Front-Eingangskanal (Channel „-“) des DMM wird ein PT100-Thermoelement angeschlossen. Das Element wird auf der Leiteroberfläche fixiert, um die Oberflächentemperatur zu messen. Dies sollte wegen der Störung der Luftkonvektion möglichst nicht im Bereich zwischen den Meßanschlüssen liegen.

Die Temperaturmessung mit dem PT100-Element am Front-Eingangskanal dient nur als Kontrollmöglichkeit für das Programm CPOW, um die zulässige Leitertemperatur bei einer falschen Benutzereingabe nicht zu überschreiten (siehe Kapitel 2.3 Sicherheit). Unter Umständen kann auf den Sensor komplett verzichtet werden. Es wird (automatisch) mit Vierdraht-Methode gemessen.

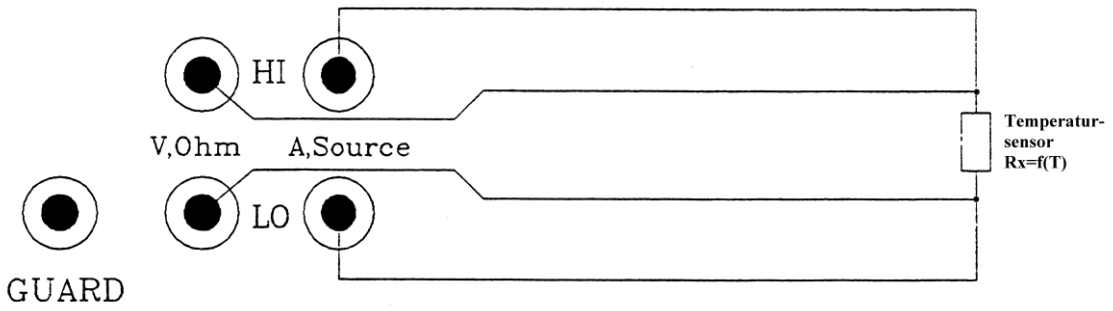


Abb. 2.1 Vierdraht Temperaturmessung

Quelle: Handbuch des PREMA Digitalmultimeter DMM6001

Ist nur ein Zweidraht-Fühler vorhanden, so kann jeweils der Hi-Ohm mit dem Hi-Source Eingang sowie der Lo-Ohm mit dem Lo-Source Eingang kurzgeschlossen werden. Eine gesonderte Offsetkorrektur des Temperaturfühlers ist möglich (siehe Handbuch), jedoch nicht unbedingt erforderlich.

Zusätzlich besteht die Möglichkeit, über die noch freien Kanäle weitere Meßdaten zu sammeln (siehe auch Kapitel 3 Ausblick). In der CPOW Programmversion 2.91 können bis zu fünf weitere PT100-Elemente an die Kanäle 5 bis 9 mit der gleichen Verdrahtung wie am Front-Eingangskanal angeschlossen werden. (Siehe auch Anhang I.)

Das Vertauschen der einzelnen Kanäle am DMM führt zu keinen bzw. zu sinnlosen Ergebnissen. Das Vertauschen von (+) und (-) Leitungen führt zu falschen Vorzeichen der Werte in der Ausgabedatei. Das PT-100 Element darf nicht falsch angeschlossen werden, da das DMM sonst überhaupt kein (Temperatur-)Ergebnis liefert.

2.2 Schematische Beschreibung des Computerprogramms

Alle Geräte sind über den IEC-Bus miteinander verbunden. Der PC agiert als Controller. Nach dem Programmstart (bzw. Start der Messung) sind keine manuellen Bedienungsschritte mehr erforderlich. Der Computer führt während einer Messung folgende Schritte aus:

1. An der Stromquelle wird ein bestimmter Strom vorgegeben.
2. Bis sich die Temperatur des Leiters nicht mehr wesentlich ändert wird eine bestimmte Zeit abgewartet (siehe Anhang I).
3. Nacheinander werden in einem vom Benutzer vorgeschriebenen Zeitabstand (mind. 25 Sekunden) n Meßwerte am DMM ausgelesen und in einer vom Benutzer benannten Ausgabedatei gespeichert.
4. Ein neuer Strom wird vorgegeben.
5. Weiter mit 1. oder nach Durchlaufen aller Ströme weiter mit 6.
6. An der Stromquelle wird ein Strom von 0A gesetzt.

- Zusätzlich wird alle 12 Sekunden die Leitertemperatur mit dem PT100-Element ausgelesen. Liegt sie höher als die vom Benutzer vorgegebene Maximaltemperatur, so wird an der Stromquelle ein Strom von 0A gesetzt und das Programm beendet sich aus Sicherheitsgründen selbst.

Welche Stromwerte nacheinander gesetzt werden, legt der Benutzer bei Programmstart fest. Das Programm fragt nach

- minimalem und maximalem Stromwert;
- einer Schrittweite;
- ob auf- oder absteigend gemessen werden soll (oder beides).

Für eine genauere Beschreibung siehe Anhang I.

2.3 Sicherheit

Die Stromquelle ist mit ihrer Leistung in der Lage, nahezu jedes beliebige Kabel stark zu erhitzen. Bei falscher Bedienung oder aber bei einem Programmfehler könnte es leicht zu einem Brand kommen. Da das Programm konzipiert ist, um lange Zeit selbständig zu arbeiten, müssen folgende „Sicherheitsmechanismen“ installiert sein:

- Der PT100 Sensor mißt die Leitertemperatur und das Programm vergleicht diesen Wert permanent mit den zulässigen Werten. Dies bildet jedoch keinen hundertprozentigen Schutz,denn wenn das Programm aus irgendeinem Grund abstürzen würde, so bliebe die Stromquelle in genau dem Zustand, in dem sie sich kurz vor dem Absturz befand – unter Umständen also mit einem sehr hohen Strom.
- Durch eine Lötverbindung mit einem niedrigschmelzenden Lot kann ein Öffnen der Lötstelle bei hohen Temperaturen erreicht werden, falls diese (leicht) auf Zug belastet wird. Dazu muß bei der Vorbereitung der Messung der zu untersuchende Leiter aufgetrennt und mit einem solchen Lot wieder verbunden werden. Das Eigengewicht

des Leiters reicht zum Öffnen im kritischen Fall aus. Als Lot wird eine Sn50PbCd18-Legierung mit einem Schmelzpunkt von 145°C verwendet. Bei der Verarbeitung sind aufgrund des Cadmium-Gehalts besondere Schutzmaßnahmen notwendig.

2.4 Durch die Messung gelieferte Daten

In „Rohform“ stehen mit dem oben beschriebenen Meßaufbau zwei Wertepaare pro Einzelmessung zur Verfügung:

1. Spannungsabfall am Shunt
2. Spannungsabfall am Leiter

Die Einzelmessungen werden in bestimmten Zeitintervallen vorgenommen, so daß zu jedem Wertepaar noch eine (Meß-)Zeit bekannt ist.

Das Programm rechnet die oben genannten Meßergebnisse um und schreibt insgesamt folgende Werte pro Einzelmessung in die Ausgabedatei:

- **Zeit** seit Beginn der Messung in Sekunden
- **Gemessene Temperatur** am Leiter in °C
- **Vorgegebener Strom** in A
- Von der Stromquelle **gemessener Strom** in A
- Anhand des Spannungsabfalls am Shunt **errechneter Strom** in A
- Anhand des Spannungsabfalls am Leiter und anhand des errechneten Stroms abgegebene **Leistung** des Leiters in W
- **Spannungsabfall am Shunt** in V
- **Spannungsabfall am Leiter** in V
- Ohmscher **Widerstand des Leiters** in Ohm

2.5 Fehlerbetrachtung

Die folgende Auflistung von möglichen systematischen und zufälligen Fehlern soll zum einen für die Angabe von Toleranzgrenzen im streng mathematischen Sinn dienen, zum anderen soll sie einen Teil der Erfahrungen vermitteln, die H.-D. Ließ mit seinen Mitarbeitern und Studenten während der ersten Messungen gemacht hat.

2.5.1 Fehler bei der Berechnung des Leiterwiderstandes

Das Ergebnis der Messungen ist der Leiterwiderstand R über dem gemessenen Strom I.

Die Formel lautet: $R = U / I$

Der Strom I errechnet sich dabei als: $I = U_S / R_S$

Also insgesamt: $R = \frac{U \cdot R_S}{U_S}$.

Spannungsabfall im Leiter

Das DMM liefert Spannungsmessungen über den gesamten Bereich mindestens mit einer Genauigkeit[1] von:
±(0,002% der Anzeige + 0,0002 bis 0,0007 % der maximalen Anzeige)

Beispiel
Bei einem 10m langen Leiter, das mit bis zu 200A belastet werden kann, ergeben sich folgende Werte:

I	U	Anzeigebereich DMM	Fehler	Relativer Fehler $\Delta U / U$
5A	0,1009834V	0,2V	±0,0003V	0,3%
10A	0,200971V	2V	±0,002V	1,0%
20A	0,404417V	2V	±0,002V	0,5%
150A	3,5311V	20V	±0,011V	0,3%

Spannungsabfall im Shunt

Das DMM liefert Spannungsmessungen bei einem Meßbereich bis 200mV mindestens mit einer Genauigkeit[1] von:
±(0,002% der Anzeige + 0,0007 % der maximalen Anzeige)

Beispiel
Shunt mit Spannungsabfall von 30mV bei 150A:

I	U	Anzeigebereich DMM	Fehler	Relativer Fehler $\Delta U_S / U_S$
5A	0,001V	0,2V	±0,00014V	14%
20A	0,004V	0,2V	±0,00014V	04%
75A	0,015V	0,2V	±0,00017V	01,1%
150A	0,030V	0,2V	±0,00020V	00,7%

Zu dem extrem großen Fehler muß gesagt werden, daß ein Leiter je nach Bereich mit verschiedenen Shunt-Widerständen vermessen werden sollte, um bei niedrigen Strömen zu einem relativ hohen Spannungsabfall zu kommen!

Zu obigem Beispiel: um den Fehler $\Delta U / U \leq 1\%$ zu halten, dürfte der Strom bei diesem Shunt nicht geringer als 85A sein! Soll bei geringeren Strömen gemessen werden, muß der Shunt ausgetauscht werden.

[1] Um die angegebene Genauigkeit mit dem Multimeter zu erreichen, müssen bestimmte Bedingungen eingehalten werden, z. B. Kalibrierung, Offsetkorrektur, Einschaltdauer. Siehe Handbuch.

Widerstand Shunt

Auf den üblichen Shunt-Widerständen ist eine Fehlerklasse aufgedruckt.

Beispiel

Der in den obigen Beispielen verwendete Shunt-Widerstand hat eine Fehlerklasse von 0,5 , d.h. $\Delta R / R = 0{,}5\%$.

Fehlerfortpflanzung

Insgesamt ergibt sich nach der Gauß'schen Fehlerfortpflanzungsregel der relative Fehler :

$$\frac{\Delta R}{R} = \sqrt{(\frac{\Delta U}{U})^2 + (\frac{\Delta R_S}{R_S})^2 + (\frac{\Delta U_S}{U_S})^2}$$

Einige konkrete Zahlenwerte als Beispiel:

Verwendeter Leiter: Flachbandkabel, bis max. 200A belastbar.

Strom I	$\Delta U / U$	Verwendeter Shunt	Fehlerklasse Shunt $\Delta R / R$	$\Delta U_S / U_S$	Gesamter relativer Fehler $\frac{\Delta R_{Cable}}{R_{Cable}}$
≈ 20A	0,5%	30mV bei 150 A	0,05%	4,0%	4,0%
≈ 150A	0,3%		0,05%	0,7%	0,8%
≈ 150A	0,3%	60mV bei 400 A	0,50%	0,8%	1,0%

2.5.2 Zusätzliche Fehlerquellen

Thermospannungen

Thermospannungen treten immer dort auf, wo unterschiedliche Metalle bei unterschiedlichen Temperaturen in Kontakt treten [1]. Kritische Stellen sind demnach die Kontakte zwischen Kabel und Meßleitung.

Anmerkungen:

1.) Thermospannungen an der Meßverbindung zum Shunt-Widerstand sind unkritisch, da dieser seine Temperatur nie ändern sollte (siehe dazu aber auch Kapitel 2.5.2) und außerdem vor Beginn der Messung eine Offsetkorrektur des DMM am Shunt vorgenommen wird. Dadurch werden vorhandene Thermospannungen ausgeglichen.

2.) Die Anschlußleiste am DMM ist laut Hersteller „Thermospannungsarm". Genauere Angaben dazu finden sich jedoch nicht im Handbuch.

3.) Da Offsetkorrekturen für Spannungsmessungen am DMM nicht einzeln für jeden Eingangskanal vorgenommen werden können, ist es denkbar, daß am Kanal für den Spannungsabfall am Leiter eine „falsche" Spannung anliegt bzw. angezeigt wird. Diese kann jedoch aufgrund von folgenden Annahmen vernachlässigt werden:

   a) Erfahrungen haben gezeigt, daß im kalten Zustand bei I=0A die Anzeige immer weit unter 1mV lag.

   b) Sollte sich dies bei erhitztem Leiter ändern, dann nicht mehr als um einige mV [1]. Bei stark erhitztem Leiter beträgt der Spannungsabfall aber typischerweise einige Volt. Der Fehler liegt also maximal im 1%-Bereich.

Einstellzeit der Leitertemperatur

Die von H.-D. Ließ aufgestellte Formel zur Berechnung einer Einstellzeit für die Leitertemperatur [1] wurde in das Programms CPOW aufgenommen. Die in der Formel vorkommende Konstante α_m, der mittlere Wärmeübergangskoeffizient für Luft, wurde aus Erfahrungswerten ermittelt und großzügig (nach unten) abgeschätzt. Es ist möglich, daß er nicht für alle Kabeltypen ausreichend dimensioniert ist. Dies würde eine manuelle Korrektur des Programms erfordern. Siehe dazu auch Anhang I.

Temperaturverlauf über den Leiter

Bei unterschiedlicher Luftkonvektion im Meßraum kann es zu lokalen Temperaturunterschieden im Leiter kommen.

Zeitlich versetzte Messung

Bei dem oben beschriebenen Meßaufbau wird nur ein Multimeter verwendet, um den Spannungsabfall sowohl am Shunt als auch am Leiter zu messen. Dies geschieht mit einer zeitlichen Versetzung von einigen Sekunden. Selbst unter der Annahme, daß sich bereits eine stabile Leitertemperatur eingestellt hat und sich diese innerhalb so kurzer Zeit nicht ändert (also R=const.), so könnte es doch zu falschen Ergebnissen führen, falls an der Stromquelle kurzzeitige Stromschwankungen auftreten. Laut Datenblatt kann dies jedoch ausgeschlossen werden.

Änderung der Raumtemperatur

Falls sich die Umgebungstemperatur während der Messung (unbekannt) ändert[1], kann es zu Verfälschungen der Meßergebnisse kommen. Als Abhilfe muß die Umgebungstemperatur permanent aufgezeichnet[2] und in die Auswertung des Widerstandes miteinbezogen werden.

Erwärmung des Shunt

Erwärmt sich der Shuntwiderstand, so macht sich dies in einem (leichten) Ansteigen des elektrischen Widerstandes bemerkbar. Dies kann den Anschein erwecken, daß sich der Leiter weiter erhitzt und noch keinen stationären Zustand erreicht hat.

Falls der Shunt durch Ventilatoren gekühlt wird, so ist dabei auf ausreichenden Abstand zum Leiter zu achten.

Verdrehte Leiter

Werden Flachleiter vermessen, so ist deren Ausrichtung im Raum von Bedeutung, da durch sie die Luftkonvektion beeinflußt wird.

Rundungsfehler im Computer

Das Computerprogramm rechnet mit REAL und EXTENDED Variablen, wobei die Maschinengenauigkeit schon beim 8-Bit REAL-Typ so genau ist, daß der relative Fehler mit Sicherheit sehr weit unter dem 1%-Bereich liegt. Aus diesem Grund wird er für die Fehlerrechnung vernachlässigt.

[1] Das Kabel stellt u.U. eine Heizquelle mit einer Leistung von einigen hundert Watt dar.
[2] Möglich mit CPOW Programmversion 1.91.

Fehler durch Innenwiderstand der Meßgeräte

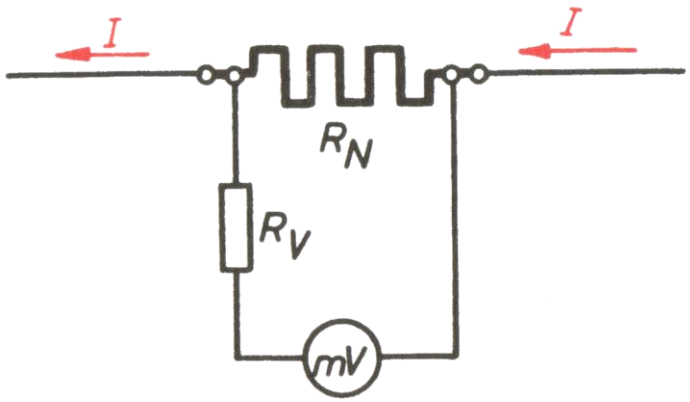


Abb 2.2 Messung des Spannungsabfalls am Shunt

Der Fehler kann vernachlässigt werden, da $R_V >> R_N$.

Quelle: [4]

2.5.3 Fehler bei der Weiterverarbeitung der Daten

Für eine korrekte Auswertung bzw. Weiterverarbeitung der Daten müssen zum Zeitpunkt der Messung noch weitere Variablen bekannt sein, die ihrerseits mit einem Fehler behaftet sein können:

- Raumtemperatur (siehe Kapitel 2.5.2)
- Länge des Leiters zwischen den Meßanschlüssen

2.6 Verarbeitung der gewonnenen Daten

Die mit den Messungen (bzw. mit dem Meßprogramm CPOW) gewonnenen Daten müssen aufgearbeitet werden, bevor mit Ihnen eine Aussage über Stom-Temperatur-Beziehungen gemacht werden kann. Anhand der folgenden Unterkapitel wird diese Weiterverarbeitung demonstriert:

2.6.1 Meßergebnisse im Textformat

Die ASCII-Dateien lassen sich mit einem Texteditor betrachten, sind aber sehr unübersichtlich.

Beispiel:

```
Measurement Time;Special Events;Measured Cable Temp;Wanted Current;[...]
55017;;+0004.638E+1; 115.00;115.07; 1.14729000000050E+0002; [...]
55056;;+0004.643E+1; 115.00;115.07; 1.14730500000063E+0002; [...]
55071;;+0004.64E+1; 115.00;115.06; 1.14720000000063E+0002; [...]
 [...]
```

2.6.2 Meßergebnisse als Microsoft Excel Dateien

Die Daten lassen sich einfach in ein Tabellenkalkulationsprogramm oder in MatLab® importieren. Eine genaue Beschreibung des Dateiformates befindet sich im Anhang I.

Beispiel:

40737	06:57 - Setting new current to 20.00 A							
40857	2,42E+01	20	20,11	2,00E+01	8,07E+00	4,00E-03	4,04E-01	2,02E-02
40896	2,43E+01	20	20,03	2,00E+01	8,07E+00	4,00E-03	4,04E-01	2,02E-02
40917	07:00 - Setting new current to 25.00 A							
41057	2,47E+01	25	24,97	2,50E+01	1,26E+01	4,99E-03	5,05E-01	2,02E-02
41097	2,49E+01	25	25,04	2,50E+01	1,26E+01	4,99E-03	5,05E-01	2,02E-02
41097	07:03 - Setting new current to 30.00 A							
41223	2,53E+01	30		2,99E+01	1,81E+01	5,98E-03	6,07E-01	2,03E-02
41257	2,54E+01	30	29,98	2,99E+01	1,82E+01	5,98E-03	6,07E-01	2,03E-02

2.6.3 Diagramme

Sind die Daten erst in eine Tabellenkalkulation oder in MatLab® importiert, lassen sich Diagramme in fast beliebiger Art erstellen. Beispiele finden sich in Anhang IV.

3. Erste Ergebnisse

3.1 Nach Messungen mit dem Programm CPOW

Nach einer vollständigen Messung mit CPOW läßt sich anhand der Ergebnisse ein Rückschluß über die Verwertbarkeit der Daten ziehen. Insbesondere Diagramme eignen sich dazu, um beispielsweise starke Schwankungen zu entdecken, die auf wechselnde Umgebungsbedingungen hindeuten könnten. Betrachtet man den Temperaturverlauf über der Zeit, so läßt sich erkennen, ob sich die Leitertemperatur bei jedem Strom stabilisiert hat.

3.2 Weiterverarbeitung der Daten

Bevor die Meßergebnisse weiterverarbeitet werden können, sind noch folgende Schritte notwendig:

a) Vor Beginn der Messung muß die Umgebungstemperatur T_e (= Leitertemperatur) manuell genau festgestellt werden[1].
b) Als nächstes muß der Widerstand des Leiters bei dieser Umgebungstemperatur festgestellt werden. Ist dies nicht möglich, so kann der Wert aus einer gegebenen Meßreihe aus Wertepaaren ($(I, R(I))$ mit einem quadratischen Polynom extrapoliert werden[2]. Dies liefert $R\big|_{I=0A} = R(T = T_e)$.
c) Die Formel zur Bestimmung von R(T) (siehe Kapitel 1.1) wird nach $R_0 = R_0(T_{0,}T, R)$ umgestellt und damit wird das notwendige Referenzwertepaar T_0 und R_0 errechnet.

Nun kann zusammen mit den (unabhängig von den oben beschriebenen Messungen) bestimmten Materialkonstanten α_0, β_0 ein Zusammenhang P(T) ermittelt werden.

3.3 Nach der Weiterberarbeitung der Daten

Es sei der einfache Fall angenommen, daß der Leiter frei in der Luft „hängt“, d. h. für den Wärmeübertragungskoeffizient α werden sich voraussichtlich Abhängigkeiten von folgenden Parametern ergeben [1]:

- Absoluttemperatur des Leiters
- Umgebungstemperatur
- Drahtquerschnitt
- Drahtgeometrie
- Wärmeleitfähigkeit der Isolierung
- Wärmeübergang an die Umgebung

[1] Ausnahme: mit Programmversion 1.91 ist es möglich, die Umgebungstemperatur zusätzlich zu den anderen Meßdaten aufzuzeichnen.
[2] Alternativ dazu kann der Kabelwiderstand bei Umgebungstemperatur mit einem (Milli-)Ohmmeter bestimmt werden.

Befindet sich der Leiter nicht vollständig in einem bestimmten Zustand, so muß zur Betrachtung differentieller Elemente übergegangen werden. Hier dürfte auch die Wärmeleitfähigkeit des Leitermaterials eine Rolle spielen.

3.4 Resümee & Ausblick

Mit dem Programm CPOW habe ich versucht, ein optimales Werkzeug für die Vermessung von Leitern zu schaffen. Dabei habe ich mich nicht streng an die Vorgaben gehalten, sondern es permanent an neue Forderungen angepaßt und unsere Erfahrungen mit einfließen lassen. Die Veränderung des Programms in kleinen Stücken hat sich vor allem nachteilig auf die Bedienerfreundlichkeit ausgewirkt, und sicherlich wäre nochmals eine grundlegende Überarbeitung ratsam. Trotzdem glaube ich, daß das Programm den Anforderungen gerecht wird und nun eine präzise und komfortable Meßmöglichkeit bietet.

Ferner kann das Programm als Grundstein für weitergehende Meßaufgaben genutzt werden (zumindest die Erfahrungen, die wir damit gemacht haben), beispielsweise bei komplexen Messungen innerhalb eines Fahrzeuges mit weitaus mehr Meßdaten.

Sollten solche Aufgaben irgendwann auf uns zu kommen, so könnten wir aus den gemachten Erfahrungen profitieren und in kurzer Zeit ein Programm (u. U. auch in einer anderen Programmiersprache) entwickeln, das die Ergebnisse komfortabel liefert. Vielleicht ist dies auch eine Rechtfertigung für den relativ großen Zeitaufwand, den die Programmentwicklung in Anspruch genommen hat.

ANHANG

(Appendix)

I. Program Documentation – the computer program "Cable Power"

Files and directories

CPOW.EXE the executable DOS Real Mode application

CPOW.INF contains some data like username, maximum cable temperature, shunt resistance, …

CPOW.LOG The ASCII file which contains all measurement data and some notes

The files must be installed in the directory C:\CPOW.

Format specifications of output files

Every time the program runs, all measured data is saved in the output file CPOW.LOG. It is not necessary (but possible) to delete this file after a measurement, because new data will be appended. In addition to this, another output file can be created. This file can be named by the user and it can only be used for one measurement. It contains only a part of the data saved in CPOW.LOG: after a current is set at the power supply, the program waits until the cable remains stationary.

Example:

```
Measurement time;Events;cable temperature;[…]
52162;10:13 - Setting new current to 135.00 A
52177;;+0005.549E+1; [...]
52217;;+0005.554E+1; [...]
52257;;+0005.556E+1; [...]
52297;;+0005.557E+1; [...]
52337;;+0005.557E+1; [...]
52377;;+0005.556E+1; [...]
52416;;+0005.557E+1; [...]
52457;;+0005.558E+1; [...]
52496;;+0005.557E+1; [...]
```

These lines can only be found in CPOW.LOG but not in the user specified output file

- Output files generated by CPOW can be found in C:\CPOW
- All output files have the ending ".log"
- File format is ASCII
- The data is formatted in lines and columns:
  - Lines are separated by CRLF ("Carriage Return" and "Line Feed")
  - Columns are separated by a semicolon (" ; ")
  - The meaning of each column can be found in the first line of the output file
  - There are 4 types of data lines:
    - First lines: the first line of the file is a "headline", the second line is empty, the third line contains descriptions for each column, separated by semicolon
    - Empty lines: in the file CPOW.LOG an empty line is added every time the program starts

- Event lines: All but the first two columns are empty. The second column starts with the system time. Examples:

```
0;19:20 - User Tom startet a new measurement! Date: 2001-3-5

17;19:20 - Setting new current to   10.00 A

54277;20:10 - Setting new current to  190.00 A
```

- Measurement lines: contain new measured data. The second column is empty. Examples:

```
55017;;+0004.638E+1; 115.00;115.07; 1.14729000000050E+0002; [...]
55056;;+0004.643E+1; 115.00;115.07; 1.14730500000063E+0002; [...]
```

- Numerical expressions:
  - A point "." is used for decimal fractions
  - Some values are given in terms of powers of ten, e.g. `+1.248E+0001` means 12.48

Software requirements

The program CPOW.EXE needs a MS-DOS compatible operating system. It also works with emulated DOS under Windows 9x. The program was not tested under other operating systems and there could be a problem with an OS like Windows NT, witch has stronger hardware access restrictions.

Hardware requirements

- A GPIB interface card "CIO-PC2A"[1] from "Measurement Computing"[2] must be installed in the computer and in the CBCONF program the following devices must be configured:

- Device "power": The power supply from EA.
- Device "prema1": The multimeter DMM 6001 from PREMA with a multichannel input card.

Remarks on the GPIB interface

- The CIO-PC2A interface card does not need a TSR program. The drivers must be installed, and in AUTOEXEC.BAT a line like this must be added:

  SET GPIBDIR=C:\GPIB

- With the CBCONF program which is delivered with the card drivers you have to set up the devices.

[1] You may also use another card from Measurement Computing, which is compatible to the CBCONF program. In this case you just have to configure the board. No program changes and no recompilations are need.
[2] You may also use a IEEE-488 compatible interface card from another manufacturer, e.g. from National Instruments. In that case you have at least to recompile the program CPOW.PAS with another "TPDecl.tpu" unit.

- The ID numbers of the devices can be chosen free. Only the device names are important for CPOW.
- Power supply options:
  - Name of device: power
  - Timeout setting: 10 secs
  - EOS byte: 10 (this is a decimal value which means LF in ASCII code)
  - Terminate read on EOS: YES
  - Force re-adressing: YES
- DMM options:
  - Name of device: prema1
  - Timeout setting: 10 secs
  - EOS byte: 10 (this is a decimal value which means LF in ASCII code)
  - Terminate read on EOS: YES
  - Force re-adressing: YES
- At the DMM you have to set the following values (internal program 12, see manual for details):
  - Addr
  - Addr nn
  - EOS=LF
  - Bus Display y

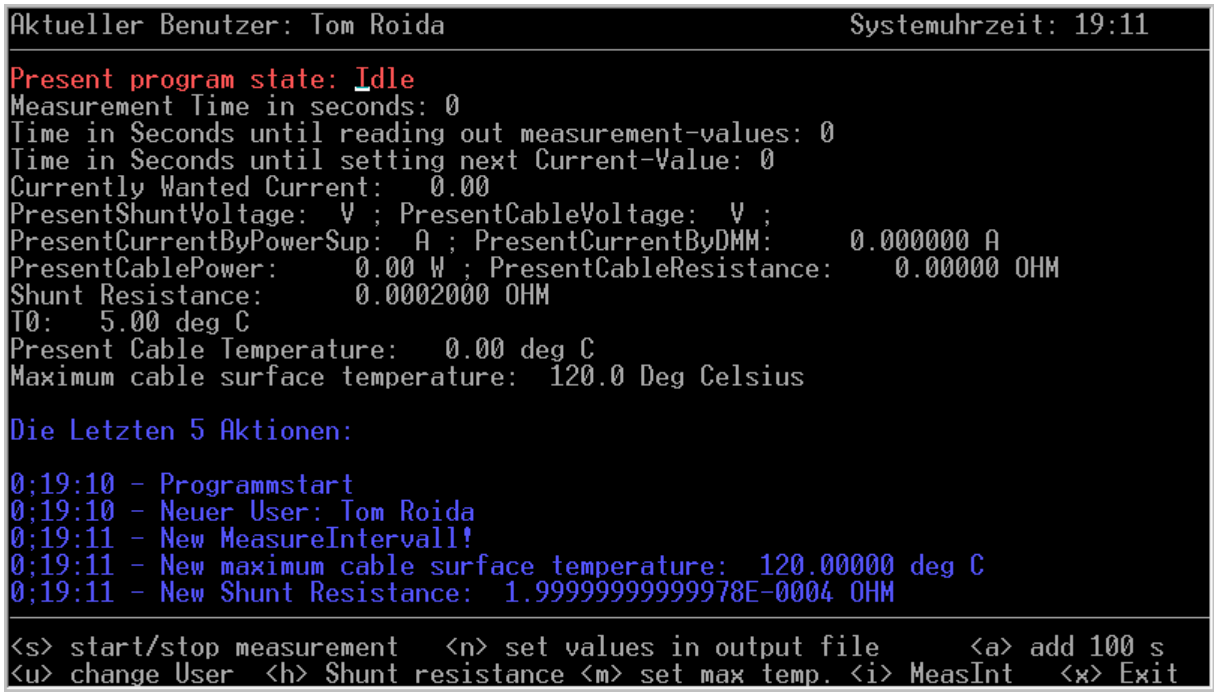
Operating manual

Starting the program

The program can be started by entering "CPOW [Enter]" in the directory C:\CPOW or by a double-click on the CPOW-Icon (e.g. if Windows® is installed).

After starting the program it tries to initialise the GPIB Devices. If this operation completes successfully the program shows the main menu.

The main menu



This screen is visible during nearly the whole measurement. It shows the many values and also what the program is doing at the moment. The blue lines show what the program did before.

From here you have following menu options by pressing a specific key:

Key	function	notes
s	Start measurement / stop measurement	See next chapter.
u	Change username	The username will be written to the output files.
n	Change amount of values in secondary output file	See “Format specifications of output files”.
h	Change value of shunt resistance	Enter a value in Ω.
m	Change maximum cable temperature	This value should be the maximum temperature which is allowed for the cable. If cable temperature exceeds this value, the program aborts immediately. Enter a value in °C.
I	Set time between two single measurements	Enter a new value in seconds.
a	Add 10 / 100 seconds to “Time until setting next current value”	Press this key to force the program to wait longer until it sets the next current value.
x	Leave program	

Starting a measurement

After pressing "s" at the main menu, the user is asked the following questions:

Take additional temperature measurements?	YES: After each single measurement the program checks Channels 5 – 9 for up to five PT100 sensors, and the temperatures are written to the output files.
Do you want to measure upwards?	See example below.
Do you want to measure downwards?	See example below. At least one of this two questions has to be answered with "YES".
Please enter minimum current value:	Enter a value in A. See example below.
Please enter maximum current value:	See example below.
Please enter step size:	See example below. The last three entered values must fit together.

Now the program shows the user which current values will be set during the measurement, and how long the measurement will take. The user must confirm this before the measurement starts.

Example 1:

Do you want to measure upwards?	NO
Do you want to measure downwards?	YES
Please enter minimum current value:	10
Please enter maximum current value:	20
Please enter step size:	2

Now the program sets the following current values:
20A, 18A, 16A, 14A, 12A, 10A

Example 2:

Do you want to measure upwards?	YES
Do you want to measure downwards?	YES
Please enter minimum current value:	0
Please enter maximum current value:	20
Please enter step size:	10

Now the program sets the following current values:
0A, 10A, 20A, 10A, 0A

Example 3:

Do you want to measure upwards?	NO
Do you want to measure downwards?	NO

The program does not accept these settings.

Example 4:

Please enter minimum current value:	0
Please enter maximum current value:	20
Please enter step size:	7

The program does not accept these settings.

Error messages

During normal program operation, the program should not "crash" even if an error occurs. The program contains internal exception handlers.

It belongs to the kind of the error, if the program stops a measurement or not. For example, if the temperature of the cable can once not be measured, the program continues after a few seconds. But if the temperature cannot be measured for a longer time (≈1 minute), the program interprets this as a "fatal error" and aborts.

Optional command line parameters

The program can be started with a command line parameter. The parameter must be written in lower characters. Only one parameter can be used at once.

- "cold" This starts the program without initialising the GPIB interface and the timer. This is for testing the program without the GPIB devices. CPOW will not work probably.
- "led" This option forces the program to show its GPIB bus activity with the keyboard-LEDs:
  - NUM-LED: Reading GPIB bus
  - CAPSLOCK-LED: Checking cable temperature (does not only belong to GPIB activity)
  - SCROLL-LED: Writing to GPIB bus

Known bugs and problems

subject: language
responsible program parts: complete pascal source code
seriousness: minor

As described above, the user-interface of CPOW is written in English. However, because some parts of the program have been used in a "German" program before, CPOW may still contain some German expressions. Some buttons are marked with "Ja" and "Nein" instead of "Yes" and "No". Also some error messages will appear in German.
This problem also affects the source code (e.g. variable names) and the source code documentation.

---

subject: program crash
responsible program parts: unknown
seriousness: SAFETY, high

Since program version 1.50 the program sometimes crashed without an error message. This caused in nearly every time the computer to hang, even if the program was running under Windows 95. The reason for the error could not be allocated to a specific program part. Maybe it is not even the source code! Since program version 1.60, after some minor changes at the hardware timer section, the error disappeared.

---

subject: temperature measurement
responsible program parts: none
seriousness: SAFETY

As described above, the cable temperature is periodically measured by a surface mounted PT100 sensor. The value is compared to the "MaxTemp" variable. If the sensor falls down the cable, not the cable temperature but the room temperature is measured. So the program can not recognize if the cable temperature is too high.

---

subject: temperature measurement
responsible program parts: cpow.pas
seriousness: SAFETY

If something happens to the PT100, e.g. a connector fails, the temperature cannot be read. This means the screen shows a blank line, and the computer interprets this as 0°C. If this value is compared to the "MaxTemp" variable, for the computer everything seems to be o.k. .

---

subject: time of measurements
responsible program parts: cpow.pas
seriousness: usability of the program

H.-D. Ließ developed a theory about the time until the cable temperature becomes stable and does no more increase / decrease. This theory is made for flat cables but not for round cables [1].
In the current program version the procedure for calculating the time with the formulas of H.-D. Ließ, is inactive. Instead a more simple function is used. For details see source code CPOW.PAS, PROCEDURE "nTau".

---

subject: failure of "IbRd",
responsible program parts: ugpib25.pas or tpdecl.tpu
seriousness: minor

Sometimes the readout of a value from the DMM fails. On the screen you see a window with a message like this: "ERROR: Could not read temperature. Press any key or wait some seconds").
The cause for this problem could not yet be located. The "TPDecl"-Function "IBRd" fails with a time out, although the DMM is ready to deliver a value.

II. Remarks on measurement

Remarks on measurement

- Before you start the program you can make an offset correction at the PREMA DMM6001 multimeter:
  - Take care that the DMM was not turned off for at least some hours;
  - turn off the power supply with the main switch;
  - at the DMM switch to channel 1 (shunt) by pressing CHA, 1, ENTER;
  - switch to “Voltage, DC” by pressing V= ;
  - switch to “autorange” by pressing AUTO; An arrow will appear at the display;
  - press ZERO to start the correction procedure. The display will show “ZERO”;
  - wait until “ZERO” disappears.
- Check all cable connections for the measurement, especially the GPIB bus cable.
- Switch on all devices before you start the program.
- At the computer the executable file CPOW.EXE must be installed in the directory C:\CPOW.
- Never set a current higher than allowed:
  - check shunt resistor for max. current;
  - check cable for max. current.
- Turn the potentiometers at the power supply to zero.
- If you decide to cool down the shunt, be aware of the air flow. The cable must not be affected.
- Take care of the PT100 sensor. It should be attached to the cable surface for safety reasons.
- After the measurements you will find all output files in C:\CPOW with the ending “.log”. You need never to delete the file CPOW.LOG, unless it becomes too big for you to handle.
- Never switch off the DMM.

III. Excerpts from source code

The Pascal source code is divided into several files:

CPOW.PAS or CPOW200.PAS	the main program file
Unit GPIB300.PAS	routines for GPIB bus (IEEE488) and the interface to all bus devices
Unit UTimer.PAS	routines for initialization of hardware timer (not described below)
Unit UScrKeyb	user interface (not described below)

The following source code is not complete. It contains only relevant data for the measurements and for the GPIB bus interface:

```
Program cpow200_for_doku;
{V2.00 29.04.2001 }

{for documentation purposes}

{! ! ! ! ! this file contains only a part of the source code ! ! ! ! ! !}
written by Patrick Mack, Tom Roida 2001

{Compiler-Befehle:}
{$A+}{$B-}{$N+}{E+}{$G+}{$I-}{$P-}{$V+}{$S+}
{erweiterte syntax}
{$X+}
{Bereichsueberpruefung}
{$R-}


Uses
  Crt, DOS,
  UGPIB300,
  utimer, UScrKeyb;



Type

  tInfoblock = record                 {Data taken from .inf file}
                                      {This record is saved in CPOW.INF}
       Username : String;
       MaxTemp : Real;              {maximun cable temperature}
       RShunt : Real;               {Resistance of Shunt in Ohm}
       MeasureIntervall:LongInt;    {time between two measurements in seconds}
       alfa, beta: real;            {temperature koeff.}
       R0, T0 : real;               {Resistance of "Cold" Cable,
                                             Environment Temperature}
       TakeN:Longint;               {How much values per current
                                             do you want in xFile}
  end;


VAR

  FileInfo          : File of tInfoblock;
  InfoBlock         : tInfoblock;
  FileLog           : Text;
  XFileLog          : Text;
  PathXFileLog      : String;
  XFileState,XFileNow        : boolean;
```

```
  LastAction        : Array[0..9] of String;

  WasErr            : Boolean;
  StateS:String;


  MeasureTime : LongInt;               {Seconds running since start of
                                        measurement}
  TargetTimeSetCurrent : Longint;      {If TargettimeSetCurrent=MeasurementTime
                                        The Next Current value will be set}
  TargetTimeTakeMeasurement : Longint; {If TargettimeTakeM. = MeasurementTime
                                        then TakeMeasurement}
  MeasureState: Boolean;               {Is a measurement-row running?}
  TakeMeasurementInProgress: boolean;  {Is currently a Read-Out of
                                        Values in progres?}

  PresentWantedCurrent,                {Wanted Current in Amps to be set NOW}
  PresentTemp,                         {Present Temperature in Degrees Celsius}
  PresentCurrentByPowerSup,            {Present Current in Amps,
                                        Measured by the Power Device itself}
  PresentShuntVoltage,                 {Measured Voltage Drop at Shunt}
  PresentCurrentByDMM,                 {Current calculated with Shunt Resistance
                                        and PresentVoltageShunt}
  PresentCableVoltage,                 {Measured Voltage Drop at Cable}
  PresentCablePower, PresentCableResistance : real;      {Calculated Values}

  PresentTempS, PresentCurrentByPowerSupS,
  PresentShuntVoltageS, PresentCableVoltageS: String;

  ExtraTempSVec : Array[0..4] of String; {For additional temp. measurement}
  ExtraTemp : boolean;

  FailedReadTempCounter : Integer;


Const
  lengthlead : real =10.00;          {meters ; Patrick's function}
  thicknesslead: real = 0.0003;       {meters ; Patrick}
  widthlead: real = 0.042;            {meters ; Patrick}
  thicknessisolation: real = 0.0002; {meters ; Patrick}

  NoError        : byte = 0;
  TempError      : byte = 8;
  ValError       : byte = 16;
  StrError       : byte = 32;
  FileError      : byte = 64;
  BusError       : byte = 128;

  PathCPow       : String ='C:\CPOW';
  FileCPow       : String ='CPOW.EXE';
  PathFileLog    : String ='C:\CPOW\CPOW.LOG';
  PathFileInfo   : String ='C:\CPOW\CPOW.INF';


  MaxTempReadFail : integer = 8;


Procedure ReadTemp;
{reads temperature from PT100 at channel "-" and converts the value}
{uses unit uGPIB200}
VAR
 TempErrorCode:Integer;

Begin
  If TakemeasurementInProgress then fwriteEvent(
  'Software problem detected: parrallel access to gpib-interface! Delaying...');
  While TakeMeasurementInProgress do Delay(10);
  TakeMeasurementInProgress:=true;
```

```
  GPIBReadTemp(PresentTempS);
  TakeMeasurementInProgress:=false;
  Val(PresenttempS,PresentTemp,TempErrorCode);
  If TempErrorCode <> 0 then
    ErrorCode:=ErrorCode or ValError;
end; {ReadTemp}

Procedure ReadDevices(Var Time: Longint);
{Reads out Measurement Values as STRINGs and Calculates them to REAL}
{Time is set to MeasureTime when the results are delivered}
{V2.00}
VAR
 TempErrorCode:Integer;

Begin
  ErrorCode:=NoError;
  While TakeMeasurementInProgress do Delay(10);
  TakeMeasurementInProgress:=true;
  GPIBReadCurrentByPower(PresentCurrentByPowerSupS);
  Val(PresentCurrentByPowerSupS,PresentCurrentByPowerSup,TempErrorCode);
  If TempErrorCode <> 0 then
    ErrorCode:=ErrorCode or ValError;
  GPIBReadcable_shunt(PresentCableVoltageS,PresentShuntVoltageS);
  Time:=MeasureTime;
  Val(PresentCableVoltageS,PresentCableVoltage,TempErrorCode);
  If TempErrorCode <> 0 then
    ErrorCode:=ErrorCode or ValError;
  Val(PresentShuntVoltageS,PresentShuntVoltage,TempErrorCode);
  If TempErrorCode <> 0 then
    ErrorCode:=ErrorCode or ValError;
  If InfoBlock.RShunt > 0 then
    PresentCurrentByDMM:=PresentShuntVoltage / InfoBlock.RShunt
  else PresentCurrentByDMM:=-1.0;
  If PresentCurrentByDMM > 0 then
    PresentCableResistance:=PresentCableVoltage / PresentCurrentByDMM
  else PresentCableResistance:=-1.0;
  PresentCablePower:=PresentCurrentByDMM * PresentCableVoltage;
  If ExtraTemp then
    Begin
      GPIBReadExtraTemp(ExtraTempSVec);
    End;
  TakeMeasurementInProgress:=false;
  If (ErrorCode and ValError) <> NoError then HandleError;
end; {ReadDevices}

Procedure TakeMeasurement;
{Written by Patrick Mack 1.03.01, modified by Tom Roida 3.4.01}
{Starts a single measurement and converts the values,
 then it writes them into the log-files using procedure fWrite}
Var
   j:integer;
   time: longint;
   PresentWantedCurrentS: String;
   PresentCurrentbyDMMS: String;
   PresentCablePowerS: String;
   text: String;
   PresentCableResistanceS: String;
   timeS: string;
Begin
ReadDevices(time);
Str(PresentWantedCurrent:5:2, PresentWantedCurrentS);
Str(PresentCurrentbyDMM, PresentCurrentbyDMMS);
Str(PresentCablePower, PresentCablePowerS);
Str(PresentCableResistance, PresentCableResistanceS);
text:=PresentTempS+';'+PresentWantedCurrentS+';'+PresentCurrentbyPowerSupS+';'+
      PresentCurrentbyDMMS+';'+PresentCablePowerS+';'+PresentShuntVoltageS+';'+
      PresentCableVoltageS+';'+PresentCableResistanceS;
If ExtraTemp then for j := 0 to 4 do
  text:=text+';'+ExtraTempSVec[j];
fwrite(time,text);
```

```
End;

Procedure CheckMaxTemp;
{V1.00 13.02.2001}
{checks if the cable temperature does not exceed the maximum allowed }
begin
  KB_Set_Led(Caps, true);
  If not (TakeMeasurementInProgress or GPIBBusTalkInProgress) then
    Begin
      FailedReadTempCounter:=0;
      ReadTemp;
      Refresh:=True;
    end
  else
    inc(FailedReadTempCounter);
  If (PresentTemp >= Infoblock.MaxTemp) then
    Begin;
      DirectStop;
      Sound(2500);
      DeInitCheckMaxTempTimer;
      fWriteEvent('Cable Temperature exeeded MaxTemp - Program aborted.'+
                  'Please check measurement setup.');
      ERR('Cable Temperature exeeded MaxTemp. Please check measurement setup.',
True);
    End;
  If (FailedReadTempCounter > MaxTempReadFail) then
    Begin;
      DirectStop;
      Sound(5000);
      DeInitCheckMaxTempTimer;
      fWriteEvent('Program hangs, probably caused by GPIB BUS Error');
      ERR('Program hangs, probably caused by GPIB BUS Error  - '+
          'Program aborted for sefety reasons.'+
          'Please check measurement setup.', True);
    End;
end;


Function nTau : real;  {in seconds}
{Mack 01.03.01, modified by Tom Roida}
{calculates the time how long it takes until the cable temperature
 becomes stable. Because of problems the main part of this procedure is
 inactive. Instead a smaller formula is used, which does not depend on the
 cable.}

Const
     alpham=4.8;               {mittlerer Wärmeübergangskoeffizient in Luft,
                                gesetzt nach Erfahrungswerten}
     alphaT= 3.81E-003;        {linearer Temperaturkoeffizient}
     betai= 3.4113E+006;       {Wärmekapazität der Isolation PVC}
     betal= 1.3245E+006;       {Wärmekapazität von Kupfer}
Var
   rho0: real;
   a,b,s,l: real;
   Tau: real;

Begin
     l:= lengthlead;
     a:= thicknesslead;
     b:= widthlead;
     s:= thicknessisolation;
     {rho0:= (a*b*InfoBlock.r0)/l;
     Tau:= (a*b*b*(betal*a+2*betai*s))/(2*alpham*a*b*b-alphaT*rho0*
       PresentwantedCurrent*PresentwantedCurrent);
     ntau:=3*tau +  (InfoBlock.TakeN+0.6)*InfoBlock.MeasureIntervall;
}

     nTau:=400+(PresentWantedCurrent/WantedCurrentMax)*800+
       (InfoBlock.TakeN+0.6)*InfoBlock.MeasureIntervall;
```

```
End;

Procedure CalculateNewTargetTimeSetCurrent;
Begin
  TargettimeSetCurrent:=MeasureTime+Round(ntau);
End;



{-----------------Layer 3 MAIN PROGRAM -------------------------------------}

Var
  PresentWantedCurrentS : String;
  Xit                   : Boolean;
  Refresh               : Boolean;

BEGIN
  Writeln('Please wait for GPIB-Bus initialisation');
  If not (ParamStr(1)='cold') then If not InitGPIBBus then
      Err('GPIB Initialisation Failed!', true);
  If not (ParamStr(1)='cold') then InitCheckMaxTempTimer;

  Xit:=false;
  Refresh:=true;
  StateS:='Idle';
  While Keypressed do ReadKey;
  Repeat
    Repeat
      If MeasureState and (MeasureTime >= TargetTimeTakeMeasurement) then
          Begin
            If ((TargettimeSetCurrent - MeasureTime) <
               ((InfoBlock.TakeN+0.13)*Infoblock.MeasureIntervall)) then
              XFileNow:=true else XFileNow:=false;
            StateS:='Reading Out GPIB Devices';
            RefreshScreen;
            TakeMeasurement;
            StateS:='Waiting';
            TargetTimeTakeMeasurement:=TargetTimeTakeMeasurement+
                                       Infoblock.MeasureIntervall;
            Refresh:=true;
          End;

      If MeasureState and (MeasureTime >= TargetTimeSetCurrent) then
          Begin
            StateS:='Setting new Current Value';
            RefreshScreen;
            CalculateNewTargetTimeSetCurrent;
            If (CalculateNewCurrent) then
              Begin
                Str(PresentWantedCurrent:6:2,PresentWantedCurrentS);
                fWriteEvent('Setting new current to ' +
                            PresentWantedCurrentS + ' A');
                SetCurrent(PresentWantedCurrent);
                StateS:='Waiting';
              end
            else
              Begin
                MeasureState:=False;
                PresentWantedCurrent:=0.0;

                SetCurrent(PresentWantedCurrent);
                fWriteEvent('Stoped because: finished measurement');
                Uuups('Stoped because: last current value');
                StateS:='Idle';
                RefreshScreen;
                Delay(800);
                Uuups('Finished measurement');
              End;
```

```
            Refresh:=true;
          End;
      Until Keypressed or Refresh;
      If Keypressed then
        Begin
          Menu;
          Refresh:=true;
          RefreshScreen;
        End
      else
        Begin
          RefreshScreen;
        End;
    Until Xit;
    If not (ParamStr(1)='cold') then DeInitCheckMaxTempTimer;
    If not (ParamStr(1)='cold') then DeInitGPIBBus;
    fWriteEvent('Programm verlassen');
END.

written by Patrick Mack, Tom Roida 2001
```

```
Unit ugpib300_for_documentation;
{V3.00, 29.04.01}

{for documentation purposes}

{! ! ! ! ! this file contains only a part of the source code ! ! ! ! ! !}

Interface

Var
  GPIBBusTalkInProgress : boolean;


Procedure SetCurrent(Current: Real);
Procedure DirectStop;
Function InitGpibBus : boolean;
Procedure DeInitGpibBus;
Procedure GPIBReadTemp(var Temps:String); {Reads surface Temperature}
Procedure GPIBInitReadCable_shunt;
Procedure GPIBReadCable_shunt(var CableS,ShuntS:String); {Reads Volt. drop at
                                                          Cable + Shunt Temperature}
Procedure GPIBReadExtraTemp(var ExtraTempSVec:Array of String); {Reads multiple
                                                          Temperatures}
Procedure GPIBReadCurrentByPower(var Currents:String); {Reads Power's opinion
                                                          of PresentCurrent}

Implementation

Uses
  TPDecl,UScrKeyb, CRT;

Const
  EOS : Char= #10; {Sets LF as Line End Character}
  brdS : String ='GPIB0';
  DMMS : String ='prema1';
  PowerS : String ='power';

Var
  brd,power,dmm:integer; {Device Handles}
  mport: String;


{***********************************************************************}
{                                                                       }
{  Name:         PrintErrors                                            }
{  Description:  This procedure prints an error message followed by the  }
{                values of the global staus variables - IBSTA, IBERR,    }
{                and IBCNT.  For IBSTA it prints the name of every bit   }
{                that is set.  For IBERR it prints the mnemonic for the  }
{                error code.                                            }
{                                                                       }
{***********************************************************************}

procedure PrintErrors (message:string);
Var i:integer;
begin
   Box;
   GotoXY(sright+2,sdown+1);
   write (message);                             { Print error message }

   GotoXY(sright+2,sdown+2);
   write ('   IBSTA=', ibsta,' <');               { Print IBSTA value }
   if ibsta and ERR  <> 0 then write (' ERR');
   if ibsta and TIMO <> 0 then write (' TIMO');  { Print names of set bits }
   if ibsta and EEND <> 0 then write (' END');
   if ibsta and SRQI <> 0 then write (' SRQI');
   if ibsta and RQS  <> 0 then write (' RQS');
   if ibsta and SPOLL <> 0 then write (' SPOLL');
   if ibsta and EVENT <> 0 then write (' EVENT');
   if ibsta and CMPL <> 0 then write (' CMPL');
```

```
   if ibsta and LOK  <> 0 then write (' LOK');
   if ibsta and REM  <> 0 then write (' REM');
   if ibsta and CIC  <> 0 then write (' CIC');
   if ibsta and ATN  <> 0 then write (' ATN');
   if ibsta and TACS <> 0 then write (' TACS');
   if ibsta and LACS <> 0 then write (' LACS');
   if ibsta and DTAS <> 0 then write (' DTAS');
   if ibsta and DCAS <> 0 then write (' DCAS');
   write( ' >');

   GotoXY(sright+2,sdown+3);
   write('   IBERR=', iberr);                      { Print IBERR value }
   if iberr = EDVR then write (' EDVR <DOS Error>');   { Print mnenmonic }
   if iberr = ECIC then write (' ECIC <Not CIC>');
   if iberr = ENOL then write (' ENOL <No Listener>');
   if iberr = EADR then write (' EADR <Address error>');
   if iberr = EARG then write (' EARG <Invalid argument>');
   if iberr = ESAC then write (' ESAC <Not Sys Ctrlr>');
   if iberr = EABO then write (' EABO <Op. aborted>');
   if iberr = ENEB then write (' ENEB <No GPIB board>');
   if iberr = EOIP then write (' EOIP <Async I/O in prg>');
   if iberr = ECAP then write (' ECAP <No capability>');
   if iberr = EFSO then write (' EFSO <File sys. error>');
   if iberr = EBUS then write (' EBUS <Command error>');
   if iberr = ESTB then write (' ESTB <Status byte lost>');
   if iberr = ESRQ then write (' ESRQ <SRQ stuck on>');
   if iberr = ETAB then write (' ETAB <Table Overflow>');
   if iberr = ECFG then write (' ECFG <Incorrect board>');

   GotoXY(sright+2,sdown+4);
   write('   IBCNT=', ibcnt);
   Snd(False);
   GotoXY(sright+2,sdown+6);
   write('Press key to continue or Wait');
   While Keypressed do Readkey;
   For i := 1 to 15 do
     begin
       Delay(200);
       If Keypressed then Break;
     end;
   While Keypressed do Readkey;
end;

Procedure MKnbuf(s:string; var nb:nbuf);
Var
  StrLen,i:LongInt;
Begin
  StrLen := length (s);
  for i:=1 to nbufsize do              { Copy the name into char array }
      begin
      if i <= StrLen then
         nb[i] := s[i]
      else
         nb[i] := ' ';
      end;
End;


{*********************************************************************}
{                                                                     }
{  Name:         WriteCommand                                         }
{  Arguments:    DevHandle - handle of opened GPIB device             }
{                CommandStr - command string to write to device       }
{  Description:  Writes a string to a GPIB device and checks for errors  }
{                                                                     }
{*********************************************************************}

procedure WriteCommand (DevHandle:integer; CommandStr:string);
```

```
var
   StrLen,i:integer;
   WriteBuffer:cbuf;

begin
   if GpibBusTalkInProgress then
     Begin
       PrintErrors ('Overlapping Read');
       Sound(100);
       while GpibBusTalkInProgress do delay(10);
       NoSound;
     end;
   GPIBBusTalkInProgress:=True;
   StrLen := length (CommandStr);             {  Copy from string to buffer }
   for i:=1 to StrLen do
      WriteBuffer[i] := CommandStr[i];
   WriteBuffer[StrLen+1]:=EOS;
   Kb_Set_Led(Scrl,true);
   ibwrt (DevHandle, WriteBuffer, StrLen+1);           {Write to GPIB  }
   Kb_Set_Led(Scrl,False);
   GPIBBusTalkInProgress:=False;
   if (ibcnt<>StrLen+1) or (ibsta and ERR <> 0) then  { Check for errors}
      PrintErrors ('   IBWRT failed while writing '+CommandStr);
end;



{*************************************************************************}
{                                                                         }
{  Name:         ReadValue                                                }
{  Arguments:    DevHandle - handle of opened GPIB device                 }
{                buffer - buffer to read data into                        }
{                bufsize - size of buffer                                 }
{  Description:  Reads from a GPIB device                                 }
{                                                                         }
{*************************************************************************}


procedure ReadValue (DevHandle:integer; var ReadStr:string);

var
   ReadBuffer:cbuf;
   i:integer;

begin
   ReadStr:='';
   if GpibBusTalkInProgress then
     Begin
       PrintErrors ('Overlapping Read');
       Sound(100);
       while GpibBusTalkInProgress do delay(10);
       NoSound;
     end;
   GPIBBusTalkInProgress:=True;
   Kb_Set_Led(Numl,true);
   ibrd (DevHandle, ReadBuffer, cbufsize);     { Read from GPIB }
   Kb_Set_Led(Numl,false);
   GPIBBusTalkInProgress:=False;
   if (ibcnt = 0) or (ibsta and ERR <> 0) then { Check for errors }
     PrintErrors ('IBRD failed')
   else
     begin
      If ReadBuffer[ibcnt] <> EOS then PrintErrors('IBRD String does not contain
EOS Byte');
      for i:=1 to ibcnt-1 do
         ReadStr:=ReadStr+ReadBuffer[i];          { Copy from buffer to string }
      end;
end;
```

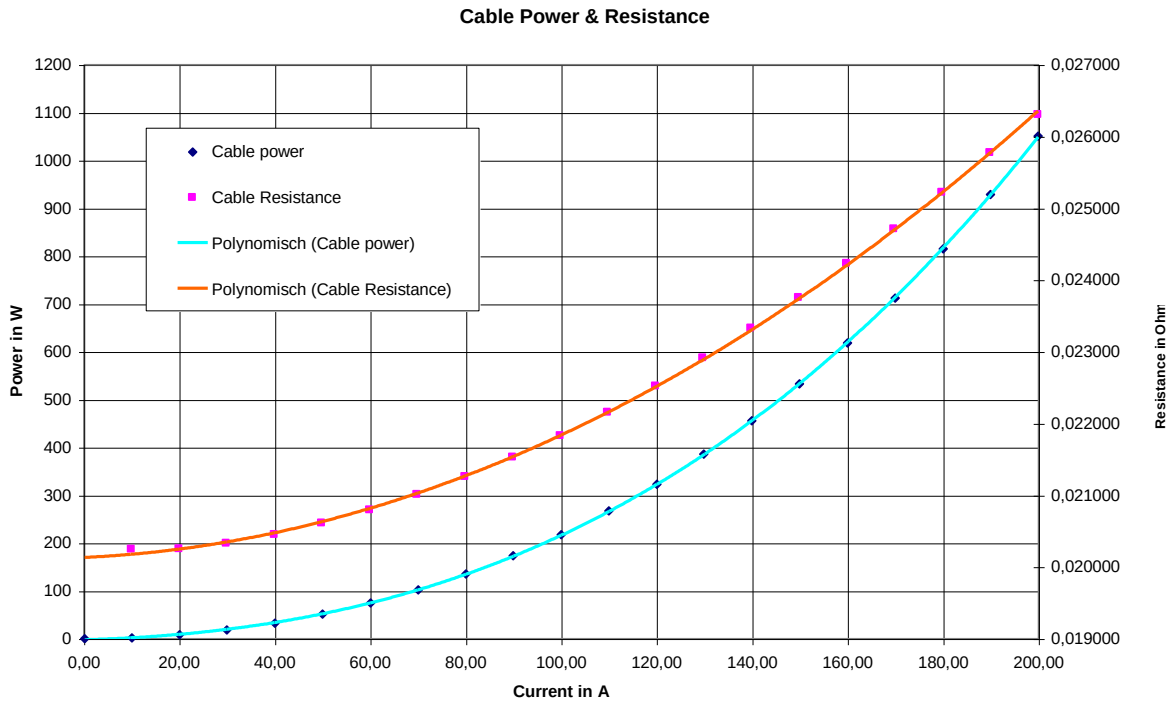
```
{-------------------------------------------------------------------------------}

Var
  DTemp, DShunt, DCable, DPower:integer;


Procedure GPIBReadTemp(var Temps:String); {Reads cable surface Temperature}
Begin
  If MPort <> 'mo' then
    Begin
      MPort:='mo';
      WriteCommand(dmm,MPort);
      WriteCommand(dmm,'TCR1T3S0');
      Delay(500);
    end;
  ReadValue(dmm,temps);
  If Pos('TC',TempS) <>13 then
    Begin PrintErrors('Could not read Temperature!!'); end
  else
    begin
      Delete(tempS,13,100);
    end;
End;

Procedure GPIBReadExtraTemp(var ExtraTempSVec:Array of String);
{Reads multiple Temperatures}
Var j:integer;
Begin
  For j:=0 to 4 do ExtraTempSVec[j] :='';
  For j:=0 to 4 do
    Begin
      MPort:='m'+Chr(48+5+j);             {'0'+5+j}
      WriteCommand(dmm,MPort);
      WriteCommand(dmm,'TCR1T1S0');
      Delay(110);
      WriteCommand(dmm,'T4S1');
      Delay(1000);
      ReadValue(dmm,ExtratempSVec[j]);
      If Pos('TC',ExtratempSVec[j]) <>13 then Begin PrintErrors('Could not read
Extra Temperature!!'); end
        else
      begin
        Delete(ExtratempSVec[j],13,100);
      end;
    End;
End;


Procedure GPIBReadCable_shunt(var CableS,ShuntS:String);
{Reads Volt. drop Cable + Shunt Temperature}
Begin
  MPort:='m3';
  WriteCommand(DMM,mport);
  WriteCommand(DMM,'VDT1A1');
  Delay(2000);
  WriteCommand(DMM,'T5S1');
  Delay(5100);
  ReadValue(dmm,Cables);
  If Pos('VD',CableS) <>13 then
    Begin PrintErrors('Could not read Cable Voltage Drop!!'); end
  else
    begin
      Delete(CableS,13,100);
    end;
  WriteCommand(DMM,'VDT1A0R1S0');
  MPort:='m1';
  WriteCommand(DMM,mport);
  Delay(200);
  WriteCommand(DMM,'T5S1');
```

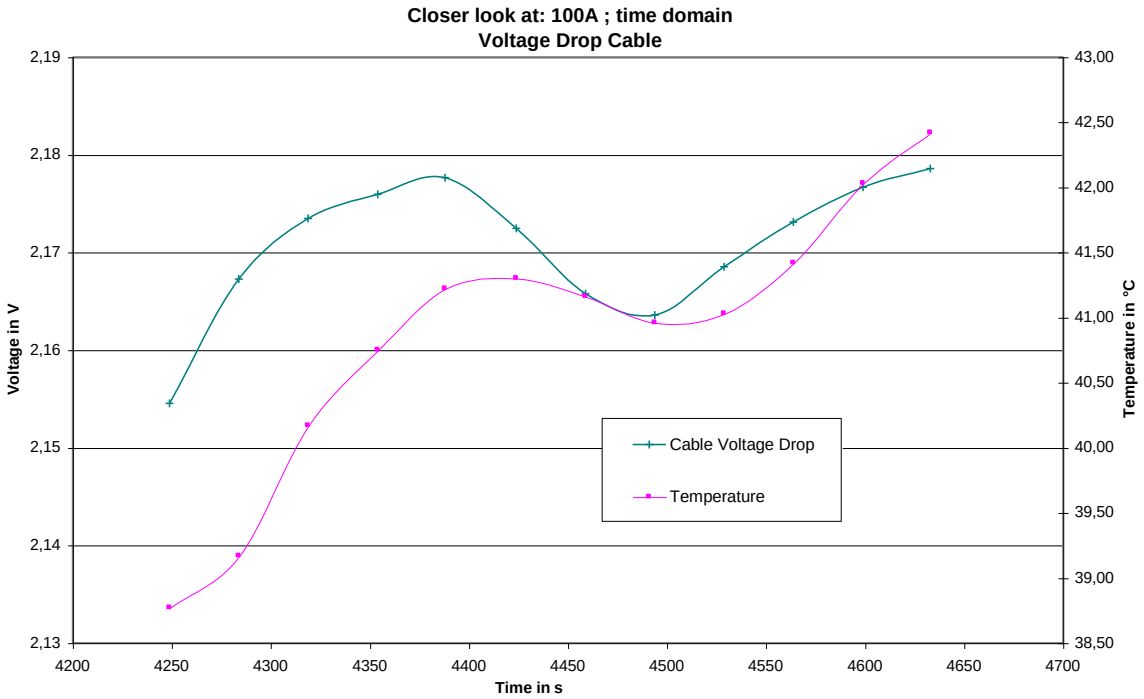
```
  Delay(5100);
  ReadValue(dmm,Shunts);
  If Pos('VD',ShuntS) <>13 then
    Begin PrintErrors('Could not read Shunt Voltage Drop!!'); end
  else
    begin
      Delete(ShuntS,13,100);
    end;
End;

Procedure GPIBInitReadCable_shunt;
Begin
End;


Procedure SetCurrent(Current: Real);
Var
  CurrentS, rcs : String;
  ec:integer;
  cc:real;

Begin
  Str(Current:10:2,CurrentS);
  While Pos(' ',CurrentS) > 0 do
    Delete(CurrentS,Pos(' ',CurrentS),1);
  CurrentS:='Current '+CurrentS;
  WriteCommand(Power,CurrentS);
  WriteCommand(power,'Current?');
  ReadValue(power,rcs);
  {Delete(rcs,1,1);}
  Val(rcs,cc,ec);
  If (cc <> Current) or (ec <> 0) then
  Begin PrintErrors('Could not set >'+CurrentS+'< at POWER-Device!!'); end;
End;

Procedure GPIBReadCurrentByPower(var Currents:String);
{Reads Power's opinion of PresentCurrent}

Begin
  WriteCommand(power,'Measure:Current?');
  ReadValue(power,CurrentS);
  {If Pos(CurrentS,'A') > 0 then
    Delete(CurrentS,Pos(CurrentS,'A'),100)
  else Begin PrintErrors('Could not MEASURE Current at POWER-Device!!'); end;}
End;


Function InitGpibBus :boolean;
Var
 nb : nbuf;
 VoltageS, IDS, Temps: String;
 InitGpibBusd: boolean;

Begin
  GPIBBusTalkInProgress:=False;
  InitGpibBusD:=true;
  Mknbuf(brds,nb);
  brd:=IbFind(nb);
  If (IBErr <> 0) or (brd < 0) then
    begin PrintErrors('IBFind(board) failed'); InitGpibBusd:=false; end;
  IBInit(brd);
  If IBErr <> 0 then
    Begin PrintErrors('IBInit board failed'); InitGpibBusd:=false; end;
  IBCac(brd,1); {Become active Controler}
  If IBErr <> 0 then
    Begin
     PrintErrors('IBCAC board (become active controler and assert ATN) failed');
     InitGpibBusd:=false;
    end;
```

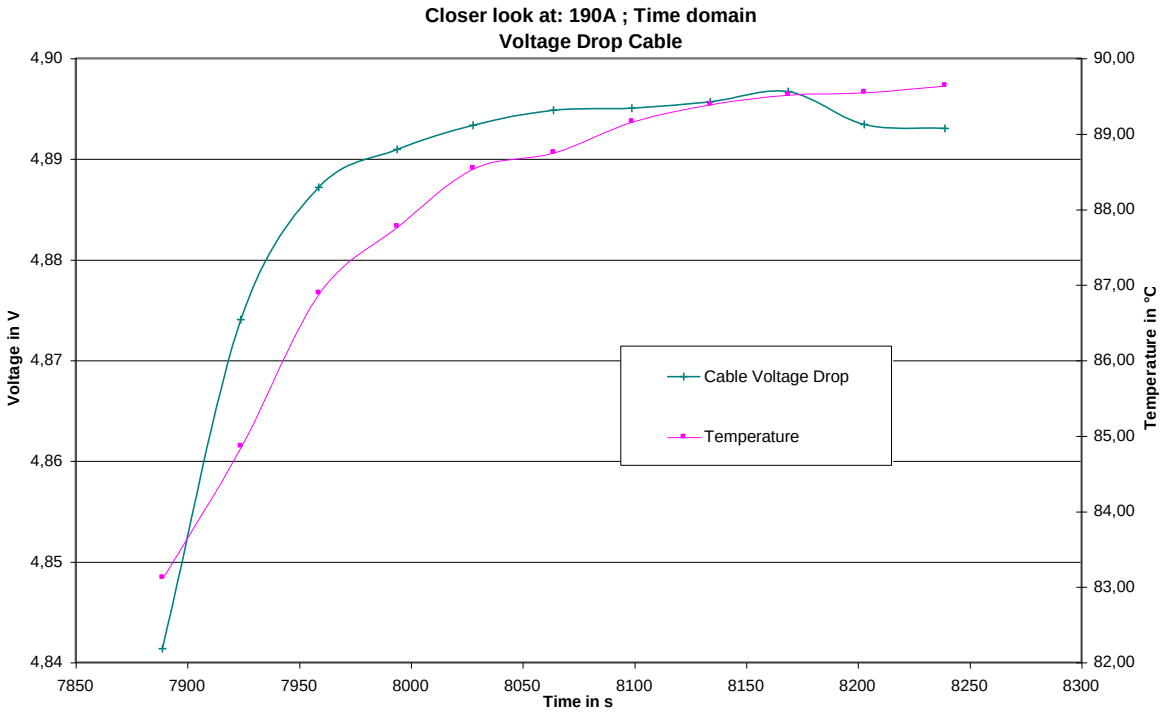
```
  Mknbuf(powers,nb);
  power:=IBFind(nb);
  If (IBErr <> 0) or (brd < 0) then
    Begin PrintErrors('IBFind(power) failed'); InitGpibBusd:=false; end;
  IBClr(power);
  WriteCommand(power,'Current 0');
  If IBErr <> 0 then InitGpibBusd:=false;
  WriteCommand(power,'Voltage 18.0');
  If IBErr <> 0 then InitGpibBusd:=false;
  WriteCommand(power,'Voltage?');
  If IBErr <> 0 then InitGpibBusd:=false;
  ReadValue(power,VoltageS);
  If IBErr <> 0 then InitGpibBusd:=false;
  If Pos('18',VoltageS) =0 then
    Begin
      PrintErrors('Could not set / read Voltage at POWER-Device!!');
      InitGpibBusd:=false;
    end;

  Mknbuf(dmms,nb);
  dmm:=IBFind(nb);
  If (IBErr <> 0) or (brd < 0) then
  Begin PrintErrors('IBFind(DMM) failed'); InitGpibBusd:=false; end;
  IBClr(dmm);
  WriteCommand(dmm,'VDR3A0T5L1Q0S1');
  If IBErr <> 0 then InitGpibBusd:=false;
  WriteCommand(dmm,'ID?');
  If IBErr <> 0 then InitGpibBusd:=false;
  ReadValue(dmm,IDS);
  If IBErr <> 0 then InitGpibBusd:=false;
  If Pos('DVM6001/SC',IDS) =0 then
  Begin
    PrintErrors('Could not set / read DMM-Device!!');
    InitGpibBusd:=false;
  end;
  If InitGpibBusD then GPIBReadTemp(Temps);
  GPIBBusTalkInProgress:=False;
  InitGpibBus:=InitGpibBusD
End;

Procedure DeInitGpibBus;
Begin
  GPIBBusTalkInProgress:=True;;
  {IBLoc(power); Go to Local}
  IBClr(power);
  IBLoc(dmm); {Go to Local}
  IBGTS(brd,0); {Go to standby}
  GPIBBusTalkInProgress:=False;
End;

Procedure DirectStop;
Begin
  GPIBBusTalkInProgress:=True;;
  IBClr(power);
  GPIBBusTalkInProgress:=False;
end;

END. {of Unit uGPIB}

written by Tom Roida 2001
```

IV. Examples of graphs and tables

The graphs and tables in this chapters shall only demonstrate the use of the output files of CPOW.



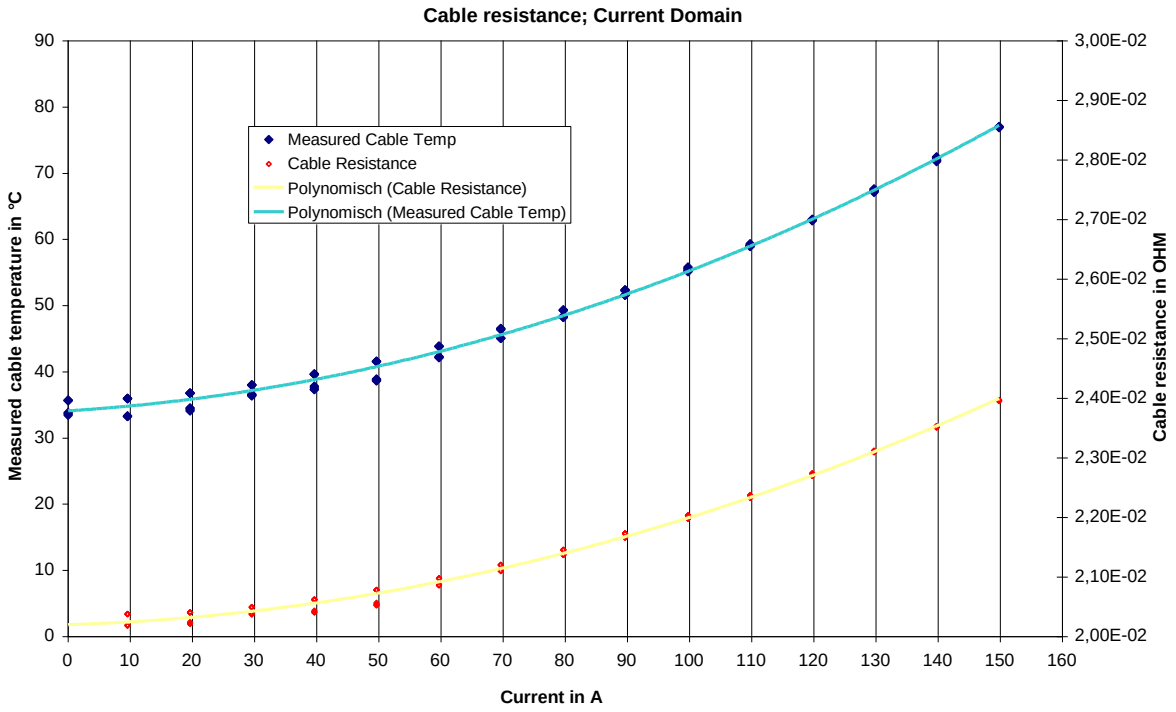


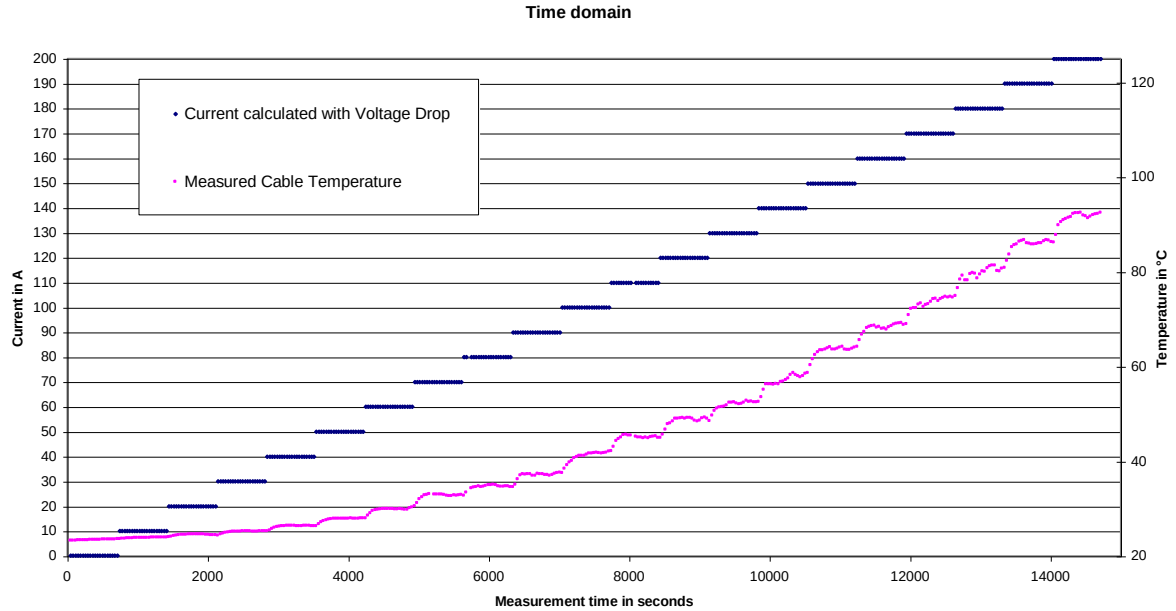


These two graphs show how the cable heated up when the current increased by 10A. The upper graph shows that the cable temperature was obviously not stable at the end, e.g. somebody cooled the cable by walking through the laboratory.

Measurement Time	Special Events	Measured Cable Temp	Wanted Current	Current measured by PowerSupply	calculated current	Cable Power	Shunt Voltage Drop	Cable Voltage Drop	Cable Resi
48	16:11 - Extra file initiated as: C:\CPOW\testfin.log								
48	16:11 - Environment temperature: 20.00000 deg C								
0	16:11 - User Tom Roida startet a new Measurement! Date: 2001-4-2								
16	16:11 - Setting new current to 0.00 A								
515		33,76	0	0,36	0,12250	0,00039	2,45000E-05	3,19790E-03	
616		33,61	0	0,36	0,12150	0,00039	2,43000E-05	3,19450E-03	
716		33,47	0	0,29	0,12300	0,00039	2,46000E-05	3,19430E-03	
816		33,29	0	0,29	0,12050	0,00038	2,41000E-05	3,19080E-03	
876	16:26 - Setting new current to 10.00 A								
1416		33,17	10	9,87	9,68050	1,89056	1,93610E-03	1,95296E-01	2,01741E-02
1516		33,17	10	9,87	9,67950	1,89034	1,93590E-03	1,95293E-01	2,01760E-02
1616		33,17	10	9,94	9,67950	1,89036	1,93590E-03	1,95295E-01	2,01762E-02
1716		33,13	10	9,94	9,67950	1,89017	1,93590E-03	1,95276E-01	2,01741E-02
1736	16:40 - Setting new current to 20.00 A								
2316		33,98	20	19,96	19,72500	7,85852	3,94500E-03	3,98404E-01	2,01979E-02
2416		34,15	20	19,96	19,72500	7,86177	3,94500E-03	3,98569E-01	2,02063E-02
2516		34,31	20	19,96	19,72400	7,86573	3,94480E-03	3,98790E-01	2,02185E-02
2615		34,46	20	19,96	19,72300	7,86993	3,94460E-03	3,99023E-01	2,02314E-02
2649	16:55 - Setting new current to 30.00 A								
3216		36,24	30	29,91	29,69550	17,95414	5,93910E-03	6,04608E-01	2,03603E-02
3315		36,33	30	29,98	29,69650	17,95839	5,93930E-03	6,04731E-01	2,03637E-02
3416		36,38	30	29,98	29,69550	17,96082	5,93910E-03	6,04833E-01	2,03678E-02
3516		36,46	30	29,98	29,69500	17,96533	5,93900E-03	6,04995E-01	2,03736E-02
3616		36,50	30	29,91	29,69300	17,96810	5,93860E-03	6,05129E-01	2,03795E-02
3616	17:11 - Setting new current to 40.00 A								
4316		37,73	40	40	39,74350	32,25896	7,94870E-03	8,11679E-01	2,04229E-02
4416		37,47	40	40	39,74350	32,23397	7,94870E-03	8,11050E-01	2,04071E-02
4516		37,27	40	40	39,74500	32,22366	7,94900E-03	8,10760E-01	2,03990E-02
4615		37,18	40	40	39,74700	32,21542	7,94940E-03	8,10512E-01	2,03918E-02
4636	17:28 - Setting new current to 50.00 A								
5315		38,61	50	49,94	49,72800	50,73365	9,94560E-03	1,02022E+00	2,05161E-02
5416		38,48	50	49,94	49,72700	50,72164	9,94540E-03	1,02000E+00	2,05120E-02
5516		38,62	50	50,02	49,72800	50,75503	9,94560E-03	1,02065E+00	2,05247E-02
5616		38,73	50	49,94	49,72750	50,78436	9,94550E-03	1,02125E+00	2,05370E-02
5716		38,83	50	49,94	49,72650	50,83431	9,94530E-03	1,02228E+00	2,05580E-02
5716	17:46 - Setting new current to 60.00 A								
6516		42,00	60	60,04	59,78150	74,50084	1,19563E-02	1,24622E+00	2,08462E-02
6616		42,07	60	60,04	59,78200	74,51892	1,19564E-02	1,24651E+00	2,08509E-02
6715		42,08	60	59,96	59,78100	74,53166	1,19562E-02	1,24675E+00	2,08552E-02
6816		42,17	60	60,04	59,78150	74,54066	1,19563E-02	1,24689E+00	2,08574E-02
6843	18:05 - Setting new current to 70.00 A								
7716		44,92	70	69,98	69,76700	102,64526	1,39534E-02	1,47126E+00	2,10882E-02
7816		44,94	70	69,98	69,76750	102,66071	1,39535E-02	1,47147E+00	2,10910E-02
8015		45,01	70	69,98	69,76800	102,67282	1,39536E-02	1,47163E+00	2,10932E-02
8023	18:25 - Setting new current to 80.00 A								
8916		48,16	80	80	79,81350	136,04219	1,59627E-02	1,70450E+00	2,13560E-02
9016		48,07	80	80	79,81350	136,03437	1,59627E-02	1,70440E+00	2,13548E-02
9116		48,31	80	80	79,81250	136,07361	1,59625E-02	1,70492E+00	2,13615E-02
9216		48,17	80	80,08	79,81300	136,06273	1,59626E-02	1,70477E+00	2,13595E-02
9256	18:45 - Setting new current to 90.00 A								
10216		51,48	90	89,95	89,74750	174,35920	1,79495E-02	1,94278E+00	2,16471E-02
10316		51,50	90	89,95	89,74750	174,40156	1,79495E-02	1,94325E+00	2,16524E-02
10416		51,45	90	89,95	89,74900	174,38087	1,79498E-02	1,94298E+00	2,16491E-02
10516		51,48	90	89,95	89,74800	174,38144	1,79496E-02	1,94301E+00	2,16496E-02
10543	19:07 - Setting new current to 100.00 A								
11516		54,99	100	99,97	99,86350	219,14446	1,99727E-02	2,19444E+00	2,19744E-02
11616		55,12	100	100,04	99,86500	219,07785	1,99730E-02	2,19374E+00	2,19671E-02
11716		55,14	100	100,04	99,86400	219,10461	1,99728E-02	2,19403E+00	2,19702E-02
11816		55,10	100	100,04	99,86600	219,06106	1,99732E-02	2,19355E+00	2,19649E-02
11883	19:29 - Setting new current to 110.00 A								
12922		58,88	110		109,84300	269,33833	2,19686E-02	2,45203E+00	2,23230E-02
13016		58,81	110	109,99	109,84450	269,30137	2,19689E-02	2,45166E+00	2,23194E-02
13116		58,81	110	110,06	109,84200	269,27327	2,19684E-02	2,45146E+00	2,23181E-02

This is an excerpt from a “secondary output file” generated by CPOW and converted to a Microsoft Excel® table.


Cable resistance; Current Domain
Measured Cable Temp
Cable Resistance
Polynomisch (Cable Resistance)
Polynomisch (Measured Cable Temp)
Measured cable temperature in °C
Cable resistance in OHM
Current in A
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
2,00E-02
2,10E-02
2,20E-02
2,30E-02
2,40E-02
2,50E-02
2,60E-02
2,70E-02
2,80E-02
2,90E-02
3,00E-02


Time domain
Current calculated with Voltage Drop
Measured Cable Temperature
Current in A
Temperature in °C
Measurement time in seconds
0
2000
4000
6000
8000
10000
12000
14000
20
40
60
80
100
120

V. Verwendete Formelzeichen

α_0	in 1/K	der lineare Temperaturkoeffizient des Leitermaterials bezogen auf die Referenztemperatur T_0
$ß_0$	in 1/K²	der quadratische Temperaturkoeffizient des Leitermaterials bezogen auf die Referenztemperatur T_0
T_0	in K	die Referenztemperatur, üblicherweise 20°C
T	in K	Temperatur allgemein, hier die Temperatur des Leiters (im temperaturkonstanten Bereich, d.h. nicht an den Leiterenden)
ΔT	in K	die Temperaturdifferenz $(T - T_0)$
$P = P_{el}$	in VA	die vom Leiter an die Umgebung abgegebene Leistung durch Eigenerwärmung
R	in Ω	der elektrische Widerstand des Leiters
R_0	in Ω	der elektrische Widerstand des Leiters bei der Referenztemperatur T_0
U	in V	Spannungsabfall am Leiter (zwischen den Meßleitungen)
I	in A	Strom durch den Leiter
R_S	in Ω	Shuntwiderstand
U_S	in V	Spannungsabfall am Shunt
α_m	in 1/K	der mittlere Wärmeübergangskoeffizient für Luft [1]

VI. Literatur

[1] Ließ, H.-D.: Bestimmung der Wärmeableitung von elektrisch beheizten Flachleitern, unveröffentlichter Entwurf vom Oktober 2000

[2] Mack, Patrick: Bestimmung des linearen und quadratischen Temperaturkoeffizienten ausgewählter elektrischer Leiter (Studienarbeit) , Universität der Bundeswehr München 2001

[3] Winkler, Arnulf: Elektrische Meßtechnik; Kamparth-Reihe kurz und bündig, Technik, Würzburg 1978 (5. Auflage)

[4] Schrüfer, E.: Elektrische Meßtechnik; Messung elektrischer und nichtelektrischer Größen, Wien 1992

V. Download von begleitenden Dateien

Zu dieser Arbeit gehört eine Floppy Disk mit dem Programm CPOW als ausführbare (*.exe) Datei sowie der Quellcode. Das Programm läuft unter DOS, der Code wurde in PASCAL geschrieben.

Die Dateien können unter folgender Internetadresse heruntergeladen werden:

www.roida.de/science