Diplomarbeit

Meßdatenerfassungssystem für Remote-Laboratory-Anwendungen

Ausgeführt am Institut für

Industrielle Elektronik und Materialwissenschaften

der Technischen Universität Wien

Betreut von
Dr. Karl Riedling

durch
Johann Smejkal
9325606

2136 Laa/Thaya, Hanfthal 124
1150 Wien, Jheringgasse 13/2/14

2. Oktober 2001

Vorwort

Dieses Dokument beschreibt den Aufbau und die Verwendungsweise des entwickelten Meßdatenerfassungssystems. Programmierbare Meßgeräte können durch dieses System eingestellt und abgefragt werden. Die erhaltenen Werte werden auf einer Webseite grafisch dargestellt.

Ich möchte mich bei Professor Riedling und seiner Frau herzlich für die Ermöglichung dieser Arbeit und ihre Unterstützung bedanken. Ebenso möchte ich meinen Eltern danken, die mir über die Jahre hinweg dieses Studium ermöglicht haben.

Inhaltsverzeichnis

1. PROBLEMSTELLUNG 1

2. LÖSUNGSANSATZ 2

3. DAS DOS-PROGRAMM MM.EXE 7

3.1 KONFIGURATION MITTELS MM.CFG 7
3.1.1 ABSCHNITT MM 7
3.1.2 ABSCHNITT ERROR 8
3.2 BEFEHLSZEILEN-PARAMETER 9
3.3 INITIALISIERUNGSDATEIEN .INI 10
3.3.1 ABSCHNITT METER 10
3.3.2 ABSCHNITT PARAMETERS 16
3.3.3 ABSCHNITT CONSTANTS 17
3.3.4 ABSCHNITTE STREAM 19
3.3.5 ABSCHNITT COMMANDS 21
3.3.6 SPEZIELLE ABSCHNITTE 24
3.4 BEFEHLSDATEIEN .CMD 25
3.4.1 SPEZIELLE BEFEHLE 25
3.4.1.1 Befehl GOTO 25
3.4.1.2 Befehl DUMP 26
3.4.1.3 Befehl IF 26
3.4.1.4 Befehl CALL 28
3.4.1.5 Befehl SLEEP 29
3.4.1.6 Befehl WAIT 29
3.4.1.7 Befehl SET 30
3.4.1.8 Befehl OUT 30
3.5 ANWENDUNGSMÖGLICHKEITEN 31
3.5.1 MM ALS SERVER 32
3.5.2 MM ALS BEFEHLSZEILENPROGRAMM 32

4. DIE WEB-BASIERTE OBERFLÄCHE 33

4.1 KONFIGURATION 33
4.2 AUFBAU 34
4.2.1 BEFEHLSEINGABE 34
4.2.2 ERGEBNISANSICHT 37
4.2.3 STATUSANSICHT 41
4.2.4 HILFEANSICHT 41

4.3 UNTERSTÜTZTE BROWSER 41
4.4 BEKANNTE PROBLEME 42

5. MM FÜR DEN PROGRAMMIERER 45

5.1 ALLGEMEINER AUFBAU 45
5.2 FEHLERBEHANDLUNG 47
5.3 OBJEKT EINER DATEI 48
5.3.1 OBJEKT EINER INITIALISIERUNGSDATEI 51
5.3.2 OBJEKT EINER BEFEHLSDATEI 52
5.4 OBJEKT EINES PORTS 53
5.4.1 OBJEKT EINES SERIELLEN COM PORTS 55
5.4.2 OBJEKT EINES IEEE PORTS 55
5.4.3 OBJEKT EINES USB PORTS 55
5.5 ERWEITERUNG DES PROGRAMMS MIT ZUSÄTZLICHEN PORTS 56
5.5.1 HINZUFÜGEN EINES NEUEN PORT-OBJEKTES 56
5.5.2 IMPLEMENTIERUNG DER FUNKTIONEN 56
5.5.3 EINBINDEN IN DAS HAUPTPROGRAMM 57

6. BEISPIEL 58

6.1 DIE MEßGERÄTE 58
6.2 DER MEßRECHNER 58
6.3 DER WEBSERVER 58
6.3.1 INITIALISIERUNGSDATEI GAUSS.INI 58
6.3.2 INITIALISIERUNGSDATEI OSCI.INI 61
6.3.3 KONFIGURATIONSDATEI INIT.PHP FÜR DIE WEBOBERFLÄCHE 69
6.3.4 KONFIGURATIONSDATEI MM.CFG FÜR DAS DOS-PROGRAMM 70
6.3.5 VORDEFINIERTE BEFEHLSDATEIEN 70
6.3.5.1 Include-Datei GAUSSAC.INC 70
6.3.5.2 Include-Datei WAVE1.INC 72

7. AUSBLICK 75

8. VERWENDETE HILFSMITTEL 76

1. Problemstellung

Manche Meßgeräte müssen in einer bestimmten Umgebung betrieben werden, die oft für den Menschen nicht leicht zugänglich ist. Ein Gaußmeter[1] sollte in einen Faraday[2]-Raum betrieben werden, damit bei der Messung möglichst alle störenden Magnetfelder abgeschirmt werden, welche die Ergebnisse stark verfälschen würden. Damit ist der Aufstellungsort eines solchen Meßgerätes sehr beschränkt, und wenn jemand eine Messung durchführen will, muß er diesen Ort aufsuchen.
Ein weiteres Problem stellt die Ausbildung an einem modernen Meßgerät dar. Die Anzahl der Geräte beschränkt die Anzahl der Studenten, die gleichzeitig Messungen durchführen können. In der Praxis sieht es dann meistens so aus, daß 5 oder mehr Studenten an einem Meßplatz agieren. Damit bedient einer das Meßgerät, und die anderen schauen zu. Somit haben diese aber so gut wie keinen Lerneffekt.
Moderne Meßgeräte verfügen über eine Rechnerschnittstelle, über die sie programmiert werden können. Daher liegt es nahe, die Messungen über diese Schnittstellen zu steuern. Jedem Meßgerät liegt eine Dokumentation zur Programmierung bei. Meistens wird hierfür BASIC verwendet. Das ist jedoch nicht sehr flexibel und erfordert neue Programme für jedes Meßgerät. Ein Anwender schreibt also für seine Zwecke kleine Programme, die aber sein Kollege kaum verwenden kann. Dieser muß nun von vorne beginnen und seine eigenen Programme erstellen. Damit wird die Handhabung eines solchen Meßgerätes sehr inflexibel.
Vorhandene Lösungen auf diesem Gebiet sind entweder teuer oder erfordern hohe Hardware- und Softwarevoraussetzungen.

[1] 1 Gauß (G) ist die Maßeinheit der magnetischen Induktion (Feldliniendichte oder magnetische Flußdichte)
[2] Michael Faraday, engl. Naturforscher, *1791, entdeckte u.a. die elektrische Induktion

2. Lösungsansatz

Die dieser Arbeit zu Grunde liegende Idee war nun, ein allgemeines Steuerprogramm für beliebige Meßgeräte zu entwerfen, welches durch eine grafische Web-basierte Oberfläche bequem bedient werden kann.
An das Meßgerät wird ein Rechner angeschlossen. Es werden nicht zu hohe Anforderungen gestellt, somit genügt ein alter ausgedienter 486. Die Verbindung des Rechners zum Meßgerät kann über die serielle Schnittstelle, den IEEE-488-Bus oder andere DOS-Gerätetreiber erfolgen.
Auf diesem Rechner wird ein DOS-Programm ausgeführt, das Befehle über die Schnittstelle zum Meßgerät sendet und Ergebnisse empfängt. Alle Eigenschaften eines Meßgerätes werden in einer Initialisierungsdatei zusammengefaßt. Jedem Meßgerät wird genau eine solche Beschreibungsdatei zugeordnet. Das erlaubt die einfache Modularisierung des kompletten Systems. Neue Meßgeräte können zusätzlich angeschlossen werden, und es muß für sie eine neue Initialisierungsdatei erstellt werden. An den vorhandenen Dateien muß keine Änderung erfolgen. Es werden automatisch alle gefundenen Initialisierungsdateien mit der Erweiterung .INI verwendet.
Eine solche Initialisierungsdatei beinhaltet alle notwendigen Angaben, um das Meßgerät zu verwenden. Dazu gehören die Bezeichnung der Schnittstelle mit ihren Parametern und die zur Verfügung stehenden Befehle. Es werden alle Meßgeräte unterstützt, die Befehle als Zeichenketten erwarten. Diese Zeichenketten sind oft kryptische Ziffernfolgen und können in der Initialisierungsdatei auf leicht merkbare Meta-Befehle abgebildet werden. Ebenso werden Parameter unterstützt, die sowohl an das Meßgerät gesendet werden, als auch als Ergebniswert vom Meßgerät geliefert werden. Um diese Ergebnisse zu speichern, beinhaltet die Initialisierungsdatei Parameterdeklarationen. Das sind Speicherbereiche, die beliebige Werte annehmen können und durch einen eindeutigen Namen ansprechbar sind. Eine Initialisierungsdatei unterstützt auch die Ausgabe von Daten in eine externe Datei. Ein solcher Datenstrom kann mit Befehlen in eine Datei geschrieben werden, oder auch automatisch, falls das Meßgerät einen Datenstrom durch Trennzeichen abgegrenzt liefert. Damit ist das aufgebaute System offen gegenüber anderen Anwendungen, und die erhaltenen Daten können beliebig weiterverarbeitet werden. Das Format der Daten kann selbst bestimmt werden, es handelt sich dabei allerdings immer um eine Textdatei. Die Struktur dieser Textdatei kann mittels der Initialisierungsdatei genau nach Wunsch beschrieben werden. Es können auch mehrere verschiedene Formate gleichzeitig verwendet werden. Die Initialisierungsdateien sind modular aufgebaut und können auf einfache Weise neuen Bedingungen angepaßt werden. Die Erweiterung mit neuen Möglichkeiten bei gleichbleibender Nutzung der vorhandenen Einstellungen gewährleistet Kompatibilität mit alten bereits erstellten Befehlsdateien. Neue Befehlsdateien können dann die erweiterte Funktionalität ohne Probleme nutzen.
Eine Befehlsdatei beinhaltet alle notwendigen Befehle, um Meßdaten zu erhalten. Durch Steuerelemente kann der lineare Ablauf einer solchen textbasierten

Befehlsdatei geändert werden. Es sind bedingte Sprünge möglich, ebenso steht eine relative Zeit zur Verfügung, damit die Messungen in bestimmten Zeitintervallen durchgeführt werden können. Diese Befehlsdateien können auch weitere Befehlsdateien einbinden, womit ein modularer Aufbau einer komplexen Messung möglich ist. Befehle, die ein Meßgerät in einen definierten Zustand versetzen, können somit in einer eigenen Datei gespeichert werden und dann immer wieder verwendet werden, ohne die Datei mit der eigentlichen Messung unübersichtlich groß zu machen. Eine Befehlsdatei ist nicht wie eine Initialisierungsdatei notwendigerweise an ein bestimmtes Meßgerät gebunden. Es können beliebige Meßgeräte aus einer Befehlsdatei in serieller Abfolge angesprochen werden. Somit kann in einer Befehlsdatei ein Funktionsgenerator auf eine bestimmte Ausgangsfunktion eingestellt werden. Dieses Signal wird an den Eingang einer zu messenden Testschaltung gelegt, und der Ausgang dieser Schaltung wird mit dem Eingang eines digitalen Oszilloskops gelegt. Damit kann nun mit einer Befehlsdatei die Versorgungsspannung der Testschaltung definiert werden und danach die Ergebniskurve vom Oszilloskop abgefragt werden.
Diese Befehlsdateien können mit Hilfe eines beliebigen Texteditors vom Anwender erstellt werden und dem DOS-Programm MM.EXE als Parameter übergeben werden. Das Programm wertet dann diese Datei aus und erstellt gegebenenfalls eine Datei mit den Meßergebnissen, die sich mit anderen Anwendungen weiterverarbeiten läßt.
Eine weitere Anwendungsmöglichkeit stellt eine grafische Webseite dar. Diese ersetzt den Texteditor zum Erstellen von Befehlsdateien durch ein Eingabefeld innerhalb einer Webseite und erlaubt das Ausführen der eingegebenen Befehlsfolge durch einfaches Klicken auf eine Schaltfläche.
Auf dem Webserver wird PHP3 verwendet, um die Webseiten dynamisch aufzubauen. Die Webseiten verwenden dieselben Initialisierungsdateien wie das DOS-Programm MM.EXE. Sie bieten daher eine einfache Auswahlmöglichkeit der verfügbaren Befehle und Parameter. Somit kann, ohne die Zuhilfenahme der Tastatur, eine komplette Befehlsdatei erstellt werden. Diese Befehlsdatei wird von der Webseite in ein Verzeichnis am Webserver gestellt. Das DOS-Programm MM.EXE überprüft dieses Verzeichnis regelmäßig auf neue Befehlsdateien. Falls es eine Datei findet, werden diese Befehle ausgeführt und der Webseite der Fortschritt der Abarbeitung mitgeteilt. Dieses Arbeitsverzeichnis befindet sich auf dem Webserver. Damit wird einer Blockierung des Servers durch einen fehlerhaften Meßrechner vorgebeugt. Der Meßrechner muß über das Netzwerk mit dem Webserver verbunden sein. Daher ist es sinnvoll, auf dem Meßrechner ein Windows-System laufen zu lassen, das die Netzwerksoftware bereits beinhaltet. Damit werden Netzwerklaufwerke auf einen Laufwerksbuchstaben abgebildet. Auch ermöglicht ein Windows der 95-er Reihe die Verwendung von langen Dateinamen innerhalb des Steuerprogramms auf DOS-Ebene. Diese Funktionen stehen im reinen DOS-Modus nicht zur Verfügung. Ist am Meßrechner aber die grafische Oberfläche geladen, so können auch DOS-Programme diese neuen Funktionen verwenden. Das ist deshalb wichtig, da der Webserver ein UNIX-basiertes System verwendet und dieses die Groß/Kleinschreibung von Dateinamen berücksichtigt. Mit den alten DOS-Funktionen wird eine Datei immer in Großbuchstaben angelegt. Greift nun die Webseite mit einem PHP-Skript auf die normalerweise übliche Kleinschreibung eines

Dateinamens zu, so wird diese Datei nicht gefunden. Werden nun die neuen Funktionen verwendet, wird die Datei bereits richtig in Kleinschreibung erzeugt. DOS-Programme unterschieden Groß/Kleinschreibung allerdings nicht.
Der Meßrechner greift auf 2 Freigaben des Webservers zu. Das ist deswegen notwendig, da auch schreibend auf das Verzeichnis mit den Befehlsdateien zugegriffen werden muß. Aus Sicherheitsgründen wird somit das Verzeichnis mit den Initialisierungsdateien nur lesbar freigegeben, und das Verzeichnis mit den Befehlsdateien als schreibbare Freigabe zugänglich gemacht. Die nur lesbare Freigabe enthält auch die Programmdatei MM.EXE des DOS-Programmes selbst. Die zweite Freigabe muß für den Meßrechner beschreibbar sein und dient als Schnittstelle der Webseite, die die Befehle liefert, und des DOS-Programms, welches die Ergebnisse in dieses Verzeichnis stellt. Dadurch, daß das DOS-Programm direkt vom Webserver aus geladen wird, ist sichergestellt, daß immer die aktuellste Version verwendet wird, und man erspart sich das Kopieren von Dateien auf den Meßrechner. Es erleichtert ebenso die Wartbarkeit. Ein laufendes System kann ohne den Meßplatz aufzusuchen verändert werden. Ein weiterer Vorteil ergibt sich, wenn mehrere Meßrechner verwendet werden. Es ist genauso möglich, für jedes Meßgerät einen eigenen Rechner zu verwenden, als auch mehrere Meßgeräte an einen Rechner anzuschließen. Ist nun das DOS-Programm MM.EXE zentral auf dem Webserver gespeichert, ist sichergestellt, daß alle Meßrechner immer die gleiche Version verwenden. Durch Nutzung von mehreren Schnittstellen können an einen Meßrechner eine große Anzahl von Geräten angeschlossen werden. Ein DOS-Rechner unterstützt bis zu 4 serielle Schnittstellen. Durch den zusätzlichen Einbau einer IEEE-488-Steckkarte können 14 weitere Geräte angeschlossen werden. Diese Vielzahl von Anschlußmöglichkeiten ist sinnvoll, wenn es sich um ein ferngesteuertes Labor handelt, in dem eine Reihe von Spannungsquelle und Aktoren vorhanden sind. Eine Instanz von MM kann nun alle diese Geräte an einem Rechner bedienen. Die Befehle werden aber der Reihe nach an die verschiedenen Meßgeräte geschickt. Deshalb sollte dieser Anwendungsfall nur in Betracht gezogen werden, wenn die verwendeten Meßgeräte miteinander in Beziehung stehen, und sowieso davon abhängig sind, was die anderen Meßgeräte liefern. Sind aber unterschiedliche Meßgeräte an einen PC angeschlossen, empfiehlt es sich, für jedes dieser Meßgeräte ein MM-Programm laufen zu lassen. Diese Geräte können in Gruppen zusammengefaßt werden. Dann wird jeder Gruppe genau ein MM-Programm zugeordnet. Den schnellsten Ablauf erreicht man natürlich, wenn für jedes Meßgerät ein eigener PC zur Verfügung steht, und jedes Meßgerät von einem MM angesteuert wird.
Der Ablauf einer Messung besteht somit im Definieren von Befehlsabläufen innerhalb der Webseite. Die eingegebenen Befehle können am lokalen Rechner zur späteren Wiederverwendung gesichert werden. Alle ausgeführten Befehlsdateien bleiben auch am Server gespeichert und können daher bei späterer Einsicht wiederverwendet werden. Ist die Befehlssequenz fertig definiert, so werden diese Befehle in eine Datei in das Arbeitsverzeichnis auf dem Webserver geschrieben, auf die das DOS-Programm zugreift, diese auswertet und Befehle an das Meßgerät schickt, die von diesem ausgeführt werden.

Das DOS-Programm wertet die Befehlsdatei zeilenweise aus. Eine Zeile kann einen Steuerbefehl oder einen Meßbefehl enthalten. Handelt es sich um einen Steuerbefehl, werden keine Daten an das Meßgerät geschickt. Diese Befehle beeinflussen die weitere Folge der Befehlsabarbeitung. Handelt es sich um einen Meßbefehl, wird dieser an Hand einer Liste in der zugehörigen Initialisierungsdatei und weiteren Einstellungen in eine für das Meßgerät verständliche Form übersetzt. Nach dem Senden eines solchen Meßbefehls wird in der Initialisierungsdatei überprüft, ob dieser Befehl eine Antwort vom Meßgerät erwartet oder nicht. Diese Antwort besteht oft aus Parametern, die vom Meßgerät abgefragt werden. In der Initialisierungsdatei kann das Format dieser Antwort genau bestimmt werden, womit die einzelnen Parameter extrahiert werden können. Um diese Werte dann zu speichern, müssen in der Befehlsdatei Parameter angegeben worden sein.
Bereits während der Messung stellt das DOS-Programm die Ergebnisse auf den Webserver, falls Daten zurückgeliefert wurden. Dadurch können die vorhandenen Meßdaten bereits kurze Zeit nach dem Start der Messung betrachtet werden. Der Fortschritt über die laufende Messung gibt Bescheid, ob die Messung gerade noch läuft oder bereits beendet worden ist. Auch bleiben diese Ergebnisse am Server gespeichert und können mit anderen Programmen weiterverarbeitet werden. Diese Ergebnisse werden in einer Textdatei gespeichert, welche die Daten durch Beistriche trennt. Somit sollte es leicht möglich sein, diese Ergebnisse mit vielen anderen Anwendungen zu verwenden.
Die Trennung in Initialisierungs- und Befehlsdateien dient einer einfachen Übersichtlichkeit. Einzelne Dateien können leicht ausgetauscht werden, ohne den Ablauf zu stören. Durch einen Mechanismus können sogar im laufenden Betrieb die Einstellungen in den Initialisierungsdateien geändert oder neue hinzugefügt werden. Der Anwender wird von diesem Vorgang nichts merken, falls er nicht gerade auf Werte zugreifen will, die in der neuen Initialisierungsdatei nicht mehr vorhanden sind. Das Programm liest die Initialisierungsdateien einmalig beim Start und behält alle relevanten Daten im Speicher. Durch das Erzeugen einer Datei mit dem Namen MM.REL (für reload) im Arbeitsverzeichnis des Programms wird ein neuerliches Einlesen der Initialisierungsdateien erzwungen.
Das DOS-Programm MM.EXE arbeitet streng linear. Es kommt immer die nächste gefundene Befehlsdatei an die Reihe. Erst wenn diese Datei komplett abgearbeitet ist, bedient MM die nächste Anforderung. Es werden also keine Befehlsdateien geschachtelt. Da die Meßgeräte viele Einstellmöglichkeiten haben, wäre das auch nicht sinnvoll. Durch diesen Mechanismus kann aber ein böswilliger Zeitgenosse ein Meßgerät für beliebig lange Zeit blockieren. Er könnte theoretisch auch eine Endlosschleife in seiner Befehlsdatei abarbeiten lassen. Alle Meßvorgänge werden aber protokolliert, und die Webseite zeigt zusätzlich den gerade aktiven und alle wartenden Meßaufträge an. Durch die Eingabe eines Kennwortes können laufende Messungen gestoppt und gelöscht werden. Die Ausführungszeit für eine Befehlsdatei kann aber auch generell beschränkt werden. Damit kann vermieden werden, daß eine einzelne Messungen das gesamte System blockieren.
Die Initialisierungsdateien sind nicht als ein abgeschlossenes System zu verstehen. Sie bilden lediglich einen Vorschlag für die Verwendung und können je nach Anwendungsfall erweitert oder geändert werden. Auch ist es somit möglich,

bestimmte Befehle kurzzeitig auszublenden. Das kann in einer Laborübung interessant sein, wobei die Studenten Messungen durchführen sollen, aber nicht alle zur Verfügung stehenden Befehle verwenden sollen.
Zur Vermeidung von Tippfehlern in Befehlsdateien ist es nicht möglich, Befehle direkt an das Meßgerät zu schicken. Das HP-Oszilloskop sendet beispielsweise auf die meisten Befehle keine Antworten zurück. Somit hat das DOS-Programm keine Möglichkeit, die Richtigkeit eines direkt eingegeben Befehls zu verifizieren. Es müssen daher alle Befehle, die verwendet werden sollen, in der Initialisierungsdatei beschrieben werden. Ebenso ist es notwendig, für jede Antwort eine Maske in dieser Datei zu definieren. Andere Meßgeräte z.B. das BELL Gaußmeter schicken wiederum immer die gesendeten Befehle zur Fehlererkennung zurück. Damit kann MM die gültige Übertragung schnell erkennen und im Fehlerfall die Befehle noch einmal senden.
Eine Instanz von MM kann entweder alle Meßgeräte oder eine Gruppe von Meßgeräten bedienen. Bei einem Aufbau mit mehreren voneinander abhängigen Meßgeräten oder Aktoren kann durch eine Gruppierung ein definierter Ablauf erreicht werden. Mit einer Befehlsdatei können dann die Meßgeräte dieser Gruppe der Reihe nach bearbeitet werden, ohne daß sie durch andere Meßdateien unterbrochen werden. Enthält eine Befehlsdatei Anweisungen für mehrere Meßgeräte die sich nicht innerhalb einer Gruppe befinden, also von mehreren MM-Instanzen verwaltet werden, ist nicht sichergestellt, daß die Abarbeitung dieser Befehlsdatei durch andere Benutzer unterbrochen werden kann. Deshalb sollte man Meßgeräte, die miteinander verbunden sind, auch in einer Gruppe von einem MM-Programm verwalten lassen. Dieses Verhalten entsteht, da MM alle ankommenden Steuerdateien in einem Verzeichnis beachtet. Hat man nun zwei Meßgeräte mit jeweils einem MM-Programm, so wird bei Ausführung einer Befehlsdatei mit Anweisung an beide Meßgeräte zuerst das MM für Meßgerät Nummer 1 an die Reihe kommen und nach erledigter Arbeit das MM für das Meßgerät Nummer 2 informieren. Danach wird MM für Meßgerät Nummer 1 gleich die nächste für ihn bestimmte Befehlsdatei ausführen.
In den folgenden Kapiteln werden die Details zum Aufbau des DOS-Programms aufgezeigt, die verwendeten Initialisierungsdateien mit allen ihren möglichen Einträgen beschrieben und der Aufbau einer Befehlsdatei analysiert.
Das Kapitel über die Webseite soll einen Einblick über deren Verwendbarkeit und möglichen Ausbau geben.
Ein Kapitel widmet sich der Programmierung des DOS-Programms, und es zeigt deutlich die Modularisierung der verwendeten Objekte.
Zuletzt soll an Hand der auf dem Institutsserver gespeicherten Installation ein Beispiel besprochen und der Ablauf erklärt werden.

3. Das DOS-Programm MM.EXE

Warum benötigt man überhaupt ein DOS-Programm? Ein Windows-Betriebssystem kann (noch) nicht auf die Schnittstellen eines benachbarten Rechners im Netzwerk zugreifen. Deswegen wird auf die Dateischnittstelle zurückgegriffen. Diese Anwendungsart entlastet auch den Webserver, da die Messung von einem lokalen Rechner durchgeführt wird. Ein gemeinsames Verzeichnis dient dem Webserver und dem Meßrechner als Kommunikationsplattform. Der Webserver braucht nur Befehle in dieses Verzeichnis zu stellen, die der Meßrechner dann verarbeitet und die Ergebnisse bereitstellt. Das Verzeichnis selbst befindet sich auf dem Webserver. Somit bereitet ein blockierter Meßrechner keine Probleme für den Ablauf, da der Webserver natürlich noch andere Aufgaben zu erledigen hat. Das Programm auf dem Meßrechner übernimmt die Kommunikation über die Schnittstellen des PCs. Das Programm ist in objektorientiertem PASCAL geschrieben. Details dazu im letzten Kapitel.

3.1 Konfiguration mittels mm.cfg

Das Programm MM.EXE muß wissen, wo es seine Daten finden kann. Das kann mittels einer Initialisierungsdatei eingestellt werden. Diese Datei hat den allgemeinen Aufbau einer INI-Datei mit Abschnitten in eckigen Klammern. Kommentare können durch ein vorangestelltes Strichpunkt-Zeichen „;“ gekennzeichnet werden. Diese Datei kann die Abschnitte [MM] und [ERROR] enthalten.

3.1.1 Abschnitt MM

Dieser Abschnitt dient zum Einstellen der Verzeichnisse und allgemeiner Programmparameter. Folgende Einträge sind möglich:

INIDIR
Dieser Eintrag enthält eine Zeichenkette, die den vollständigen oder relativen Pfad zu den Initialisierungsdateien der Meßgeräte aus der Sicht von MM.EXE angibt. Wenn dieser Eintrag fehlt, wird das Verzeichnis verwendet, in dem die Programmdatei MM.EXE steht. Ein relativer Pfad beginnt in dem Verzeichnis, das die Programmdatei MM.EXE enthält.

CMDDIR
Dieser Eintrag enthält eine Zeichenkette, die den vollständigen oder relativen Pfad zum Arbeitsverzeichnis für Befehlsdateien aus der Sicht von MM.EXE angibt. Wenn

dieser Eintrag fehlt, wird das Verzeichnis verwendet, in dem die Programmdatei MM.EXE steht. Ein relativer Pfad beginnt in dem Verzeichnis, das die Programmdatei MM.EXE enthält. In dieses Verzeichnis legt die Webseite Befehlsdateien und das Programm MM.EXE Ergebnisdateien ab.

SERVERMODE
Dieser Eintrag enthält eine Zahlenangabe, die das Verhalten des Programms als Server beschreibt. Zur Zeit sind die Werte 1 und 2 gültig. Mit dieser Einstellung kann die Kommunikation mit der Webseite gesteuert werden. Details werden im Kapitel „MM als Server“ dargestellt. Wenn dieser Eintrag fehlt, wird 1 angenommen.

LOGFILE
Dieser Eintrag enthält eine Zeichenkette, die den Namen einer Protokolldatei angibt. Alle durchgeführten Messungen und eventuell auftretende Fehler werden in dieser Datei gespeichert. Fehlt dieser Eintrag, wird kein Protokoll angelegt.

Ein Beispiel:

```
[MM]
INIDIR=W:\HTTPD\html
CMDDIR=X:\
SERVERMODE=1
LOGFILE=mm.log
```

Hier wird das Verzeichnis mit den Initialisierungsdateien auf dem Webserver verwendet, wo sich auch die Webseiten befinden. Als Arbeitsverzeichnis wird die zweite Freigabe auf dem Webserver verwendet. W: und X: können mit dem Windows Netzwerk verbunden werden und bei jedem Einloggen wiederverwendet werden. Als Protokolldatei wird der Name „mm.log“ gewählt, der ohne Pfadangabe im Arbeitsverzeichnis (CMDDIR) angelegt wird.

3.1.2 Abschnitt ERROR

Diese Abschnitt dient zur Definition, wie Fehler behandelt werden sollen. Er kann die folgenden Einträge enthalten:

FILE
Hier kann der Name einer Datei angegeben werden, in die auftretende Fehler geschrieben werden sollen. Fehlt dieser Eintrag, wird automatisch der Name vom zugehörigen Befehlsdateinamen abgeleitet (mit der Erweiterung .ERR). Aus einer Befehlsdatei C2.CMD wird also eine Fehlerdatei mit dem Namen C2.ERR.

HEAD
Diese Zeile wird beim Öffnen der Fehlerdatei unverändert an den Anfang der Datei geschrieben.

LINE
Dieser Eintrag enthält das Format, wie eine Fehlermeldung des DOS-Programms in die Fehlerdatei eingetragen werden soll. Er sollte ein Fragezeichen (?) enthalten, welches durch den Fehlertext ersetzt wird. Enthält der Eintrag kein Fragezeichen, so wird die Fehlermeldung anstelle diese Eintrages in der Datei gespeichert.

TAIL
Diese Zeile wird beim Schließen der Fehlerdatei unverändert noch hinzugefügt

Ein Beispiel:

```
[ERROR]
LINE=MM: ?
```

Hier wird der Dateiname der Befehlsdatei verwendet und nur eine Zeile eingetragen. Wird bei LINE kein Fragezeichen angegeben, so wird nur die wirkliche Fehlermeldung eingetragen und die Angabe von LINE ignoriert. Ein Anwendungsbeispiel für HEAD und TAIL wäre z.B. das Erzeugen einer HTML-Datei, die eine Fehlermeldung anzeigt.

```
[ERROR]
FILE=error.htm
HEAD=<html>
LINE=<body onload="alert('?');"></body>
TAIL=</html>
```

Dieses Format wird auch für die Ausgabe der Ergebnisse verwendet. Mehr dazu im Kapitel „Spezielle Befehle: OUT“.

3.2 Befehlszeilen-Parameter

Das Programm MM.EXE kann auch über die Eingabeaufforderung aufgerufen werden. Mit Befehlszeilenparametern können die Einstellungen in der Initialisierungsdatei mm.cfg überschrieben werden.

Die Syntax lautet:

```
MM [name ...] [/Iverzeichnis] [/Cverzeichnis]
[/Mx] [/S] [/Agruppenname]
```

Für name können beliebig viele Befehlsdateien angegeben werden, die dann der Reihe nach ausgeführt werden. Man beachte, daß DOS die Anzahl der Zeichen für einen Programmaufruf begrenzt; damit ist auch die Anzahl der möglichen Dateinamen begrenzt.

Der Schalter /I erlaubt mit der Angabe eines absoluten oder relatives Verzeichnisses, dieses als neues Initialisierungsverzeichnis zu benutzen. In diesem Verzeichnis werden die Initialisierungsdateien für die einzelnen Meßgeräte gesucht. Die Dateinamenserweiterung sollte .INI lauten.

Der Schalter /C erlaubt mit der Angabe eines absoluten oder relativen Verzeichnisses, dieses als neues Arbeitsverzeichnis zu benutzen. In diesem Verzeichnis werden die von der Webseite stammenden Befehlsdateien erwartet, falls MM als Server gestartet wurde. MM wird als Server gestartet, wenn keine Dateinamen angegeben wurden. Werden ein oder mehrere Dateinamen angegeben, so werden diese ausgeführt und MM beendet.

Der Schalter /M erlaubt das Verhalten von MM als Server zu beeinflussen. Mögliche Werte sind hier zur Zeit 1 und 2. Details zum Serververhalten im Kapitel „MM als Server“.

Der Schalter /S bringt MM dazu, keine Meldungen über gesendete Befehle und empfangene Antworten auszugeben. Normalerweise zeigt MM alle Befehle, die es an ein Meßgerät schickt, sowie alle empfangenen Antworten auf dem Bildschirm an. Dies kann aber beim Serverbetrieb unnötig sein und erfordert natürlich auch Ausführungszeit. Deshalb kann mit diesem Schalter die Ausgabe abgeschaltet werden (S = silent). Fehlermeldungen werden natürlich immer noch angezeigt.

Der Schalter /A schaltet den MM Server in den Modus, daß er nur mehr die einer Gruppe zugeordneten Meßgeräte bedient. Normalerweise beantwortet ein laufendes MM jede im Arbeitsverzeichnis ankommende Befehlsdatei. Mit diesem Schalter kann man mehrere Meßgeräte jeweils mit einem eigenem MM, wenn gewünscht auch mit eigenem Rechner, laufen lassen. Dann verwendet ein MM nur Befehle, die an die angegebenen Meßgerät geschickt werden sollen, und ignoriert alle anderen.

3.3 Initialisierungsdateien .INI

Die Initialisierungsdateien stellen den Mittelpunkt eines Meßsystems dar. Eine Initialisierungsdatei beschreibt genau ein Meßgerät und besteht aus Abschnitten und Werteeinträgen. Kommentare können an jeder beliebigen Stelle mit Hilfe eines Semikolons eingefügt werden. Werteeinträge haben den Aufbau NAME=WERT. Es ist nicht zulässig, einen leeren Wert anzugeben. Der Hauptabschnitt in einer Meßgeräte-Initialisierungsdatei ist der Abschnitt METER.

3.3.1 Abschnitt METER

Dieser Abschnitt beschreibt die Schnittstelle des Meßgerätes. Er muß vorhanden sein, damit diese INI-Datei von MM verwendet wird. Folgende Einträge sind möglich:

NAME
Dieser Eintrag beschreibt das verwendete Meßgerät und kann ein beliebiger Name sein. Er wird in den Befehlsdateien verwendet, um genau dieses eine Meßgerät anzusprechen. Die Webseite zeigt diesen Namen ebenso an. Somit dient dieser Name als Identifizierungshilfe und sollte daher entsprechend aussagekräftig gewählt werden. Ohne diesen Eintrag wird diese INI-Datei nicht als Meßgeräte-Initialisierungsdatei erkannt und daher vom Programm nicht verwendet.

GROUP
Mit Hilfe dieses Eintrages kann das angegebenen Meßgerät einer Gruppe zugeordnet werden. Eine Instanz des DOS-Programms MM.EXE steuert eine komplette Gruppe. Damit ist sichergestellt, daß eine Befehlsdatei mit Anweisungen für mehrere Meßgeräte atomar ausgeführt wird, wenn sich alle Meßgeräte in einer Gruppe befinden.

PORT
Dieser Eintrag beschreibt, wie daß Meßgerät mit dem Meßrechner verbunden ist. Hier sind nur vordefinierte Abkürzungen erlaubt. Um einen seriellen Port anzusprechen muß hier COM1, COM2, COM3 oder COM4 eingetragen werden.
Wird hier IEEE eingetragen, wird der IEEE-488 Gerätetreiber verwendet, der durch weitere Einträge genauer definiert werden kann. Der Eintrag USB ist zur Zeit auch möglich, allerdings ist die Implementierung des USB-Ports nur ein Feedback der gesendeten Befehle. Für den Programmierer dient die USB-Port-Implementierung als Maske für eventuell zusätzliche Ports.

BAUDRATE (bei PORT=COMx)
Dieser Eintrag beschreibt die Übertragungsrate bei serieller Übertragung in Bits pro Sekunde. Hier sollte derselbe Wert eingetragen werden, der auch beim Meßgerät laut Dokumentation eingestellt ist. Falls dieser Eintrag fehlt, wird als Standard-Übertragungsrate 9600 Baud angenommen. Mögliche Werte sind hier 50 – 115200.

DATABITS (bei PORT=COMx)
Dieser Eintrag beschreibt die Anzahl der Datenbits bei serieller Übertragung. Er sollte ebenfalls äquivalent zur Einstellung am Meßgerät sein. Falls dieser Eintrag fehlt, wird als Standardwert 8 Datenbits angenommen. Mögliche Werte sind hier 5 – 8.

PARITY (bei PORT=COMx)
Dieser Eintrag beschreibt die Verwendung einer Parity-Prüfsumme bei serieller Übertragung. Er entspricht der Einstellung am Meßgerät. Mögliche Werte für Parity sind NONE, ODD, EVEN, MARK oder SPACE. Falls dieser Eintrag fehlt, wird als Parity NONE verwendet.

STOPBITS (bei PORT=COMx)
Dieser Eintrag beschreibt die Anzahl der Stoppbits bei serieller Übertragung. Mögliche Werte sind hier 1 – 2. Falls dieser Eintrag fehlt, wird genau ein Stoppbit angenommen.

INT (bei PORT=COMx)
Dieser Eintrag kann verwendet werden, falls die Interruptnummer des seriellen Ports nicht dem Standard entspricht. Die Standardzuordnung von Interrupts zu den seriellen Schnittstellen bei einem PC ist in Tabelle 1 dargestellt.

Serieller Port	Interrupt
COM1	4
COM2	3
COM3	4
COM4	3

Tabelle 1: Interruptzuordnung zu seriellen Schnittstellen

Wird eine zweite (ISA) Schnittstellenkarte in einem PC eingebaut, so wird diese auf COM3 und COM4 gesetzt, und als Interrupt stehen oft nur die Nummer 5 und 7 zur Verfügung, um Kollisionen mit COM1 und COM2 zu verhindern. Dann kann mit diesem Eintrag die Nummer richtig gesetzt werden.

BUFFERSIZE (bei PORT=COMx)
Mit diesem Eintrag kann die Größe des internen Datenpuffers für die serielle Übertragung verändert werden. Das kann dann notwendig werden, falls das Meßgerät die Daten schneller schickt, als MM die Daten in eine Datei auf dem Netzwerk schreiben kann. Bei Versuchen ist das mit einer Übertragungsrate von 115200 Baud passiert, als der Meßrechner kurzzeitig mit anderen Aufgaben belastet war und daher dem MM-Programm keine Rechenzeit zugeteilt wurde. Allerdings muß erwähnt werden, daß ein PC als Meßgerätesimulation verwendet wurde, der natürlich mit voller Leistung einfach Zufallszahlen schickte. Der Standardwert für die Größe des internen Datenpuffer ist 2048 Bytes. Die Maximalgröße ist auf 64kBytes beschränkt.
Die serielle Übertragung wird interruptgesteuert abgewickelt. Das bedeutet, daß zu dem Zeitpunkt, wenn ein Zeichen über die Schnittstelle kommt, das MM-Programm vermutlich mit etwas anderem beschäftigt ist, und die Interruptroutine das Zeichen in einen Puffer schreibt. Das MM-Programm prüft diesen Puffer und holt die Zeichen zur Weiterverarbeitung ab. Deshalb ist dieser Puffer notwendig, und man kann ersehen, daß, wenn die Zeichen zu schnell ankommen und MM keine Zeit zum Abholen hat, dieser Puffer überlaufen kann.

ADDR (bei PORT=IEEE)
Mit diesem Eintrag kann die Speicheradresse bei Übertragung über den IEEE-488-Bus eingestellt werden. Wird dieser Eintrag nicht gesetzt, wird als Standardadresse $C0000 angenommen. Das $ kommt von der hexadezimalen

Schreibweise in PASCAL. Natürlich kann hier auch der dezimale Wert eingetragen werden. ($C0000 entspricht 786432)

IO (bei PORT=IEEE)
Mit diesem Eintrag kann die Input/Output-Basisadresse bei Übertragung über den IEEE-488-Bus eingestellt werden. Fehlt dieser Eintrag, wird diese Adresse automatisch auf $300 gesetzt.

DMA (bei PORT=IEEE)
Mit diesem Eintrag kann der DMA-Kanal bei Übertragung über den IEEE-488-Bus eingestellt werden. Der Standardwert für den DMA-Kanal ist 3.

INT (bei PORT=IEEE)
Mit diesem Eintrag kann die Interruptnummer bei Übertragung über den IEEE-488-Bus eingestellt werden. Der Standardwert ist hier 5.

DEVICE (bei PORT=IEEE)
Mit diesem Eintrag kann die Gerätenummer für das Meßgerät auf dem IEEE-488-Bus eingestellt werden. Der Standardwert für die Gerätenummer ist 15.

DRIVER (bei PORT=IEEE)
Mit diesem Eintrag kann der Name des Gerätetreibers für den IEEE-488-Bus eingestellt werden. Der Standardwert ist die Zeichenfolge „IEEE“.

TIMEOUT
Mit diesem Eintrag kann die Zeitspanne in hundertstel Sekunden eingestellt werden, nach deren Überschreiten ohne Antwort vom Meßgerät eine Fehlermeldung ausgegeben wird. Der Standardwert entspricht 200, was eine Zeitspanne von 2 Sekunden bedeutet.

COMMANDDELAY
Mit diesem Eintrag kann eine Verzögerungszeit in hundertstel Sekunden nach jeder empfangener Antwort gesetzt werden, die mindestens gewartet wird, bis ein neuer Befehl an das Meßgerät geschickt wird. Das ist deshalb notwendig, da manche Meßgeräte bei Erhalt eines Befehls zum Setzen von internen Einstellungen den Befehl bereits mit einer gültigen Antwort quittieren, danach aber erst die Einstellungen vornehmen und somit auf den nächsten folgenden Befehl nicht reagieren können, falls dieser zu schnell nachkommt. Der Standardwert ist hier gleich 0. Es wird also mit maximaler Geschwindigkeit kommuniziert.

TIMELIMIT
Mit diesem Eintrag kann eine maximale Ausführungszeit für eine Befehlsdatei definiert werden. Eine laufende Befehlsdatei wird nach Überschreiten der eingestellten Zeit in hundertstel Sekunden abgebrochen. Somit wird ein Blockieren des Meßsystems verhindert. Der Standardwert für diesen Eintrag ist gleich 0, was

bedeutet, daß maximale Ausführungszeit zur Verfügung steht. Dieser Eintrag kann im Einsatz als Laborübung sinnvoll sein.

RETRIES
Mit diesem Eintrag kann die Anzahl der Wiederholungen gesetzt werden. Ein Befehl wird wiederholt, falls eine ungültige Antwort zurückkommt. Das kann an Störungen am Übertragungsweg liegen, oder die Geschwindigkeit der Übertragung ist zu hoch gesetzt. Erhält man immer wieder falsche Antworten, kann man mit dem Eintrag BAUDRATE die Geschwindigkeit etwas drosseln. Der Standardwert für die Versuche beträgt 3. Das bedeutet, daß nach dreimaliger falscher Antwort der Vorgang mit einem Fehler abgebrochen wird.

SOT (**S**tart **O**f **T**ransmit)
Diese Zeichenkette oder ein einzelnes Zeichen beschreibt die führenden Zeichen, die jeder Übertragung an das Meßgerät vorangehen müssen, damit dieses die folgende Befehlssequenz akzeptiert. Der Standardwert ist die leere Zeichenkette. Maximal 4 Zeichen sind hier möglich.

EOT (**E**nd **O**f **T**ransmit)
Diese Zeichenkette oder ein einzelnes Zeichen beschreibt die abschließenden Zeichen, die jeder Übertragung an das Meßgerät folgen müssen, damit dieses das Ende des Befehls interpretieren kann. Der Standardwert ist die leere Zeichenkette. Es sollte aber unbedingt eine vom Meßgerät vorgegebene Markierung definiert werden. Oft wird hier NL oder LF verwendet, die einen Zeilenwechsel anzeigen. Solche Sonderzeichen können durch deren ASCII-Wert mit einem beliebigen Editor leicht eingegeben werden. Das Format dieser Zeichendarstellung ist auf der nächsten Seite ersichtlich.

SOR (**S**tart **O**f **R**eceive)
Diese Zeichenkette oder ein einzelnes Zeichen beschreibt die einleitenden Zeichen, die jeder empfangenen Antwort vom Meßgerät vorangehen. Der Standardwert ist die leere Zeichenkette.

EOR (**E**nd **O**f **R**eceive)
Diese Zeichenkette oder ein einzelnes Zeichen beschreibt die abschließenden Zeichen, die jede empfangene Anwort eindeutig terminieren. Der Standardwert ist die leere Zeichenfolge. Hier sollte aber unbedingt mindestens ein Zeichen definiert sein, damit das Ende erkannt werden kann. Ohne Angabe dieses Eintrages wird bis zum Timeout gewartet und dann erst das Ergebnis akzeptiert.

BUSY
Diese Zeichenkette oder ein einzelnes Zeichen beschreibt das Erkennungsmerkmal, das anzeigt, daß das Meßgerät gerade keine Befehle empfangen kann. Es wird damit solange gewartet, bis das Meßgerät die Zeichenfolge READY sendet. Der Standardwert ist wieder die leere Zeichenfolge, was bedeutet, BUSY/READY wird nicht unterstützt.

READY
Diese Zeichenkette oder ein einzelnes Zeichen beschreibt das Erkennungsmerkmal, das anzeigt, daß das Meßgerät wieder Befehle verarbeiten kann. Diese Zeichen werden nur als READY interpretiert, falls vorher eine BUSY Zeichenfolge gesendet wurde. Der Standardwert ist die leere Zeichenfolge, was bedeutet, BUSY/READY wird nicht unterstützt.

Ein Beispiel für einen METER Abschnitt aus der Initialisierungsdatei für das Bell Gaußmeter 9500:

```
[METER]
NAME=Gaussmeter          ; Name für Meßgerät
GROUP=Messung1           ; Gruppenname
PORT=COM1                ; erster serieller Port
BAUDRATE=9600            ; 9600 Bit/s werden übertragen
PARITY=N                 ; N(one) keine Parityüberprüfung
DATABITS=8               ; es werden 8 Datenbits verwendet
STOPBITS=1               ; es folgt 1 Stoppbit
INT=4                    ; Unterbrechungsanforderung auf 4
TIMEOUT=200              ; 2 Sekunden Wartezeit
RETRIES=3                ; 3 Versuche
COMMANDDELAY=100         ; 1 Sekunde zwischen Befehlen
TIMELIMIT=0              ; maximale Ausführungszeit
EOR=#13                  ; CR ist abschließendes Zeichen
SOT=$1B                  ; ESC beginnt Befehlssequenz
EOT=#13                  ; CR beendet Befehlssequenz
BUSY=#13                 ; Meßgerät ist beschäftigt
READY=#11                ; Meßgerät ist wieder verfügbar
```

Zeichen können in den Initialisierungsdateien auf unterschiedliche Arten dargestellt werden. Zahlen können durch dezimale Zahlen 1, 2, 3, ... und durch hexadezimale Angaben $1, $2, $3, ... beschrieben werden.
Zeichenketten können, falls Verwechslungen möglich sind durch umschließende einfache Hochkommas dargestellt werden ('TEXT').

Spezielle nicht druckbare Zeichen können durch ihre ASCII Darstellung leicht eingegeben werden. Folgende Formate sind möglich:

#xx
Diese Angabe beschreibt genau ein Zeichen mit dem ASCII-Wert xx. Genau 2 Stellen werden erwartet.

@xxx
Diese Angabe beschreibt genau ein Zeichen mit dem ASCII-Wert xxx. Es werden 3 Stellen erwartet. Mögliche Werte sind 000 – 255, was die Angabe der kompletten Menge aller ASCII-Zeichen erlaubt.

$xx
Diese Angabe beschreibt genau eine hexadezimale Zahl. Es können beliebig viele Stellen angegeben werden. Im obigen Beispiel wird damit die Zahl 27 beschrieben, die dem Eintrag SOT zugeordnet wird, der daraus die Zeichenkette #27 macht. Das ist nur möglich, wenn nur ein Zeichen verwendet wird.
EOT=#13#10 definiert eine Zeichenkette mit 2 Zeichen (CR und LF). Hier ist die Verwendung von $xx nicht zulässig.

3.3.2 Abschnitt PARAMETERS

In diesem Abschnitt können Variablen definiert werden, die zum Speichern von Informationen verwendet werden. Die Definition erfolgt wieder als Wertezuordnung, wobei der angegebene Wert als Initialwert dieser Variablen zu verstehen ist. Da dieser Wert beim Starten des Programms MM.EXE einmalig aus der Initialisierungsdatei gelesen wird, ist der Inhalt dieses Parameters nach der ersten Befehlsdatei, die diesen Parameter ändert, gleich dem neuen Wert. Deshalb sollten Befehlsdateien die Variablen immer vor deren Verwendung initialisieren, deren Inhalt sie verwenden wollen. Ein Beispiel für eine solche Verwendung wäre etwa eine Zählvariable innerhalb einer Schleife.

Eine Initialisierungsdatei enthält beispielsweise folgende Definition einer Variablen:

```
[PARAMETERS]
I=0
```

Eine Befehlsdatei, die auf diese Initialisierungsdatei aufbaut, kann eine Schleife in dieser Art realisieren:

```
SET I 0
:AGAIN
BEFEHL1
BEFEHL2
...
SET I I+1
IF I < 10 GOTO AGAIN
```

Hier werden die Befehle in einer Schleife 10 mal ausgeführt. Die Laufvariable I wird hier auf den Wert 0 initialisiert, da nicht sichergestellt ist, daß I zu diesem Zeitpunkt den Wert aus der Initialisierungsdatei entspricht. Nach dem Durchlauf dieser Befehle enthält I den Wert 10.

Ein Beispiel wieder an Hand des Gaußmeters:

```
[PARAMETERS]
RANGE=0
SIGN=0
```

```
READING=0
```

Hier werden 3 Variablen verwendet, die in der Befehlsdatei zur Speicherung der Rückgabewerte des Meßgerätes verwendet werden.
Als Spezialparameter sind T und PI definiert. Diese beiden Namen können nicht im Abschnitt PARAMETERS neu definiert werden.
Der Parameter T steht für die Zeit in hundertstel Sekunden relativ zum Start der Befehlsdatei. Diese Zeit kann durch Befehle beeinflußt werden. Siehe dazu den Abschnitt „Spezielle Befehle“.
Der Parameter PI liefert die mathematische Konstante PI und kann somit in Formeln verwendet werden. Als Zahlenwert wird 3.1415926536 verwendet.

3.3.3 Abschnitt CONSTANTS

Dieser Abschnitt beschreibt Variablen, die nicht zur Speicherung von Daten herangezogen werden können, sondern Initialisierungswerte enthalten, die für immer den Wert behalten und als Parameter für Befehle verwendet werden können. Ein Anwendungsbeispiel wäre die Übersetzung von komplizierten Parametern in leicht merkbare Abkürzungen.

```
[CONSTANTS]
ON=1
OFF=2
```

Hier muß sich der Anwender nicht mehr merken, welche Zahl für das Aktivieren einer Funktion notwendig ist. Er verwendet einfach den Parameter ON.

```
[CONSTANTS]
GAUSSAC=1
GAUSSDC=2
TESLAAC=3
TESLADC=4
```

Hier wird für den Anwender eine Befehlsdatei leichter lesbar, wenn er anstelle einer Zahl als Parameter die Eigenschaft dieser Zahl sieht und damit sofort erkennen kann, daß hier der Modus des Meßgerätes eingestellt wird.

```
[CONSTANTS]
CHANNEL1='CHANNEL1'
CHANNEL2='CHANNEL2'
```

Bei manchen Befehlen wird als Parameter eine Textfolge erwartet. Diese müßte der Anwender mit Hochkommas in der Befehlsdatei eingeben. Mit diesen Einträgen, kann der Anwender den Text ohne Hochkommas eingeben und die möglichen Werte werden auch auf der Webseite angeboten. Mehr dazu im Abschnitt „Web-basierte Oberfläche‘.

Der Abschnitt CONSTANTS zeigt lediglich die Eigenschaft an, daß diese Variablen nicht innerhalb einer Befehlsdatei auf neue Werte geändert werden können. In diesem Abschnitt können auch Formeln definiert werden. Diese liefern immer eine Zahl als Ergebnis, die aber nicht konstant sein muß. Der Ausdruck „konstant" bedeutet hier, daß die rechte Seite der Zuweisung nicht verändert werden kann. Die Auswertung dieser rechten Seite kann aber variable Werte liefern. Eine Formel könnte theoretisch auch im Abschnitt PARAMETERS definiert werden, dann kann aber eine Befehlsdatei diese Variable auf eine Zahl setzen, und die Formel ist verschwunden.

Eine Formel ist ein Ausdruck, der Zahlen und Parameter enthalten kann, die durch Rechenzeichen verbunden werden. Folgende Rechenoperationen werden unterstützt: Addition (+), Subtraktion (-), Multiplikation (*), Division (/), Ganzzahldivision (\) und Modulooperation (%). Die Reihenfolge der zu durchführenden Operationen kann durch Klammersetzung geändert werden. Tabelle 2 zeigt alle möglichen verwendbaren Funktionen.

Funktionsname	**Vorgang**
ABS(x)	Liefert den absoluten Wert von x zurück.
INT(x)	Liefert den ganzzahligen Wert von x zurück.
FRAC(x)	Liefert die Kommastellen von x zurück.
TRUNC(x)	Liefert einen Integer-Wert von x als 16-bit-Zahl zurück.
ROUND(x)	Rundet x und liefert dann den Integer-Wert als 16-bit-Zahl zurück.
COS(x)	Liefert den Kosinus von x zurück.
SIN(x)	Liefert den Sinus von x zurück.
ARCTAN(x)	Liefert den ArcusTangens von x zurück.
SQR(x)	Liefert das Quadrat von x zurück.
SQRT(x)	Liefert die Wurzel von x zurück.
EXP(x)	Liefert die Exponentialfunktion von x zurück.
LN(x)	Liefert den Logarithmus Naturalis von x zurück.
INC(x)	Erhöht x um 1.
DEC(x)	Erniedrigt x um 1.

Tabelle 2: Funktionen

Die Großschreibung bei den Funktionsnamen ist zu beachten. Die Formelauswertung arbeitet immer für alle Berechnungen mit dem PASCAL-Datentyp REAL. Die Argumente x sind Realzahlen für alle Funktionen. Wenn nicht anders angegeben liefert die Funktion wieder eine Realzahl.

Ein Beispiel:

```
[CONSTANTS]
```

```
V/DIV=32 * YINC
OFFSET=(128 - YREF) * YINC + YORG
S/DIV=POINTS * XINC / 10
DELAY=(POINTS/2 - XREF) * XINC + XORG
```

Hier werden 4 Formeln definiert die aus einigen Parameters Werte ausrechnen. Bei jedem Zugriff auf die definierten Formelnamen wird die Formel neu ausgewertet; damit ist sichergestellt, daß immer die aktuellsten Werte der Parameter zur Berechnung herangezogen werden.

3.3.4 Abschnitte STREAM

Ein Stream beschreibt eine Datenfolge zur Speicherung von Informationen in einer Datei. Eine Initialisierungsdatei kann mehrere Streams enthalten. Er hat denselben Aufbau wie der ERROR Eintrag in der Datei mm.cfg. Folgende Einträge sind in diesem Abschnitt möglich:

FILE
Dieser Eintrag enthält den Namen der Ausgabedatei. Pfadangaben werden relativ zum Arbeitsverzeichnis behandelt. Bei der Verwendung als Server ist es sinnvoll, diesen Eintrag wegzulassen, damit als Dateiname der Name der Befehlsdatei verwendet wird. Als Dateinamenserweiterung wird dann .OUT angenommen. Heißt eine Befehlsdatei z.B. C8.CMD, dann erfolgt die Ausgabe der Daten in eine Datei C8.OUT, die sich im Arbeitsverzeichnis des Programms MM befindet. (Dort wurde auch C8.CMD gefunden.)

HEAD
Dieser Eintrag enthält eine Zeichenkette, die beim Öffnen der Ausgabedatei in diese geschrieben werden soll. Enthält diese Zeichenkette Fragezeichen (?), dann werden beim Öffnen dieses Streams genau so viele Parameter erwartet wie Fragezeichen. Mehrere Zeilen können durch das Einfügen von #13#10 realisiert werden. Der zugehörige Befehl in einer Befehlsdatei lautet OUT STREAM OPEN [Parameter]. Mehr dazu im Abschnitt „Befehlsdateien“.

LINE
Dieser Eintrag enthält eine Maske für eine Zeile in der Ausgabedatei. Es ist sinnvoll, hier Fragezeichen zu verwenden. Für jedes Fragezeichen wird ein Parameter erwartet. Der zugehörige Befehl in einer Befehlsdatei lautet OUT STREAM LINE [Parameter].

TAIL
Dieser Eintrag ist ähnlich zu HEAD. Nur daß hier die Zeichenfolge beim Schließen der Datei in diese geschrieben wird. Der zugehörige Befehl in einer Befehlsdatei lautet OUT STREAM CLOSE [Parameter].

PARAMS
Dieser Eintrag definiert die zu verwendenden Parameter, wenn bei der Ausgabe keine angegeben wurden. (Befehl: OUT STREAM LINE). Das ist ebenso der Fall, wenn das Meßgerät Zahlenfolgen liefert. Das Oszilloskop z.B. liefert Zahlen durch Beistriche getrennt. Diese Zahlen werden durch die speziellen Parameter #COUNT und #VALUE beschrieben. Für jede ankommende Zahl wird OUT STREAM LINE aufgerufen, und #COUNT beinhaltet die Nummer der aktuellen Zahl und #VALUE den Wert der aktuellen Zahl. Diese 2 Werte können in diesem Eintrag verwendet werden. Der Aufbau dieses Eintrages entspricht genau dem Format, wie er auch als Parameter an OUT STREAM LINE angehängt werden kann.

Ein Stream muß einen eindeutigen Namen haben; dieser wird als Abschnittsnamen mit einem Doppelpunkt an das Schlüsselwort STREAM angehängt.

Ein Beispiel:

```
[STREAM:DATA]
HEAD=';Measurement Results#13#10;x,y,S,V'
LINE='?,?,?,?'
PARAMS=#COUNT #VALUE TIME VOLTAGE
```

Hier wird eine Datenfolge mit Namen DATA definiert. Dieser Name wird für den Befehl OUT verwendet. (OUT DATA OPEN)
Der Eintrag FILE wurde hier nicht angegeben, somit wird als Name der Name der zugehörigen Befehlsdatei mit der Erweiterung .OUT verwendet.
Der Eintrag HEAD beschreibt hier die Überschrift für die Meßergebnisse. In der zweiten Kopfzeile werden die Bezeichnungen für die 4 zu speichernden Werte angegeben. Man beachte das Semikolon, womit die Kopfzeilen als Kommentar in die Ausgabedatei geschrieben werden. Diese Darstellung ist für die Webseite wichtig.
Der Eintrag LINE enthält 4 Fragezeichen; es werden daher 4 Parameter erwartet. Die Ausgabe erfolgt durch 4 Zahlen pro Zeile, die durch Beistriche getrennt werden.
Der Eintrag PARAMS beschreibt, welche 4 Zahlen in jeder Zeile ausgegeben werden sollen. Die erste Zahl soll #COUNT sein, also die laufende Nummer der erhaltenen Zahlen vom Meßgerät. Die zweite Zahl ist mit #VALUE definiert. Das entspricht dem Wert der aktuellen vom Meßgerät zurückgelieferten Zahl. Die beiden anderen Zahlen sind Formeln oder Parameter, die im Abschnitt CONSTANTS oder PARAMETERS definiert worden sind. Ein Beispiel für deren Definition:

```
[CONSTANTS]
VOLTAGE=(#VALUE - YREF) * YINC + YORG
TIME=(#COUNT - XORG) * XINC + XORG
```

Hier wird VOLTAGE als Formel definiert, die den aktuellen Zahlenwert (#VALUE) verwendet und mit anderen zurückgelieferten Parametern eine neue Zahl berechnet. In diesem Fall ist es der aktuelle Spannungswert vom Oszilloskop. Der zweite Parameter TIME verwendet die aktuelle Laufnummer der vom Meßgerät gelieferten Zahlen (#COUNT) und berechnet den Zeitpunkt für den Spannungswert.

Formeln und Konstanten können auch im Abschnitt PARAMETERS definiert werden. Dann ist aber nicht sichergestellt, daß sie immer als Formel berechnet werden. Ein Benutzer kann dann mit dem SET Befehl diesen Parameter neu definieren und somit die Formelanweisung löschen. Daher ist es sinnvoll, Formeln und Abkürzungen nur im Abschnitt CONSTANTS unterzubringen.

3.3.5 Abschnitt COMMANDS

Dieser Abschnitt enthält Zuordnungen von Metabefehlen zu wirklichen Meßgerätebefehlen. Wird in der Befehlsdatei ein Metabefehl angegeben, so wird in diesem Abschnitt nach einer Übersetzung gesucht und diese zum Meßgerät geschickt. Antworten vom Meßgerät können durch einen Eintrag, der dem Namen des Metabefehls und einem angehängtem _RESPONSE entspricht, dargestellt werden. Fehlt dieser RESPONSE-Eintrag, so wird angenommen, daß das Meßgerät keine Antwort liefert und den gesendeten Befehl stumm angenommen hat.
Befehle an das Meßgerät enthalten meistens Parameter. Diese können mit den speziellen Zeichen ? und * angegeben werden. Ein Fragezeichen (?) in einem Befehl bedeutet, daß hier exakt ein Zeichen als Parameter eingesetzt werden soll. Werden z.B. 3 Zeichen erwartet und die Zahl ist 21, dann wird durch die Angabe von ??? die Zahl 021 in die Befehlsfolge eingesetzt. Ein Stern (*) in einem Befehl bedeutet, daß hier der Parameter ohne Beachtung seiner Länge eingesetzt werden soll. Treffen 2 Parameter direkt aufeinander, so kann als Trennzeichen | (ALTGR+124) verwendet werden, um eindeutig die Werte zuordnen zu können.
Einträge im Abschnitt COMMANDS sind immer automatisch Zeichenketten; somit erübrigt sich die Verwendung von umschlossenen Hochkommas. Spezielle Sonderzeichen können mit Hilfe der Zeichen # und @ beschrieben werden. Werden diese Zeichen selbst im Befehl benötigt, kann dies durch Voranstellen von \ erfolgen. Ebenso können die hier verwendeten Steuerzeichen ? und * als Zeichen dargestellt werden (\? und *).
Für einen Befehl müssen in der Befehlsdatei genau so viele Parameter, Konstanten oder Zahlen angegeben werden, wie in diesem Abschnitt durch ? und * definiert sind. Enthält der Abschnitt auch einen zugehörigen RESPONSE Eintrag, so muß in der Befehlsdatei auch noch die jeweilige Zahl von Parametern zum Speichern der Antwort mit angegeben werden.
Für RESPONSE Einträge gelten noch weitere Sonderzeichen. Das Zeichen ! ignoriert einen folgenden Parameter. Das Meßgerät kann irgendeinen Wert schicken; er hat keinen Einfluß auf das Ergebnis. Das Zeichen % ermöglicht es, gesendete Parameter, welche das Meßgerät zurückschickt zu überprüfen. Der Aufbau ist dermaßen, daß %? bedeutet, den ersten als ein Zeichen gesendeten Parameter zu überprüfen. Die Anzahl der % Zeichen bestimmt die Nummer des Parameters. %%? überprüft an dieser Stelle den zweiten gesendeten Parameter. Somit kann z.B. der erste Parameter ignoriert werden, der zweite jedoch überprüft (!?%%?). Jeder Parameter ist hier durch das ? wieder genau eine Stelle lang.
Ein weiteres besonders Zeichen ist die eckige Klammer [. Sie leitet eine Datenfolge ein, die erst mit Angabe von] wieder beendet wird. Folgt eine Zahl oder ein Parameter der Klammer, so gibt dieser die Anzahl der zu erwartenden Zahlen in der

Folge an. Danach sollte das Format beschrieben werden. Wieder steht ein * für beliebige Zeichen, ein oder mehrere ? definieren die exakte Anzahl von Zeichen für jede zu erwartende Zahl, danach folgen ein oder mehrere Trennzeichen, die die einzelnen Zahlen definiert voneinander trennen. Bei Angabe von * für die Anzahl ist die Angabe mindestens eines Trennzeichens erforderlich. Bei Verwendung von ? zur genauen Angabe der Zeichen pro Zahl kann auch ohne Trennzeichen übertragen werden, es wird aber dann vermutlich zu nicht lösbaren Synchronisationsproblemen kommen, und Fehler können nicht erkannt werden. Die Datenfolge wird durch] abgeschlossen, danach können noch andere Abschlußzeichen folgen.

Beispiele sollen das Gesagte verdeutlichen.

```
[COMMANDS]
CLS=\*CLS
```

Hier wird einfach die Zeichenkette „*CLS“ an das Meßgerät geschickt. * entspricht hier keinem Parameterplatzhalter, da es mit \ eingeleitet wird. Ein RESPONSE Eintrag fehlt, deshalb wird nach Senden von „*CLS“ mit dem nächsten Befehl fortgesetzt, ohne auf eine Antwort vom Meßgerät zu warten.

```
[COMMANDS]
GETIDN=\*IDN\?
GETIDN_RESPONSE=*
```

Hier wird die Zeichenfolge ‚*IDN?‘ an das Meßgerät gesendet. Ein RESPONSE Eintrag existiert, deshalb wird eine Antwort erwartet. Diese ist als * definiert, was bedeutet, daß die komplette Anwort in einer Variablen gespeichert wird. Daher muß der Aufruf in der Befehlsdatei z.B. GETIDN IDENTIFY lauten, wobei IDENTIFY im Abschnitt PARAMETERS definiert worden sein muß. Dieser Parameter enthält nach erfolgreichem Empfang die Antwort des Meßgerätes und kann weiter verwendet werden.

```
[COMMANDS]
MODE=MO1?|?
MODE_RESPONSE=MO1%?|%%?
```

Der Befehl MODE erwartet 2 Parameter beim Aufruf. Jeder Parameter sollte genau einem Zeichen entsprechen. Da die 2 Parameter genau aufeinanderfolgen, ist das Trennzeichen | notwendig. Als Antwort liefert das Meßgerät den gesendeten Befehl zurück. Die 2 gesendeten Parameter werden hier auch überprüft. Beim RESPONSE Eintrag könnte man das Trennzeichen | auch weglassen, da hier die Angabe eindeutig ist. Er erhöht aber die Lesbarkeit. Will man die Parameter nicht überprüfen und begnügt sich damit, daß MO1.. bereits als gültige Antwort geliefert wird, sind folgende RESPONSE Einträge gleichbedeutend:

```
a) MODE_RESPONDE=MO1!?!?
b) MODE_RESPONSE=MO1!??
c) MODE_RESPONSE=MO1!*
```

Bei a) werden zwei zurückgelieferte aus einem Zeichen bestehende Parameter ignoriert. Bei b) wird ein aus zwei Zeichen bestehender Parameter ignoriert. Bei c) werden alle nachfolgenden Zeichen ignoriert. Bei Verwendung von c) könnte das Meßgerät auch mehrere Zeichen zurück liefern und die Antwort würde immer noch akzeptiert werden.

```
[COMMANDS]
WAVGETDATA=:WAVEFORM:DATA\?
WAVGETDATA_RESPONSE=##8!????????[*,]
```

Hier wird der Befehl zum Senden der Daten an das Oszilloskop gerichtet. Das Meßgerät antwortet danach mit einer Datenfolge die mit ‚#8' eingeleitet wird, worauf 8 Zeichen folgen, die hier ignoriert werden. Dann wird mit der Übertragung von Zahlen begonnen, die durch Beistriche getrennt werden. Als STREAM wird ein Parameter erwartet, der in der Befehlsdatei angegeben werden muß. Ein gültiger Aufruf lautet also ‚WAVGETDATA DATA', falls DATA als [STREAM:DATA] Abschnitt in der INI-Datei aufscheint. Für jedes gefundene Zeichen wird OUT DATA LINE aufgerufen. Die Parameter werden aus dem PARAMS Eintrag abgeleitet und gemäß dem STREAM Abschnitt in eine Datei eingetragen. Durch die Angabe von * sind beliebig viele Zeichen in einer Zahl möglich. Wenn das Meßgerät immer 3 Zeichen schickt, ist es besser, ??? anzugeben. Damit ist es möglich, Übertragungsfehler zu erkennen. Geht ein Zeichen verloren, stimmt die Anzahl mit den 3 Fragezeichen nicht mehr überein, und die Übertragung wird mit einem Fehler abgebrochen. Bei Verwendung von * können solche Fehler nicht erkannt werden, und es wird dann ein falscher Zahlenwert in der Datei gespeichert.

```
[COMMANDS]
WAVPOINTS=:WAVEFORM:POINTS *
WAVGETDATA=:WAVEFORM:DATA\?
WAVGETDATA_RESPONSE=##8!????????[RESULT*,]
```

Hier wurde der obige Befehl um die Überprüfung der Ergebnisanzahl erweitert. Der Parameter RESULT wird vor Aufruf von WAVGETDATA auf die erwartete Anzahl der Werte gesetzt. Damit wird beim Empfang der Zeichen überprüft, ob genau die richtige Anzahl angekommen ist. Ein Befehlsablauf für dieses Beispiel könnte so aussehen:

```
...
SET RESULT 400
WAVPOINTS RESULT
WAVGETDATA DATA
...
```

Der Parameter RESULT wird auf den Wert 400 gesetzt, der Befehl zum Einstellen der Anzahl der Punkte wird aufgerufen. Danach werden die Werte abgeholt, und es wird überprüft, ob genau 400 Werte geschickt wurden.

3.3.6 Spezielle Abschnitte

Meßgeräte liefern Ergebnisse nicht immer als unmittelbar verwendbaren Zahlenwert zurück. Das Gaußmeter liefert z.B. eine Zahl für den Wertebereich, mit dem das Ergebnis noch multipliziert werden muß, um den wahren Wert in Gauß oder Tesla zu erhalten. Dafür sind spezielle Abschnitte vorgesehen, deren Namen dem eines Parameters entsprechen. In diesem Abschnitt können nun Zuordnungen von erhaltenen Zeichen zu Zahlen vorgenommen werden. Ein spezieller Eintrag namens DEFAULT sorgt dafür, daß die Ergebnismenge der verfügbaren Zahlen beschränkt und übersichtlich bleibt.

Ein Beispiel:

```
[RANGE]
DEFAULT=1
1=1
2=10
3=100
4=1000
5=10000
6=100000
```

Hier wird der Parameter RANGE jedesmal, wenn auf ihn schreibend zugegriffen wird, umgewandelt. Das Meßgerät liefert hier Werte zwischen 1 und 6 als Bereich zurück, die einer 10er Potenz entsprechen. Falls ein anderer Wert geliefert wird, wird RANGE auf 1 gesetzt.
Das funktioniert auch mit Zeichen:

```
[SIGN]
DEFAULT=1
-=-1
```

Hier wird ein Zeichen als Vorzeichen zurückgeliefert. Alle Zeichen sollen dem Faktor 1 entsprechen. Kommt ein Minuszeichen zurück, so entspricht dies dem Faktor –1.
Der endgültige Ergebniswert kann mit einer Formel berechnet werden, die im Abschnitt CONSTANTS definiert werden sollte:

```
[CONSTANTS]
RESULT=SIGN * RANGE * READING
```

Hier wird SIGN als Vorzeichen auf 1 oder –1 gesetzt. RANGE beinhaltet den Wertebereich und READING ist der Zahlenwert des Ergebnisses. Alle 3 Werte miteinander multipliziert ergeben das Ergebnis in der richtigen Größeneinheit.

3.4 Befehlsdateien .CMD

Eine Befehlsdatei hat den gleichen Aufbau wie eine Initialisierungsdatei. Sie kann mehrere Abschnitte enthalten. Ein Abschnitt wird immer genau einem Meßgerät zugeordnet. Der Name des Abschnittes muß einem NAME-Eintrag in einer der vorhandenen Initialisierungsdateien entsprechen. Damit erfolgt die Zuordnung von Befehlsdateien zu den Initialisierungsdateien für die Meßgeräte. In derselben Zeile mit dem Abschnittsnamen kann zusätzlich als Kommentar die Eigenschaft dieser Befehle beschrieben werden. Diese Beschreibung wird von der Webseite verwendet. Jeder Befehl sollte wegen guter Übersichtlichkeit in einer eigenen Zeile stehen. Die Einträge in diesen Abschnitten unterscheiden sich von normalen Initialisierungsdateien dadurch, daß es hier keine Zuordnungen gibt. Befehlsdateien sind also Initialisierungsdateien ohne das Zeichen „=“ und ohne Werte. Deshalb ist das Zeichen „=“ auch in Befehlsnamen verboten. Die Befehlsdateien sind ebenso wie die Initialisierungsdateien von der Groß/Kleinschreibung abhängig. Kommentare können an beliebiger Stelle mit Semikolons eingefügt werden.

3.4.1 Spezielle Befehle

3.4.1.1 Befehl GOTO

Der Befehl GOTO bewirkt einen Sprung zur angegebenen Marke, und die Befehlsausführung wird mit dem darauffolgenden Befehl fortgesetzt. Marken sind mit einem einleitenden Doppelpunkt gekennzeichnet. Sprünge sind sowohl aufwärts als auch abwärts möglich, aber nur innerhalb eines Abschnittes. Es kann also nicht von einem Abschnitt in einen anderen Abschnitt verzweigt werden.

Die Syntax lautet:

```
GOTO marke
:marke
```

Beispiel:

```
[Gaussmeter]
BEFEHL1
GOTO WEITER
BEFEHL2
:WEITER
BEFEHL3
```

In diesem Beispiel wird nach Ausführung des Befehls BEFEHL1 der zweite Befehl übersprungen und gleich der Befehl BEFEHL3 ausgeführt. Sinn macht das natürlich erst, wenn Bedingungen geprüft werden, ob einige Befehle übersprungen werden sollen. Das obige Beispiel kann eingesetzt werden, um kurzzeitig mehrere Zeilen in der Befehlsdatei auszublenden. Man erspart sich somit das Schreiben von Semikolons am Anfang jedes nicht auszuführenden Befehls.

3.4.1.2 Befehl DUMP

Der Befehl DUMP bewirkt eine Ausgabe des aktuellen Wertes für einen angegebenen Parameter. Die Ausgabe erfolgt auf dem Bildschirm, auf dem das Programm MM läuft. Deshalb ist dieser Befehl nur bei Verwendung von MM als Befehlszeilenprogramm sinnvoll. Will man Werte in Dateien zur Weiterbearbeitung ausgeben, kann man den Befehl OUT verwenden. Dieser Befehl erleichtert die Fehlersuche in einer Befehlsdatei, falls am Ende falsche Ergebnisse geliefert werden. Somit kann man überprüfen, wann die Werte nicht mehr richtig sind. Der angegebene Parameter muß im Abschnitt PARAMETERS in der zugehörigen Initialisierungsdatei definiert worden sein.

Die Syntax lautet:

```
DUMP parameter
```

Beispiel:

```
[Gaussmeter]
DUMP RESULT
```

Hier wird der Ergebniswert der Messung in der Form RESULT=wert auf dem Bildschirm angezeigt.

3.4.1.3 Befehl IF

Der Befehl IF prüft eine Bedingung und führt danach alle folgenden Befehle in der selben Zeile aus oder nicht. Die Bedingungen können Zahlen, Parameter oder Zeichenketten enthalten. Zeichenketten und Zahlen können aber nicht gemischt verwendet werden. Die Tabelle 3 soll die möglichen Abfragebedingungen verdeutlichen.

Operator	**Bedeutung**
= oder ==	Hier wird auf Gleichheit der beiden Operanden überprüft.
<> oder !=	Hier wird auf Ungleichheit der beiden Operanden überprüft.
<	Hier wird überprüft, ob der linke Operand kleiner als der rechte ist.
>	Hier wird überprüft, ob der linke Operand größer als der rechte ist.
<=	Hier wird überprüft, ob der linke Operand kleiner oder gleich dem rechten ist.
>=	Hier wird überprüft, ob der linke Operand größer oder gleich dem rechten ist.

Tabelle 3: Vergleichsoperatoren für den IF-Befehl

Die Syntax lautet:

```
IF x OPERATOR y BEFEHL1 [BEFEHL2 ...]
```

Für x und y kann hier ein Parameter aus dem PARAMETERS- oder CONSTANTS-Abschnitt der zugehörigen Initialisierungsdatei, eine Zahl oder eine Zeichenkette eingesetzt werden. Trifft die Bedingung zu, werden die Befehle, die auf y folgen, ausgeführt.

Beispiel:

```
[Gaussmeter]
BEFEHL1
IF RESULT == 0 GOTO WEITER
BEFEHL2
:WEITER
BEFEHL3
```

Hier wird nach Ausführung des Befehls BEFEHL1 überprüft, ob der Parameter RESULT dem Wert 0 entspricht. Falls dies zutrifft, wird der Befehl BEFEHL2 übersprungen und gleich mit dem Befehl BEFEHL3 fortgesetzt. Falls aber RESULT einen Wert ungleich 0 angenommen hat, werden die Befehle in der Zeile mit dem IF Befehl ignoriert, und es wird in der nächsten Zeile mit der Ausführung fortgesetzt. Es kommt also BEFEHL2 als nächster an die Reihe. Diese Kopplung von IF und GOTO entspricht also einer Konstruktion gleich wie IF THEN ELSE, wie sie aus Programmiersprachen bekannt ist.

Beispiel:

```
[Gaussmeter]
BEFEHL1
IF RESULT == 0 IF SIGN != 1 GOTO WEITER
BEFEHL2
:WEITER
BEFEHL3
```

In diesem Beispiel werden zwei IF Anweisungen verbunden und stellen damit eine UND Verknüpfung dar. RESULT muß 0 und SIGN ungleich 1 sein, damit der Befehl BEFEHL2 übersprungen wird.

Beispiel:

```
[Gaussmeter]
BEFEHL1
IF RESULT == 0 GOTO WEITER
IF SIGN != 1 GOTO WEITER
BEFEHL2
:WEITER
BEFEHL3
```

In diesem Beispiel wird eine ODER Verknüpfung zweier Bedingungen dargestellt. Die Abfragen müssen in getrennten Zeilen erfolgen. Wenn mindestens eine Abfrage wahr liefert, wird der Befehl BEFEHL2 übersprungen.

3.4.1.4 Befehl CALL

Der Befehl CALL dient zum Aufruf von weiteren Befehlsdateien. Ein Dateiname kann mit absoluter oder relativer Pfadangaben als Parameter angegeben werden. Relative Pfadangaben starten vom selben Verzeichnis wie die rufende Befehlsdatei. Die angegebene Datei muß mindestens einen Abschnitt mit dem selben Meßgerätenamen enthalten, wie die rufende, ansonsten wird die Befehlsabarbeitung abgebrochen. Dieser Befehl erlaubt die Modulisierung von einzelnen Messungen. So können z.B. alle Initialisierungsbefehle, die bei Beginn der Messung notwendig sind, in einer Datei zusammengefaßt werden. Die eigentliche Messung kann dann übersichtlich in der Hauptbefehlsdatei angegeben werden.

Die Syntax lautet:

```
CALL dateiname
```

Beispiel:

Datei messung.cmd:

```
[Gaussmeter]
CALL init.inc
BEFEHL1
...
CALL done.inc
```

Datei init.inc:

```
[Gaussmeter]
LOCKOUT ON
MODE GAUSSAC FILTERON
```

Datei done.inc:

```
[Gaussmeter]
LOCKOUT OFF
```

In diesem Beispiel wird bei Aufruf der Befehlsdatei messung.cmd die eingebundene Datei init.inc aufgerufen, die das Meßgerät in einen definierten Zustand versetzt. Danach werden einige Meßbefehle ausgeführt; als Abschluß wird mit done.inc das Meßgerät wieder freigegeben. Damit kann man mehrere Initialisierungs- und Freigabebefehle zu einem CALL Befehl zusammenfassen. Die Dateinamenserweiterung kann beliebig gewählt werden. Bei Verwendung von .INC kann aber die Webseite diese Befehlsdateien als spezielle aufrufbare Befehle erkennen und entsprechend anzeigen.

3.4.1.5 Befehl SLEEP

Der Befehl SLEEP bewirkt ein Anhalten der Befehlsabarbeitung für die angegebene Zeit in hundertstel Sekunden. Durch einen Aufruf von SLEEP wird die angegeben Zeit gewartet, bevor der nächste Befehl bearbeitet wird.

Die Syntax lautet:

```
SLEEP hs
```

Beispiel:

```
[Gaussmeter]
SLEEP 200
BEFEHL1
```

Hier wird erst einmal 2 Sekunden gewartet, bis mit der Ausführung von Befehl BEFEHL1 fortgesetzt wird.

3.4.1.6 Befehl WAIT

Der Befehl WAIT bewirkt ein Anhalten der Befehlsabarbeitung für die angegebene Zeit in hundertstel Sekunden. Der Unterschied zum SLEEP-Befehl besteht darin, daß WAIT immer den letzten Zeitpunkt, zu dem WAIT verwendet wurde, als Anfangszeit verwendet. Am Beginn einer Befehlsdatei wird die relative Zeit mit 0 initialisiert. Die relative Zeit kann an beliebiger Stelle mit dem Aufruf von WAIT mit dem Parameter 0 neu initialisiert werden. Der erste Aufruf von WAIT wartet die angegebene Zeit vom Anfangszeitpunkt der Befehlsdatei. Die dazwischen liegenden Befehle wirken sich also nicht auf die Zeitmessung aus. Falls die Ausführungszeit der Befehle zwischen zwei WAIT-Befehlen länger dauert als die angegebene Wartezeit, so wird sofort mit der Befehlsausführung nach dem WAIT-Befehl fortgefahren. Mit dem WAIT-Befehl ist es deshalb möglich, ein festes Intervall vorzugeben, mit dem Messungen durchzuführen sind. Immer nach gleich langen Intervallen wird die Messung gestartet, ohne auf die Laufzeit der Befehle Rücksicht nehmen zu müssen. Durch additive Fehler kann es aber bei längeren Messungen zu Zeitungenauigkeiten kommen.

Die Syntax lautet:

```
WAIT hs
```

Beispiel:

```
[Gaussmeter]
BEFEHL1
WAIT 0              ; relative Zeit initialisieren
:AGAIN
BEFEHL2
WAIT 100
GOTO AGAIN
```

Hier wird nach dem Ausführen von BEFEHL1 (es kann sich hier z.B. um Initialisierungen handeln) die relative Zeit mit dem Spezialbefehl WAIT 0 neu initialisiert. Danach wird BEFEHL2 in einer Schleife jede Sekunde ausgeführt. In diesem Beispiel wird angenommen, daß die Ausführungszeit von BEFEHL2 kleiner als 1 Sekunde ist.

3.4.1.7 Befehl SET

Der Befehl SET dient zum Setzen von Parametern. Parameter können als Zeichenketten, als Zahl oder als Formel gespeichert werden. Zeichenketten werden in einfachen Hochkommas eingeschlossen angegeben, um sie von Formeln unterscheiden zu können.

Die Syntax lautet:

```
SET Parameter [Zeichenkette | Formel | Zahl]
```

Der Parameter muß im Abschnitt [PARAMETERS] definiert worden sein. Eine Formel kann Zahlen, Parameter, Konstante und vordefinierte Funktionen enthalten.

Beispiel:

```
[Gaussmeter]
...
SET A 'Text'
SET B 19
SET C SIN(PI)*B
```

Hier wird die Zeichenkette ‚Text' im Parameter A gespeichert. Der Parameter B wird als Zahl 19 definiert. Danach wird der Parameter C aus der angegebenen Formel ausgerechnet, wobei die Konstante PI und der Parameter B verwendet werden. Der Parameter B hat hier schon den Wert 19.

3.4.1.8 Befehl OUT

Der Befehl OUT wird zur Ausgabe von erhaltenen Meßdaten verwendet. Es arbeitet immer mit einem Stream, der als Abschnitt in der INI-Datei definiert sein muß. Der erste Parameter für den OUT-Befehl ist immer der Name eines zugeordneten Streams. Der zweite Parameter kann nur die definierten Steuerbefehle OPEN, LINE und CLOSE enthalten. Der Befehl OPEN bewirkt ein Öffnen der zugehörigen Ausgabedatei und das Auswerten des HEAD Eintrages im angegebenen Stream. Der Name der Ausgabedatei wird als FILE Eintrag im Stream-Abschnitt definiert. Fehlt dieser Eintrag, so wird der Name aus dem Namen der Befehlsdatei abgeleitet. Falls der HEAD Eintrag im Stream-Abschnitt Parameter enthält (die Zeichenkette enthält Fragezeichen), so müssen diese hier angegeben werden. Oftmals wird es sich hier um Achsenbeschriftungen handeln. Der Befehl LINE schreibt eine Zeile in die Ausgabedatei. Er wertet den zugehörigen Eintrag LINE im Stream-Abschnitt aus. Falls dieser Eintrag Parameter enthält, müssen diese hier angegeben werden. Der Befehl CLOSE schreibt gegebenenfalls den Eintrag TAIL aus dem Stream-Abschnitt

in die Ausgabedatei und schließt diese. Mehrere Streams können gleichzeitig geöffnet sein.

Die Syntax lautet:
```
OUT Stream OPEN | LINE | CLOSE Parameter ...
```

Stream muß als Stream-Abschnitt in der INI-Datei definiert sein. Parameter müssen in der gleichen Anzahl angeben werden, wie Fragezeichen im zugehörigen Eintrag im Stream-Abschnitt definiert wurden. Als Sonderfall wird LINE behandelt. Werden hier keine Parameter angegeben, so wird statt dessen der Eintrag PARAMS verwendet.

Beispiel:

```
[Gaussmeter]
OUT DATA OPEN ‘Nummer‘ ‘Wert‘
OUT DATA LINE 1 A
OUT DATA LINE 2 B
OUT DATA CLOSE
```

Hier wird ein Stream mit Namen DATA geöffnet und als Parameter die Zeichenketten ‚Nummer‘ und ‚Wert‘ übergeben, die mit dem HEAD-Eintrag in die Ausgabedatei geschrieben werden. Danach werden 2 Zeilen mit den Parameter A und B in die Datei geschrieben, woraufhin die Datei wieder geschlossen wird. Ein zugehöriger Abschnitt in der INI-Datei könnte folgendermaßen aussehen:

```
[STREAM:DATA]
HEAD=‘;Results#13#10;?,?‘
LINE=‘?,?‘
```

Da der Eintrag FILE fehlt, wird der Name aus der Befehlsdatei abgeleitet. Der Eintrag HEAD erwartet 2 Parameter, da sich in der Zeichenkette 2 Fragezeichen befinden. Der Eintrag LINE schreibt 2 Parameter durch Beistrich getrennt in die Ausgabedatei. Der Eintrag TAIL fehlt, daher bewirkt der Aufruf von OUT DATA CLOSE nur ein Schließen der Datei.

3.5 Anwendungsmöglichkeiten

Das Programm MM kann auf zwei Arten verwendet werden. Einerseits kann es als Hintergrundprogramm laufen und auf Befehle warten, andererseits kann es als normaler Befehl auf der Eingabeaufforderung verwendet werden.

3.5.1 MM als Server

Wird MM ohne die Angabe einer Befehlsdatei gestartet, verhält es sich als Server und wartet auf Befehlsdateien. Diese Art ist der bevorzugte Modus bei Verwendung einer Webseite zur Steuerung von MM. Das Programm MM läuft als DOS-Anwendung auf einem Windows-Meßrechner und überwacht ein vorgegebenes Verzeichnis, in das die Webseite ihre Befehlsdateien stellt. Diese werden vom Programm MM ausgeführt und die Ergebnisse ebenfalls in dieses Verzeichnis gestellt.

3.5.2 MM als Befehlszeilenprogramm

Wird beim Aufruf von MM eine oder mehrere Befehlsdateien angegeben, so werden diese abgearbeitet und das Programm danach beendet. Diese Anwendungsmöglichkeit ist sinnvoll, falls Messungen schnell direkt vor Ort durchgeführt werden müssen und keine Webseite zur Verfügung steht.

4. Die Web-basierte Oberfläche

Eine Webseite soll die Verwendung von MM erleichtern und vor allen die Steuerung von einem beliebigen Arbeitsplatz aus ermöglichen. Die Webseite verwendet PHP3 auf der Serverseite und JavaScript auf der Clientseite. Die Initialisierungsdateien für die Meßgeräte befinden sich im selben Verzeichnis wie die HTML-Dateien. Sie sind daher für die Webseite leicht zugänglich. Dieses Verzeichnis muß für das DOS-Programm auch lesbar zugänglich gemacht werden, damit es die Initialisierungsdateien auswerten und verwenden kann. Ein Arbeitsverzeichnis muß eingerichtet sein. Standardmäßig wird ./tmp als Unterverzeichnis des Webverzeichnisses verwendet. Dieses Verzeichnis muß für das DOS-Programm schreibbar zugänglich gemacht werden.

4.1 Konfiguration

Die Webseite kann mit Hilfe einer Konfigurationsdatei auf die unterschiedlichen Gegebenheiten angepaßt werden. Diese Konfigurationsdatei hat den Namen INIT.PHP und wird von den anderen PHP Skripts verwendet. Sie enthält Konstantendeklarationen, die im folgenden beschrieben werden. Diese Einträge sind unbedingt notwendig und können nicht weggelassen werden. Ein Fehlen führt zu Problemen mit dem PHP-Interpreter.

WORKDIR
Dieser Eintrag dient zum Einstellen des Arbeitsverzeichnisses, wo die Webseite ihre Befehlsdateien ablegt, MM diese auswertet und die Ergebnisse wiederum bereitstellt. Der Default-Eintrag ist ./tmp als temporäres Unterverzeichnis des Webverzeichnisses.

INIDIR
Dieser Eintrag dient zum Einstellen des Verzeichnisses, welches die Meßgeräte-Initialisierungsdateien enthält. Der Default-Eintrag ist . und damit das Webverzeichnis selbst.

DIRSEP
Dieser Eintrag dient zur Definition des Zeichens, welches Verzeichnisnamen für unterschiedliche Betriebssysteme trennt. UNIX-basierte Systeme verwenden den einfachen Schrägstrich (/), während DOS-Systeme den umgekehrten Schrägstrich (\) verwenden.

LOGFILE
Dieser Eintrag enthält den Namen der aktuellen Protokolldatei. Damit kann sehr einfach zwischen mehreren Dateien umgeschaltet werden. Für das DOS-Programm erfolgt die Angabe der Protokolldatei in der Datei mm.cfg.

PASSWORD
Dieser Eintrag definiert ein Kennwort, das benötigt wird, um Meßergebnisse zu löschen.

Als Beispiel die auf dem Webserver des Institutes verwendete Initialisierungsdatei init.php.

```
<?php
 define("WORKDIR", "./tmp");
 define("INIDIR", ".");
 define("DIRSEP", "/");
 define("LOGFILE", "mm.log");
 define("PASSWORD", "xxxxx");
?>
```

4.2 Aufbau

Die Webseite teilt sich in 2 Arbeitsbereiche auf. Auf der einen Seite erfolgt die Eingabe von Meßbefehlen, und die andere Seite dient der Auswertung der Ergebnisse.

4.2.1 Befehlseingabe

Zur Eingabe der Meßbefehle wählt man ‚Control Panel' vom Hauptmenü der Weboberfläche. Die angezeigte Seite enthält ein Eingabefeld für Befehle, die an das Meßgerät geschickt werden sollen. Die Abbildung 1 zeigt dieses Eingabefeld. Eine Befehlsdatei kann Befehle an mehrere Meßgeräte enthalten. Ein Meßgerät wird durch die Eingabe des Namens, umschlossen von eckigen Klammern identifiziert. Dieser Name wird als NAME-Eintrag im Abschnitt METER in einer Initialisierungsdatei definiert. Auf der Oberfläche kann fast alles aus Klapplisten ausgewählt werden. Als erstes muß ein Meßgerät ausgewählt werden. Die Oberfläche fügt daraufhin automatisch den Namen mit eckigen Klammern in das Eingabefeld und zeigt zusätzlich eine Reihe von Klapplisten für das eben ausgewählte Meßgerät an. Diese Eingabefelder sind in Abbildung 2 dargestellt. Die Klapplisten sind wie folgt beschriftet:

Include:
Diese Liste enthält alle vordefinierten Befehlssequenzen. Diese werden im Arbeitsverzeichnis mit der Erweiterung .INC gesucht und verwendet, falls sie zum ausgewählten Meßgerät passen. Die Verbindung erfolgt wieder über den Namen. Wird hier ein Eintrag ausgewählt, so wird in das Eingabefeld ein CALL-Befehl eingebaut, der diese Befehlsdatei im Zuge der Befehlsabarbeitung aufruft.

Command:
Diese Liste enthält die Namen aller definierten Befehle aus dem Abschnitt [COMMANDS] aus der zum ausgewählten Meßgerät gehörenden Initialisierungsdatei. Die Reihenfolge der Befehle wird durch deren Anordnung in der INI-Datei bestimmt. An das Ende der Liste werden die Standardbefehle CALL, DUMP, GOTO, IF, OUT, SET, SLEEP und WAIT angehängt. Wird ein Eintrag in dieser Liste ausgewählt, so wird der Befehl in einer neuen Zeile in das Eingabefeld hinzugefügt.

Constant:
Diese Liste enthält die Namen aller definierten Konstanten und Formeln aus dem Abschnitt [CONSTANTS] aus der zum ausgewählten Meßgerät gehörenden Initialisierungsdatei. Die Reihenfolge der Konstanten wird durch deren Anordnung in der INI-Datei bestimmt. Wird ein Eintrag in dieser Liste ausgewählt, so wird die Konstante an den letzten Befehl im Eingabefeld angehängt.

Parameter:
Diese Liste enthält die Namen aller definierten Parameter aus dem Abschnitt [PARAMETERS] aus der zum ausgewählten Meßgerät gehörenden Initialisierungsdatei. Die Reihenfolge der Parameter wird durch deren Anordnung in der INI-Datei bestimmt. Wird ein Eintrag in dieser Liste ausgewählt, so wird der Parameter an den letzten Befehl im Eingabefeld angehängt.

Stream:
Diese Liste enthält die Namen aller Ausgabestreams, die in der zum ausgewählten Meßgerät gehörenden Initialisierungsdatei mit [STREAM:Name] definiert sind. Die Reihenfolge der Streams wird durch deren Anordnung in der INI-Datei bestimmt. Wird ein Eintrag in dieser Liste ausgewählt, so wird der Parameter an den letzten Befehl im Eingabefeld angehängt. Vorzugsweise sollten Streams nur nach dem Befehl OUT und als Parameter für Meßbefehle, die Streams liefern, verwendet werden.

Action:
Diese Liste enthält immer die Einträge OPEN, LINE und CLOSE und dient nur der einfachen Auswahl der möglichen Befehle für den Befehl OUT.

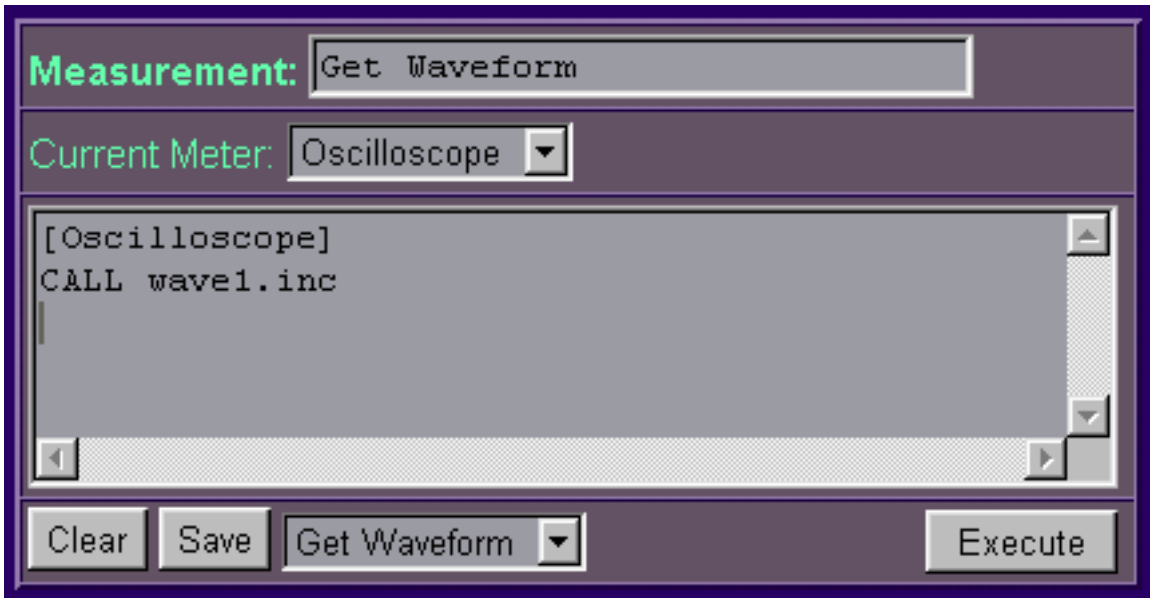

Abbildung 1: Befehlseingabefeld

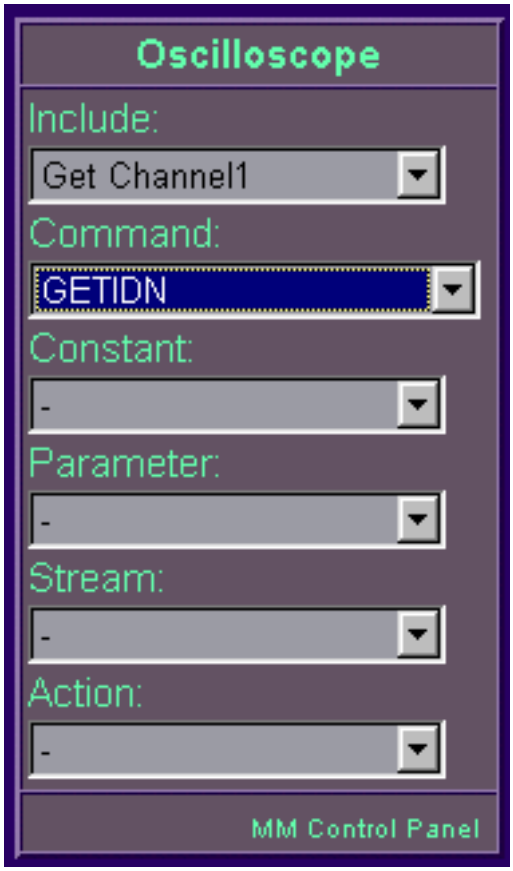

Abbildung 2: Befehls- und Parameterauswahl

Ist eine Befehlsdatei fertig definiert, so kann ihr ein Name zur repräsentativen Beschreibung mittels des Eingabefeldes ‚Measurement' gegeben werden. Mit Hilfe des Buttons ‚Save' kann die aktuelle Befehlsdatei auf dem lokalen Arbeitsplatz als Cookie im Webbrowser gespeichert werden. Dazu muß im Browser die Verwendung von Cookies aktiviert werden. Wird diese Schaltfläche bei leerem Eingabefeld für

Befehle ausgewählt, so bewirkt das ein Löschen der unter dem gegebenen Namen gespeicherten Befehle.
Die eingegebenen Befehle können nun durch Auswahl der Schaltfläche ‚Execute' ausgeführt werden. Die Weboberfläche wechselt zur Seite mit den Ergebnissen.

4.2.2 Ergebnisansicht

Die Ergebnisansicht wird automatisch beim Start einer Messung angezeigt, sie kann aber auch durch Auswahl von ‚Results' aus dem Hauptmenü dargestellt werden. Das Fenster wird wieder in zwei Bereiche aufgeteilt. Im oberen Bereich erfolgt die Darstellung der Ergebnisse, während im unteren Bereich Steuerelemente ihren Platz finden.
Die Ansicht der Ergebnisse kann auf unterschiedliche Art erfolgen. Diese können mit der Klappliste ‚Display' ausgewählt werden. Folgende Ansichtsarten sind zur Zeit möglich:

Information
Diese Ansicht zeigt eine Zusammenfassung über die ausgewählte Messung an. Der Status der Messung sowie alle als Kommentare in die Ergebnisdatei eingefügten Zeilen werden angezeigt. Ebenso wird die Anzahl der Meßergebnisse mitgeteilt. Damit eignet sich diese Anzeige vor allem zur Überwachung einer laufenden Messung. Diese Ansicht wird dargestellt, wenn automatisch beim Start einer Messung auf die Ergebnisansicht gewechselt wird.

Table (Tabelle)
Diese Ansicht stellt alle Meßdaten in einer Tabelle übersichtlich dar. Die erste Zeile in einer Meßdatei sollte immer eine generelle Überschrift für diese Messung sein. Die Überschriften der einzelnen Spalten werden aus der zweiten Kommentarzeile der Ergebnisdatei gewonnen. Die Tabelle verwendet ein Komma als Trennzeichen für die einzelnen Zahlenwerte. Die Anzahl der Zahlen in einer Zeile wird durch den Formateintrag LINE in der Initialisierungsdatei im Abschnitt STREAM bestimmt. Werden mehrere Messungen in eine Ausgabedatei geschrieben, so bestimmt die Messung mit der maximalen Anzahl von Zahlen die Anzahl der Spalten der Tabelle.

Numbers (Zahlen)
Diese Ansicht stellt alle Meßdaten aus der Ergebnisdatei ohne die Kommentare genauso dar, wie sie in der Datei gespeichert sind. Ist z.B. das Format als ?,?,?,? definiert, so werden einzelne Zeilen mit jeweils 4 Zahlen durch Kommas getrennt dargestellt.

Bar chart (Balkengrafik)
Diese Ansicht stellt für jedes Meßergebnis aus der Datei einen Balken dar. Die maximale Anzahl dieser Balken ist standardmäßig auf 20 Balken gesetzt. Diese Beschränkung kann aber aufgehoben werden und auf beliebig hohe Anzahl gesetzt werden. Allzu viele Balken benötigen aber extreme Rechenleistung. Jeder Balken ist programmtechnisch ein GIF-Bild in einer Tabelle. Die gesamte Balkenansicht wird also als eine Tabelle dargestellt. Die Anzeige von 100 Balken ist noch akzeptabel und

erlaubt ein weiches Scrollen und exaktes Überprüfen einer längeren Meßreihe. Ein Beispiel für diese Ansichtsart zeigt Abbildung 3.

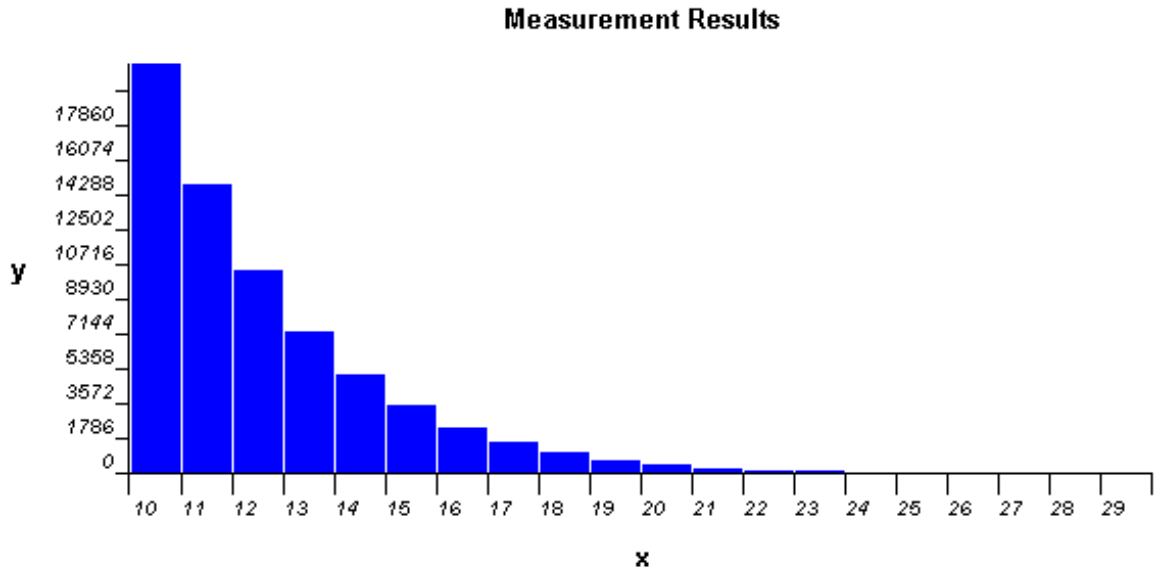

Abbildung 3: Darstellung eines Meßergebnisses mit Balken

Points (Punktegrafik)

Diese Ansicht stellt ein grafisches Fenster dar, das dem Bildschirm eines Oszilloskops nachgebildet wurde. Es erlaubt die Anzeige der Meßdaten als Linienkurve. Falls die Anzahl der Meßdaten 100 Werte überschreitet, werden diese Punkte auf 400 Punkte interpoliert; falls die Anzahl der Meßdaten 400 Werte überschreitet, werden aus diesen Punkten maximal 400 zur Anzeige verwendet. Eine optimale Ansicht erreicht man also, wenn das Meßgerät für eine Kurve exakt 400 Punkte liefert. Das zur Entwicklung verwendete Oszilloskop kann 100, 400, 1000 und 2000 Punkte liefern. Je mehr Punkte geliefert werden, desto genauer können kleine Bereiche analysiert werden. Dazu empfiehlt sich die Balkenansicht. Für die Darstellung als Punktemenge sind 400 Punkte bestens geeignet. Der Wertebereich der einzelnen Punkte liegt zwischen 0 und 255. Falls das Meßgerät einen anderen Bereich vorsieht, können die einzelnen Punkte in der Ergebnisansicht für eine optimale Darstellung transferiert werden.
Die Punkteansicht erlaubt auch, mehrere Kurven in unterschiedlichen Farben darzustellen. Dazu ist es notwendig, die einzelnen Punktemengen in der Ergebnisdatei durch Kommentare eindeutig zu trennen. Alle aufeinander folgenden, nicht durch Kommentare unterbrochenen Ergebnisse werden als zu einer Liniengrafik gehörenden Menge betrachtet.
Falls die Anzahl einer Menge unter 100 Punkten liegt, werden diese Punkte links in der Ansicht dargestellt, ohne sie auf den gesamten Bereich auszudehnen.
Ein Beispiel von einer Messung mit dem Oszilloskop zeigt Abbildung 4.

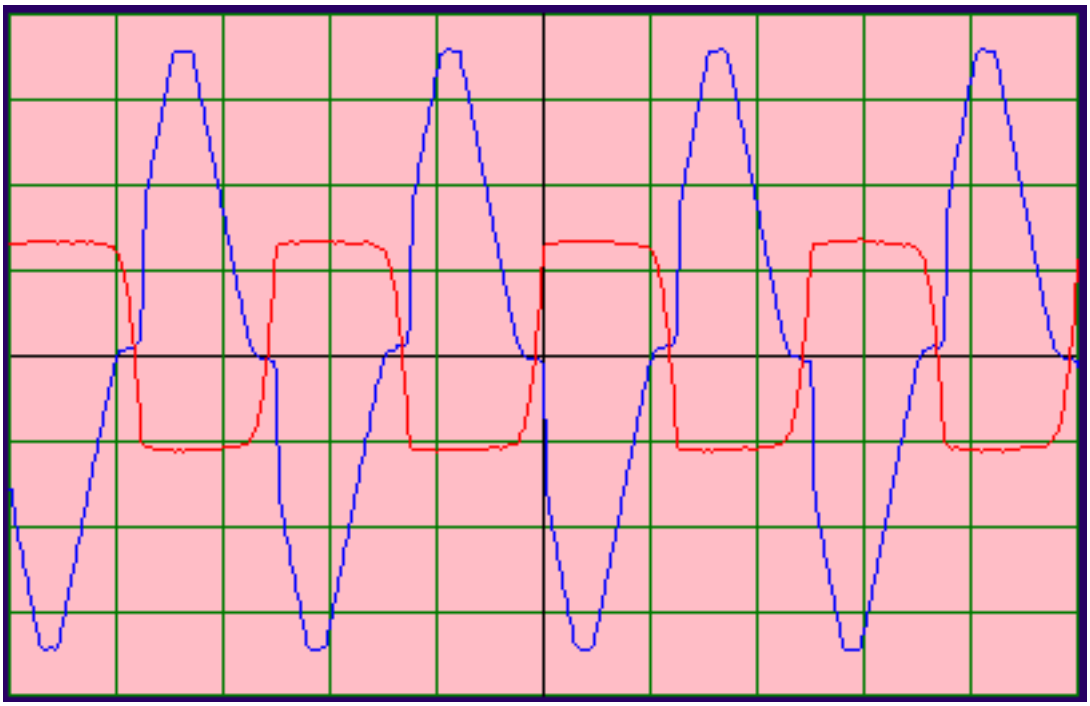

Abbildung 4: Kurvenverlauf, dem eines Oszilloskops nachgebildet

Commands (Befehle)
Diese Ansicht stellt die Befehlsdatei dar, die zu diesen Ergebnissen geführt hat. Ebenso wird der temporäre Name für diese Messung angezeigt. Dieser kann für Verwaltungszwecke notwendig sein.

Die Steuerung dieser Ansichten kann auf vielfältige Weise beeinflußt werden:

First (Erstes Element)
Dieses Eingabefeld erlaubt die Definition des ersten anzuzeigenden Elements. Es wird die Nummer des Elements erwartet, welches als erstes zur Anzeige verwendet werden soll. Ein leeres Feld bedeutet, daß mit dem ersten Ergebniswert die Anzeige begonnen wird. Die Eingabe eines ersten Elements kann z.B. in der Balkengrafik sinnvoll sein, um bestimmte Bereiche näher zu analysieren, ohne alle davor liegenden Punkte anzeigen zu müssen.

Last (Letztes Element)
Dieses Eingabefeld erlaubt die Definition des als letztes anzuzeigenden Elements. Es wird die Nummer des Elements erwartet, welches als letztes zur Anzeige verwendet werden soll. Ein leeres Feld bedeutet hier, daß alle folgenden Elemente zur Ansicht herangezogen werden. Mit Hilfe von First und Last kann ein bestimmter Bereich in der Ergebnismenge dargestellt werden.

Limit (Maximale Anzahl von Elementen)
Dieses Eingabefeld erlaubt die Festsetzung der maximalen Anzahl der darzustellenden Elemente. Ein leeres Feld bedeutet hier, daß alle Elemente angezeigt

werden. Eine Ausnahme bildet die Balkengrafik; hier bedeutet ein leeres Feld, daß maximal 20 Elemente dargestellt werden. Limit hat Priorität vor Last – First, falls diese Anzahl größer als Limit sein sollte.

Factor (Multiplikationsfaktor)
Mit dieser Zahl werden alle Ergebnisse vor der Ansicht multipliziert. Das kann sinnvoll sein, falls das Meßgerät die Ergebnisse in einem anderen Wertebereich liefert, als die Ansicht verwendet. Der gewünschte Wertebereich liegt zwischen 0 und 255 für die Punktegrafik. Für alle anderen Ansichten ist der Wertebereich belanglos. Die Balkengrafik paßt sich dem maximalen Wert der Ergebnisse an.

Add (Verschiebungsfaktor)
Mit dieser Zahl können Ergebnisse in der Höhe verschoben werden. Diese Zahl wird nach einer eventuellen Multiplikation mit einem Faktor auf das Ergebnis addiert. Man kann damit z.B. die Nullinie in einer Grafik neu ausrichten. Wiederum ist dieses Steuerelement für die Punktegrafik gedacht.

Update Now (Jetzt aktualisieren)
Diese Schaltfläche stellt die ausgewählt Ansicht neu dar. Das ist sinnvoll, falls die Messung noch nicht abgeschlossen ist, und deshalb immer wieder neue Ergebnisse eintreffen können. Eine laufende Messung wird mit einer Meldung vor jeder Ergebnisansicht angezeigt.

Update automatically every 10 seconds (Alle 10 Sekunden aktualisieren)
Diese Steuerfläche ermöglicht es, die Ansicht automatisch nach einer bestimmten angegebenen Zeit neu darzustellen. Allzu kleine Sekundenwerte sollte aber vermieden werden, da es sonst wegen der länger dauernden Übertragung der Daten über das Netzwerk zur totalen Blockierung des Browsers kommen kann.

Refresh (Neu messen)
Diese Schaltfläche startet die ausgewählte Messung neu und liefert damit neue aktuelle Werte im Gegensatz zu den Update Funktionen, die nur die vorhandenen Werte neu darstellen. Eine Auswahl von Refresh entspricht also einem neuerlichen Start der aktuellen Messung. Die alten Ergebnisse gehen jedoch dabei verloren. Will man eine Meßreihe aufbauen, so muß man neuerlich auf die Befehlseingabeseite wechseln und die Befehlsdatei neu ausführen. Damit werden neue Meßergebnisse angelegt, und die alten Daten bleiben erhalten. Die Daten sind am Webserver gespeichert und können daher auch später neu angezeigt oder mit anderen Anwendungen weiterverarbeitet werden.

Stop (Messungen abbrechen)
Mit dieser Schaltfläche kann eine laufende Messung beendet werden. Diese Funktion erfordert die Eingabe eines Kennwortes, um unbefugtem Verändern des Meßfortschrittes vorzubeugen.

Delete (Ergebnisse löschen)
Mit dieser Schaltfläche können Ergebnisdaten vom Webserver entfernt werden. Diese Schaltfläche erfüllt zwei Funktionen. Falls die aktuelle Messung gerade noch läuft, so wird sie bei Auswahl dieser Schaltfläche abgebrochen. Bei nochmaliger Auswahl werden die Meßdaten gelöscht. Diese Funktion erfordert die Eingabe eines Kennwortes, um unbefugtem Löschen der Daten vorzubeugen.

Clear (Eingabe löschen)
Diese Schaltfläche löscht alle eingegebenen Werte in den Feldern und stellt sie wieder auf die Standardwerte.

Download (Ergebnisse in einer Datei speichern)
Diese Schaltfläche erlaubt das bequeme Speichern der Ergebnisdaten in einer lokalen Textdatei. Diese Datei kann später mit anderen Anwendungen bearbeitet werden.

4.2.3 Statusansicht

Bei Auswahl von ‚Status' aus dem Hauptmenü wird eine Übersicht über den aktuellen Status des Meßsystems dargestellt. Alle laufenden und verfügbaren MM-Server werden angezeigt. Ebenso werden eventuell gerade laufende Messungen dargestellt. Damit hat man einen Überblick, wie stark das System gerade ausgelastet ist und kann feststellen, warum die eigene Messung nie Daten liefert, falls eine andere Messung gerade das gewünschte Meßgerät blockiert.
Bei Auswahl von ‚Logfile' wird die Protokolldatei über die letzten Messungen angezeigt. Ein Eintrag in dieser Datei zeigt den Zeitpunkt am Meßrechner, an dem die Messung dort zur Ausführung kam. Der Zeitpunkt ist die lokale Zeit am Meßrechner. Ebenso wird der Name der Messung und der Zeitpunkt, zu dem sie am Webserver gespeichert wurde, angezeigt. Falls möglich wird auch der Name des Rechners, von dem aus die Messung gestartet wurde, und seine IP-Adresse angegeben.

4.2.4 Hilfeansicht

Die Hilfe zeigt eine allgemeine Einführung zur Bedienung der Oberfläche und beschreibt die wichtigsten Eingabefelder.

4.3 Unterstützte Browser

Die Webseite wurde zur Verwendung von Netscape Navigator und Microsoft Internet Explorer angepaßt. Zur Steuerung wird JavaScript verwendet, das beide Browser ab einer gewissen Version zur Verfügung stellen. Wird die Seite mit älteren Versionen betrachtet, so ist keine vollständige Anwendung möglich. In der gesamte Implementierung erfolgt keine Abfrage des verwendeten Browsers und danach eine

Verzweigung in unterschiedliche Bereiche. Das wurde bewußt verhindert und nur standardisierte HTML-Befehle verwendet. Falls eine Unterscheidung notwendig ist, beruht sie auf der Abfrage der Fähigkeit und nicht der Version des Browsers. Ein Beispiel hierzu ist das Zeichnen von Punkten. Hier wird geprüft, ob der Browser layer unterstützt. Layer sind Schichten für Zeichenelemente. Falls keine Schichten unterstützt werden, müssen andere HTML-Befehle verwendet werden.
Eingegebene Befehle können als Cookie auf der lokalen Festplatte gespeichert werden. Will man diese Funktion verwenden, sollte man Cookies im Browser erlauben. Ein Cookie ist einfach eine Wertezuordnung zur Speicherung von Parametern, die dann beim neuerlichen Besuch der Seite wieder zur Verfügung stehen. Cookies werden hier verwendet, um eine komplette Meßdatei zu speichern. Leider ist die Anzahl und Größe je nach Browser beschränkt. Deshalb kann man nicht beliebig viele Meßbefehle speichern. Diese Funktion soll aber auch nicht zur Archivierung von großen Beständen dienen, sondern zur schnellen Wiederverwendung der letzten benutzten Befehlssequenzen.

4.4 Bekannte Probleme

In diesem Kapitel soll auf bestehende Probleme mit der Oberfläche hingewiesen werden. Diese Probleme resultieren aus dem Verhalten der verwendeten Webbrowser, das nicht dem Standardverhalten entspricht.

history.back() Problem
Bei Verwendung von Netscape wird der Befehl zum Rücksprung auf die vorhergehende Seite nicht richtig durchgeführt. Einen solchen Rücksprung verwendet die Seite, falls bei Messungen Fehler auftreten, oder die Befehlsdatei ungültige Befehle enthält. Dann wird zur einfachen Fehlerkorrektur nach einer Meldung wieder auf die Eingabeseite verzweigt. Netscape behandelt den history.back() Befehl aber derart, daß einfach die vorherige Seite an die aktuelle angehängt wird. Man merkt dieses Verhalten, wenn man die Seite aktualisieren läßt (Mit Hilfe des Refresh-Buttons der Anwendung). Dann wird die aktuelle Seite neu geladen und danach die vorhergehende. Das ist aber bei Messungen fatal. Diese würden dann neuerlich durchgeführt werden. Um das zu verhindern, lädt die Datei zur Messung jetzt eine leere HTML-Datei, welche dann den History.back() Befehl durchführt. Damit kann sich dieses Problem nicht mehr so stark auswirken. Es ist somit nicht mehr störend für den Ablauf. Allerdings kann man in bestimmten Situationen bei Auswahl des Back-Buttons der Anwendung nicht auf die vorherige Seite wechseln, da immer wieder die Seite geladen wird, die dann ein history.back() ausführt und somit wieder im Kreis zurückspringt.
Dieses Verhalten tritt mit jeder bekannten Netscape Version auf. Eine Lösung für dieses Problem soll im folgenden dargestellt werden.

```
<!-- meter.htm by Johann Smejkal (c) 2001 -->
<!-- Dummy File to reload meter.php -->
<html>
```

```
<head>
<title>Multi Measurement Control Panel</title>
</head>
<body bgcolor="#330066" text="#66FFB2" link="#66FFFF"
vlink="#66FFFF" alink="#66FFFF"
onload="window.history.back(1);">
</body>
</html>
```

Dieses Beispiel zeigt die verwendete Datei mit dem Namen meter.htm. Beim Laden dieser Datei in einem Browser wird durch die Angabe bei onload in der Verlaufsliste einen Schritt zurückgesprungen und damit die vorherige Seite angezeigt.

Dateidownload mit Internet Explorer

Ein Webbrowser zeigt bekannte Dateien auf dem Bildschirm an. Handelt es sich aber um ein unbekanntes Format, fordert er zum Speichern der Datei auf den lokalen Datenträger auf. Man kann diese Aufforderung auch für bekannte Dateien erzwingen, wenn man diese nicht angezeigt, sondern gespeichert haben will. Im konkreten Fall sollen Meßdaten in einer Textdatei gespeichert werden. Netscape verhält sich hier richtig und fordert zur Eingabe eines Namens auf. Der Internet Explorer von Microsoft aber versucht trotzdem diese Datei auf dem Bildschirm darzustellen, falls er erkannt, daß es sich um Text handelt. Damit ist es ein bekanntes Format, und er ignoriert den Wunsch des Programmierers, diese Datei doch zu speichern. Es sind aber nicht alle Versionen betroffen. Jede verhält sich aber unterschiedlich, und es gibt eine Menge Diskussionen darüber, was nun die optimale Lösung ist, damit möglichst immer die Datei gespeichert wird.
Eine Lösung aus Erfahrungen mit mehreren verschiedenen Versionen ist wie folgt:

```
header("Content-Type: application/octet-stream; name=a.txt");
header("Content-Transfer-Encoding: binary");
header("Content-Description: MM Result Data");
header("Content-Disposition: attachment; filename=a.txt");
```

Damit erledigt Netscape seine Aufgabe zufriedenstellend. Leider ignoriert der Internet Explorer, der mit Windows 98 erste Ausgabe geliefert wird, immer noch den Befehl, die Daten doch zu speichern. Neuere Versionen erledigen die Arbeit dann doch noch zufriedenstellend.

Laufwerksverbindungen

Werden auf dem Meßrechner die Laufwerksverbindungen mit der Option „Schnelle Anmeldung“ eingerichtet, treten beim ersten Zugriff von MM Probleme mit den langen Dateinamen auf. Die „Schnelle Anmeldung“ bewirkt, daß das Betriebssystem schnell zur Verfügung steht und die angegebenen Netzwerklaufwerke erst beim ersten Zugriff auf Verfügbarkeit überprüft. Ist nun MM.EXE in der Autostartgruppe enthalten, so geht der erste Zugriff auf das Netzwerklaufwerk von MM.EXE aus. Und der allererste Zugriff ist das Erstellen einer temporären Arbeitsdatei, die die Zuordnung zu einem Meßgerät anzeigt. Das ist deshalb notwendig, um feststellen zu können, ob für ein bestimmtes Meßgerät ein DOS-Programm zur Verfügung steht.

Allerdings kommt es bei dieser Konfiguration dann zu Problemen, da der erste Zugriff auf ein Netzwerklaufwerk von einem DOS-Programm nicht die erwünschte Datei mit langem Namen erzeugt. Windows erzeugt einfach nur den zugehörigen 8.3-Dateinamen ohne einen Fehler zu melden. Der Dateiname entspricht aber dem angegebenen Meßgerätenamen. Somit ist er länger als 8 Zeichen und wird deswegen abgeschnitten mit der Tilde-Notation angelegt. Die Webseite erkennt daher nicht die Verfügbarkeit eines MM-Servers.
Um dieses Problem zu beheben, sollte man die Option „schnelle Anmeldung“ deaktivieren. Da der Meßrechner sowieso immer auf das Netzwerklaufwerk zugreifen muß, ist das keine erhebliche Einschränkung.

5. MM für den Programmierer

5.1 Allgemeiner Aufbau

Das Programm zur Kommunikation mit dem Meßgerät ist in Borland PASCAL Version 7.0 geschrieben. Für die schnelle Abwicklung der seriellen Kommunikation wurde die Assemblersprache verwendet, welche mit Turbo Assembler Version 3.2 übersetzt wurde. Vor allem die leichte Erweiterbarkeit stand bei der Realisierung im Vordergrund. Deshalb wurde das DOS-Programm objektorientiert aufgebaut. Das Hauptprogramm wertet die Konfigurationsdatei mm.cfg und danach die übergebenen Parameter aus. Daraufhin werden alle Dateien mit der Erweiterung .INI im angegebenen Initialisierungsverzeichnis untersucht, und es wird für jede gefundene Meßgeräte-Initialisierungsdatei ein neues Dateiobjekt angelegt. Die kompletten Dateien stehen somit im Speicher zur Verfügung. Der Zugriff auf die Dateien erfolgt nur einmalig beim Programmstart. Das ist deswegen schon sinnvoll, da die Initialisierungsdateien auf einem Netzwerkverzeichnis stehen und der Zugriff damit eine erhebliche Verzögerung der Abarbeitung mit sich ziehen kann. Nachdem diese Objekte erfolgreich aufgebaut wurden, werden, falls ein oder mehrere Befehlsdateien mit übergeben wurden, diese ausgeführt, ansonsten wird das Programm in einer Schleife das Arbeitsverzeichnis nach eintreffenden Befehlsdateien überprüfen. Im konkreten wird auf alle Dateien mit der Erweiterung .RUN geachtet. Wird eine solche gefunden, so öffnet MM die zugehörige .CMD-Datei und legt wieder ein Dateiobjekt an, welches danach ausgeführt und wieder aus dem Speicher entfernt wird. Ändert eine Befehlsdatei Parameter aus einer Initialisierungsdatei, so bleiben diese Änderungen erhalten und beim Start der nächsten Befehlsdatei stehen die neuen Werte in den Parametern.

Das folgende Diagramm in Abbildung 5 soll das Zusammenspiel der einzelnen Programmeinheiten verdeutlichen.

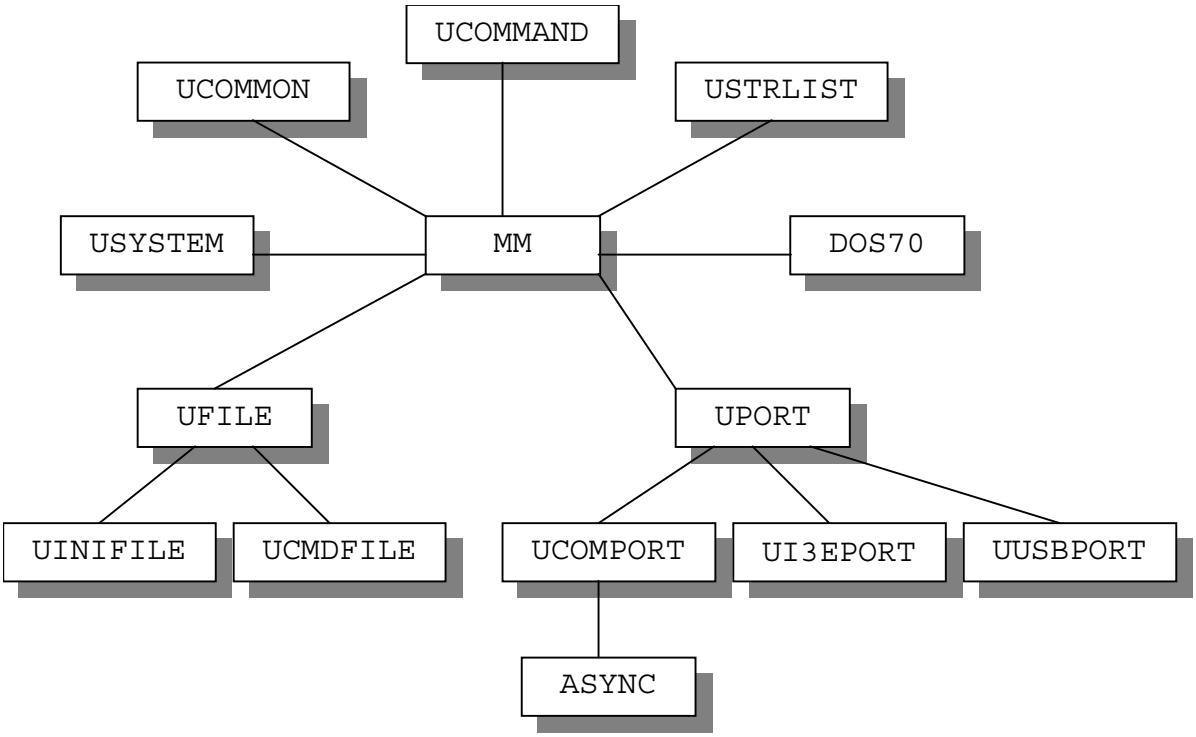

Abbildung 5: Struktur des DOS-Programms MM.EXE

MM
Das Kernstück des Programms, welches die Hauptschleife für den Serverbetrieb enthält und die anderen Module verwendet.

UCOMMON
Diese Datei enthält allgemeine Zeichenkonstanten und Funktionen, die in mehreren Modulen zur Anwendung kommen.

USTRLIST
Diese Datei enthält das Objekt einer Zeichenkettenliste, die zum Speichern von vordefinierten Initialisierungseinträgen verwendet wird. Damit wird das unangenehme Problem von Tippfehlern bei der Verwendung von Initialisierungsdateien vermieden. Falsche Einträge resultieren in einer Fehlermeldung.

USYSTEM
Diese Datei enthält eine Funktion, mit der das Standardfehlerverhalten von PASCAL verändert wird. Mehr dazu im Abschnitt Fehlerbehandlung.

DOS70
Diese Datei enthält Funktionen, damit einer DOS-Anwendung die langen Dateinamen von Windowssystemen zugänglich gemacht werden können.

UFILE
Diese Datei enthält das Objekt einer allgemeinen Datei. Es liest die Datei von einem Datenträger und speichert sie in verwendbarer Form im Arbeitsspeicher.

UINIFILE
Diese Datei enthält das Objekt einer Initialisierungsdatei, einer Datei bestehend aus beliebigen Abschnitten und darin enthaltenen Wertezuweisungen.

UCMDFILE
Diese Datei enthält das Objekt einer Befehlsdatei, einer Datei bestehend aus beliebigen Abschnitten und darin enthaltenen Befehlsketten.

UPORT
Diese Datei enthält das Objekt einer Schnittstelle zur Verbindung des Hauptprogramms mit einem Meßgerät über definierte Ein/Ausgabefunktionen.

UCOMPORT
Diese Datei enthält das Objekt einer seriellen Schnittstelle mit ihren Einstellmöglichkeiten.

ASYNC
Diese Datei enthält die Funktionen zur Kommunikation mit der seriellen Schnittstelle, die von der Datei UCOMPORT verwendet werden. Diese Datei inkludiert ebenso eine kompilierte Assemblerdatei.

UI3EPORT
Diese Datei enthält das Objekt eines Gerätes am IEEE-488-Bus. Die Kommunikation erfolgt über einen Gerätetreiber, der am Meßrechner installiert sein muß.

UUSBPORT
Diese Datei enthält ein leeres Objekt für ein beliebiges Gerät, z.B. am USB Port. Mit Hilfe eines speziellen USB4DOS Gerätetreibers kann dann ähnlich dem UI3EPORT auf diese Meßgeräte zugegriffen werden.

5.2 Fehlerbehandlung

Wenn das DOS Programm MM als Server im Einsatz ist, ist es nicht sinnvoll, daß es bei auftretenden Fehlern beendet wird. Das Standardverhalten von PASCAL Programmen ist aber, mit einer nichts sagenden Fehlermeldung der Art `Runtime Error xxx at xxxx:xxxx` abzubrechen. Die Datei USYSTEM sorgt nun dafür, daß bei auftretenden Problemen bei der Abarbeitung einer Befehlsdatei das Programm nicht beendet wird. Statt dessen wird eine beschreibende Fehlermeldung auf der Webseite ausgegeben und die Behandlung dieser Befehlsdatei eingestellt. Der

einzige Fehlerfall, bei dem MM doch beendet werden muß, ist ein interner Stapelüberlauf der z.B. durch Rekursion schnell erreicht werden kann. Bekannte Ursachen für solche Rekursionen sind jedoch programmtechnisch abfragbar und werden deshalb verhindert. Ein gutes Beispiel ist ein CALL Aufruf an eine Datei, welche die Ursprungsdatei ebenfalls wieder aufruft. Ein weiteres Beispiel sind Rekursionen in Formeln. Theoretisch sollte also ein Stapelüberlauf nicht mehr eintreten können. Falls man aber doch sicher sein will, kann man z.B. MM in einer DOS-BATCH Stapeldatei wiederholt aufrufen, wie das folgende Beispiel zeigt:

```
:AGAIN
MM
GOTO AGAIN
```

Hier wird nach einem Fehlerfall, der zum Beenden von MM führt, das Programm gleich noch einmal gestartet.
Das MM Programm ferngesteuert neu zu starten kann unter Umständen auch nötig sein, falls die Einstellungen in den Initialisierungsdateien geändert wurden. Da diese Dateien auf dem Webserver stehen, kann man sie leicht bearbeiten und dem eigenen Wünschen anpassen. Danach würde aber das DOS-Programm immer noch die alten Einstellungen verwenden, da es, wie bereits erwähnt, nur beim Starten auf diese Dateien zugreift. Um jetzt zu vermeiden, daß der Anwender den Meßplatz aufsuchen muß, gibt es die Möglichkeit, eine Datei mit dem Namen mm.rel in das Arbeitsverzeichnis zu stellen. Die Erweiterung .REL kommt von reload. Diese Datei bewirkt ein Neustarten des MM Programms und damit eine Übernahme der eventuell in den Initialisierungsdateien geänderten Einträgen.

5.3 Objekt einer Datei

MM betrachtet alle verwendeten Dateien als Textdateien, die dem Format einer Initialisierungsdatei entsprechen. Eine Initialisierungsdatei kann mehrere Abschnitte enthalten; diese werden durch deren Namen in eckigen Klammern gekennzeichnet. In jedem Abschnitt können beliebige Einträge enthalten sein. So sieht das Objekt OFILE eine Datei. Sie stellt die Basisklasse für die zwei folgenden Objekte dar, welche die Einträge in einem Abschnitt genauer unterscheiden. Die Deklaration dieses Objektes wird im folgenden genauer erläutert.

```
type PVAssign = ^VAssign;
     VAssign = Record
        Name: PString;
        SValue: PString;
        RValue: Real;
        Line: Word;
        Next: PVAssign;
     end;

     PSection = ^Section;
```

```
Section = Record
   ID: PString;
   List: PVAssign;
   Line: Word;
   Next: PSection;
end;

POFile = ^OFile;
OFile = Object
   FileName: String;
   LineNumber: Word;

   constructor Init(fAllowDup: Boolean);
   function Add(SectionName, Name: String;
     PValue: PString; RValue: Real;
     Line: Word): Boolean;
   function Get(SectionName, Name: String;
     VAR PValue: PString; VAR SValue: String;
     VAR RValue: Real): Boolean;
   function SetV(SectionName, Name: String;
     PValue: PString; RValue: Real;
     DoEval: Boolean): Boolean;
   function EvaluateValue(Name: String;
     VAR PValue: PString;
     VAR RValue: Real): Boolean;
   function FindSection(SectionName: String):
     PSection;
   function GetNextSection(VAR SectionName:
     String): Boolean;
   function GetNextAssign(VAR Name, SValue:
     String; VAR RValue: Real): Boolean;
   procedure DoReset(Section, Assign: Boolean);
   procedure Error(Info: String);
   destructor Done;
   private
   SList: PSection;
   AllowDup: Boolean;
   ActSection: PSection;
   ActAssign: PVAssign;
end;
```

VAssign

Dieser Typ beschreibt eine Zeile innerhalb eines Abschnittes einer Datei. Er enthält verschiedene Variablen zur Speicherung eines Namens (Name), eines Wertes (SValue), einer Zahl (RValue) und einer Zeilennummer für diesen Eintrag relativ zum Anfang der Datei. Für Zeichenketten wird hier ein Zeiger verwendet, der benötigte Platz wird dynamisch angelegt, um keinen kostbaren Speicher zu verschwenden. Falls der Eintrag ein Gleichheitszeichen (‚=') enthält, wird der linke Teil in der Variable Name gespeichert, und der rechte Teil wird je nach Typ als Zeichenkette in der Variable SValue oder als Zahl in RValue gespeichert. Wenn der

Eintrag kein Gleichheitszeichen enthält, so wird die komplette Zeile in der Variable Name gespeichert.

PSection
Dieser Typ beschreibt einen Abschnitt in einer Datei. Ein Abschnitt besteht aus einem Namen und mehreren Einträgen. Der Name es Abschnittes wird ohne die eckigen Klammern in der Variable ID gespeichert. Die Variable List enthält eine lineare Liste mit allen gefundenen Einträgen unterhalb dieses Abschnittes. Die Variable Line gibt wiederum die Position des Abschnittes innerhalb der Datei an. Sie dient hauptsächlich dazu, Fehlermeldungen ansprechend zu formatieren. Der Sonderfall, daß in einer Datei zwei Abschnitte mit dem selben Namen auftauchen, wird in den verschiedenen Kinderobjekten unterschiedlich behandelt.

OFile
Dieses Objekt beschreibt eine Datei mit ihren Zugriffsfunktionen. Sie stellt die Basisklasse für die Initialisierungsdatei und die Befehlsdatei dar und enthält somit alle gemeinsamen Eigenschaften.
Die Variable FileName enthält den absoluten Dateinamen der zugeordneten Datei, die Variable LineNumber enthält die aktuelle Zeilennummer beim Durcharbeiten der Datei, damit wird ein Parameter für alle Funktionen eingespart. Die Zeilennummer dient im Fehlerfall zur schnellen Lokalisierung des Fehlers.
Der Konstruktor Init initialisiert das Dateiobjekt. Die ihm übergebenen Variable fAllowDup zeigt an, ob Duplikate innerhalb eines Abschnittes erlaubt sind.
Die Funktion Add fügt einen weiteren Eintrag in den angegebenen Abschnitt hinzu. Zur Typunterscheidung wird folgendes Prinzip angewandt: Ist der Parameter PValue gleich dem Nullpointer, dann enthält der Parameter eine Zahl, die in der Variable RValue gespeichert ist. Ist der Parameter PValue ungleich dem Nullpointer, so zeigt RValue an, ob es sich um eine Formel (0) oder um eine Zeichenkette handelt (1).
Die Funktion Get liefert die Daten zu einem gegebenen Namen in einem Abschnitt zurück. Wiederum wird das selbe Prinzip angewandt, um zu unterschieden, ob es sich um eine Zahl, eine Zeichenkette oder eine Formel handelt. Hier taucht eine neue Variable SValue auf, die auch für Zahlen eine Zeichenkettenabbildung des zurückgelieferten Wertes enthält. Das ist sinnvoll, da im Programmablauf meistens mit Zeichenketten operiert wird.
Die Funktion SetV ändert einen Eintrag mit dem gegebenen Namen in einem Abschnitt. Die Variable DoEval entscheidet, ob beim Speichern der übergebenen Werte diese ausgewertet oder direkt gespeichert werden sollen. Die Auswertung betrifft hier die speziellen Abschnitte in einer Initialisierungsdatei, die Parametern neue Werte zuordnen. Diese Zuordnung ist sinnvoll, wenn ein Ergebnis von einem Meßbefehl gespeichert werden soll. Wird aber der SET-Befehl verwendet, um einen Wert zu speichern, so sollte dieser direkt in dem Parameter hinterlegt werden und nicht erst umgewandelt.
Die Funktion EvaluateValue implementiert diese Umsetzung der Parameter. Sie sucht in der Initialisierungsdatei nach einem Abschnitt mit dem Namen des Parameters. Findet sie einen solchen, so wird ein Eintrag mit dem Namen des Wertes

gesucht und angewendet, falls er gefunden wird. Ansonsten wird der DEFAULT-Eintrag verwendet. Wenn auch dieser Eintrag fehlt, wird der Parameter ohne Änderungen gespeichert.
Die Funktion FindSection sucht einen Abschnitt mit dem übergebenen Namen und liefert einen Zeiger darauf zurück. Sie kann zur Überprüfung auf Existenz eines bestimmten Abschnittes verwendet werden.
Die Funktion GetNextSection liefert solange den nächsten Abschnittsnamen zurück, wie Abschnitte in der Datei vorhanden sind.
Die Funktion GetNextAssign liefert alle Einträge innerhalb des aktuellen Abschnittes zurück.
Die Funktion DoReset setzt den aktuellen Abschnitts- und den aktuellen Eintragszeiger wieder auf den Anfang zurück. Diese Funktionen dienen der einfachen Auflistung und Abarbeitung einer Datei.
Der Destruktor Done gibt den verwendeten Speicher wieder frei.
Die Variable SList speichert eine lineare Liste mit den Abschnitten aus der Datei.
Die Variable AllowDup speichert den bei der Konstruktion des Objektes übergebenen Wert.
Die Variablen ActSection und ActAssign enthalten die Zeiger auf die aktuell von den GetNext-Funktionen behandelten Variablen. Diese Variablen dienen der internen Verwaltung und sind deshalb von außen nicht zugänglich.

5.3.1 Objekt einer Initialisierungsdatei

Das Objekt OINIFILE baut auf die Basisklasse OFILE auf und speichert alle relevanten Daten für die Verwendung einer Initialisierungsdatei. Die Deklaration dieses Objektes wird im folgenden näher erklärt.

```
type PIniFile = ^OIniFile;
     OIniFile = Object(OFile)
        constructor Init(FName: String);
        private
        DosHandle: Text;
        lfnHandle: Word;
        function ReadIniFile: Integer;
     end;

type PIniList = ^IniList;
     IniList = Record
        Ini: PIniFile;
        Next: PIniList;
     end;
```

OIniFile
Dieses Objekt baut auf der Basis von OFile auf und implementiert im allgemeinen nur das Auslesen der Daten aus einer Datei und das Speichern in den definierten Variablen.

Dem Konstruktor Init wird hier der Name der zu verwendenden Datei übergeben. Die Klasse selbst bestimmt bereits den Wert von fAllowDup. Für Initialisierungsdateien ist dieser Wert auf FALSE gesetzt. Es sind keine doppelten Einträge innerhalb eines Abschnittes erlaubt.
Die Funktion ReadIniFile liest die Datei zeilenweise aus und speichert die gefundenen Daten in dem Objekt. Alle Einträge in einer Initialisierungsdatei müssen ein Gleichheitszeichen enthalten.
Die Variablen DosHandle und lfnHandle dienen zum Speichern der Dateizeiger. Es sind zwei Variablen notwendig, da MM sowohl die guten alten DOS-Funktion sowie die neuen langen Dateinamen von Windows unterstützt. Läuft MM unter einem 32-bit Windows Betriebssystem, so werden automatisch die Funktionen für lange Dateinamen verwendet. Ansonsten wird auf die herkömmlichen 8.3 Dateinamen zugegriffen. Wird MM als Server eingesetzt, so sollte man es unter Windows verwenden, damit keine Probleme mit dem Webserver auftreten können. Beim Einsatz als Befehlszeilenprogramm kann man MM getrost auch unter DOS starten.

IniList
Dieser Typ beschreibt eine lineare Liste mit allen gefundenen Initialisierungsdateien. Eine Variable dieses Typs wird im Hauptprogramm angelegt und damit auf alle vorhandenen Dateien zugegriffen.

5.3.2 Objekt einer Befehlsdatei

Das Objekt OCMDFILE baut auf die Basisklasse OFILE auf und speichert alle relevanten Daten für die Verwendung einer Befehlsdatei. Die Deklaration dieses Objektes entspricht im wesentlichen dem des OINIFILE Objektes.
Die Funktion ReadCmdFile liest die Datei zeilenweise aus und startet mit dem ersten gefundenen Abschnitt, speichert alle Einträge und schließt diese Datei. Es ist auch möglich, mehrere Einträge in einer Befehlsdatei zu verwalten. Das ist nur beim Einsatz als Server sinnvoll. Dann kann in der .RUN Datei die Zeilennummer übergeben werden, am dem die Funktion ReadCmdFile mit dem Auswerten der Datei starten soll. Das ist deshalb notwendig, da eine Befehlsdatei Befehle für verschieden Meßgeräte enthalten kann. An Hand eines Beispieles soll dieses Verhalten erklärt werden.

Ein Anwender gibt folgende Zeilen auf der Webseite ein:

```
[Gaussmeter]
BEFEHL1

[Oscilloscope]
BEFEHL2

[Gaussmeter]
BEFEHL3
```

Hier stehen BEFEHLX für beliebige gültige Befehle an das jeweilige Meßgerät. Startet der Anwender nun diese Befehlsdatei, legt die Webseite eine .RUN Datei mit dem folgendem Inhalt an:

```
[Gaussmeter]
```

Damit kommt der MM Server an die Reihe, der das Meßgerät mit dem Namen Gaussmeter verwaltet. Er liest die Datei ein und baut eine Befehlsdatei für dieses Meßgerät auf. Trifft er auf einen neuen Abschnitt so legt er nach erfolgreicher Ausführung von BEFEHL1 eine neue .RUN Datei mit gleichem Namen an, die folgenden Inhalt aufweist:

```
[Oscilloscope]
4
```

In der zweiten Zeile einer .RUN Datei kann also die Anfangszeilennummer für den MM Server angegeben werden. Der Server mit der Zuständigkeit für das Oszilloskop wird nun aktiv und beginnt diese Befehlsdatei ab der Zeile Nummer 4 auszuwerten. Er trifft wiederum auf einen neuen Abschnitt und legt wieder eine .RUN Datei für den Server des Gaußmeters an. Als Zeilennummer wird er diesmal 7 eintragen.

5.4 Objekt eines Ports

Für die Kommunikation mit den Meßgeräten verwendet das Programm MM ein Objekt, welches hauptsächlich aus einem Schreiben und Lesen auf eine beliebige Schnittstelle besteht. Voraussetzung dafür ist, daß das Meßgerät Befehle und Antworten in einer textuellen Form versteht und sendet, die in einer Zeichenkette gespeichert werden können. Die Befehle bestehen also aus einer Folge von Zeichen aus dem ASCII Code. Will das Meßgerät z.B. als Antwort eine 1 schicken, so sendet es das Zeichen ‚1' mit dem ASCII Code 49 und nicht die binäre Darstellung als 01.
Die Deklaration des Objektes OPort wird im folgenden dargestellt:

```
type PPort = ^OPort;
     OPort = Object
        TimeOut: Word;
        CommandDelay: Word;
        TimeLimit: Word;
        Retries: Byte;
        CurrentRetry: Byte;
        Busy,
        Ready,
        SOT,
        EOT,
        SOR,
        EOR: String[4];
        Ini: PIniFile;
```

```
    DataPending: Boolean;
    CharsTransfer: Longint;

    constructor Init(pini: PIniFile);
    function Write(S: String): Boolean; virtual;
    function Read(VAR S: String): Boolean;
    virtual;
    procedure Clear; virtual;
    function GetPortName: String; virtual;
    function LockPort: Boolean; virtual;
    destructor Done; virtual;
    private
    DosLockFile: File;
    LongLockFile: Word;
end;
```

OPort

Dieses Objekt definiert eine Schnittstelle für die Kommunikation mit einem Meßgerät. Es enthält Funktionen für das Senden von Befehlen und Empfangen von Antworten. Die verwendeten Variablen werden aus der zugehörigen Initialisierungsdatei gewonnen.

TimeOut gibt die Zeit an, die maximal auf eine Antwort vom Meßgerät gewartet wird. Der Standardwert entspricht 2 Sekunden.

CommandDelay gibt die Wartezeit nach jedem abgearbeitetem Befehl an, nach der erst mit dem folgenden Befehl fortgefahren wird. Der Standardwert ist hier 0, was keine Verzögerung bedeutet.

TimeLimit gibt die maximale Ausführungszeit für eine Befehlsdatei an. Nach Ablauf dieser eingestellten Zeit wird eine laufende Befehlsausführung abgebrochen. Der Standardwert ist hier 0, d.h. es besteht keine Zeitbeschränkung.

Retries gibt die Anzahl der Versuche an, um eine gültige Antwort vom Meßgerät zu erhalten. Mit steigender Übertragungsgeschwindigkeit steigt auch die Fehleranfälligkeit.

Die Variable CurrentRetry enthält die Nummer des aktuellen Versuches, den Befehl doch noch richtig zu übertragen.

Busy enthält maximal vier Zeichen, die als Kennzeichnung für einen Zustand dienen, daß keine weiteren Befehle an das Meßgerät geschickt werden können.

Ready enthält maximal vier Zeichen, die als Kennzeichnung dienen, daß neue Befehle wieder übertragen werden können.

SOT, EOT, SOR, EOR enthalten maximal vier Zeichen zur Kennzeichnung bestimmter Zeichenfolgen zur Steuerung der Befehlsübertragung, deren Details bereits im Abschnitt über die Initialisierungsdatei beschrieben wurden.

Die Variable Ini enthält einen Zeiger auf die dem Port zugeordnete Initialisierungsdatei.

DataPending kennzeichnet den Zustand, daß nach dem Einlesen der maximalen Zeichenanzahl in eine Variable immer noch Daten vorhanden sind. Die Verwendung von Zeichenketten in PASCAL ist auf eine Länge von 255 Zeichen beschränkt. Deshalb wird die Übertragung bei mehr als 255 Zeichen unterbrochen, und diese Variable zeigt an, daß ein neuerlicher Aufruf der Lesefunktion notwendig ist.

CharsTransfer dient nur der Information über die Geschwindigkeit der Übertragung und enthält die Anzahl aller über die Schnittstelle übertragenen Zeichen.
Der Konstruktor Init legt ein neues Objekt an und initialisiert die verwendeten Variablen, deren Werte er aus der zugehörigen Initialisierungsdatei gewinnt, die als Parameter übergeben wird.
Die Funktion Write schreibt einen Befehl auf die Schnittstelle. Die Implementierung als virtuelle Funktion bedeutet, daß bei Überschreibung in geerbten Objekten immer die richtige zugehörende Funktion aufgerufen wird.
Die Funktion Read liest eine Antwort vom Meßgerät in eine Zeichenkette ein. Sie kehrt entweder nach einer maximalen Zeit, die durch die Variable TimeOut definiert ist, oder durch den Empfang von EOR zurück.
Die Funktion Clear dient zum Rücksetzen der Schnittstelle. Sie löscht eventuell anstehende Antworten von früheren Befehlen und ermöglicht somit die beste Ausgangssituation für einen neuerlichen Versuch der Übertragung.
Die Funktion GetPortName liefert einen beschreibenden Namen der aktuellen Schnittstelle zurück. Dieser Name ist im allgemeinen gleich dem Eintrag PORT in der zugehörenden Initialisierungsdatei.
Die Funktion LockPort dient dazu, den Zugriff auf eine Schnittstelle exklusiv zu ermöglichen. Es wird dabei unabhängig vom verwendeten Port eine Datei im Arbeitsverzeichnis angelegt und exklusiv geöffnet. Damit ist ausgeschlossen, daß andere MM Instanzen gleichzeitig auf eine Schnittstelle zugreifen. Dazu werden die Variablen DosLockFile und LongLockFile verwendet.
Der Destruktor Done gibt verwendeten Speicher frei und schließt einen eventuell geöffneten Port.

5.4.1 Objekt eines seriellen COM Ports

Das Objekt OCOMPORT erbt die Eigenschaften der Basisklasse OPORT und überschreibt die Funktionen Init, Write, Read, Clear und Done.
In der Funktion Init wird der in der Initialisierungsdatei angegebene serielle Anschluß gemäß den eingestellten Parametern geöffnet und steht damit den Read und Write Funktionen zur Verfügung.

5.4.2 Objekt eines IEEE Ports

Das Objekt OI3EPORT erbt die Eigenschaften der Basisklasse OPORT und überschreibt die Funktionen mit den für diese Schnittstelle notwendigen Befehlen.

5.4.3 Objekt eines USB Ports

Das Objekt OUSBPORT stammt ebenfalls von der Basisklasse OPORT ab und stellt eine Maske für Erweiterungen mit zusätzlichen Schnittstellen dar.

5.5 Erweiterung des Programms mit zusätzlichen Ports

Das Programm MM ist derart aufgebaut, daß es einfach möglich ist, neue Arten von Schnittstellen hinzuzufügen.

5.5.1 Hinzufügen eines neuen Port-Objektes

Als Ausgangsbasis kann die Datei UUSBPORT.PAS verwendet werden. Diese enthält bereits ein gültiges Beispiel mit allen notwendigen Funktionen für eine neue Schnittstelle.

5.5.2 Implementierung der Funktionen

Um ein korrektes Funktionieren zu ermöglichen, müssen bestimmte Funktionen in der neuen Objektdatei von der Basisklasse abgeleitet werden. Folgende Funktionen werden vom Hauptprogramm aus aufgerufen:

Init
Diese Funktion dient der Initialisierung für eventuell benötigte Variablen der neuen Schnittstelle. Als erster Aufruf in dieser Funktion sollte die Basisklasse aufgerufen werden, damit die bereits vorhandenen Variablen auf die Startwerte gesetzt werden können. Ein Beispiel soll die Verwendung eines solchen Aufrufes demonstrieren.

```
{ Basisklasse rufen }
if (NOT inherited Init(pini)) then Fail;
```

Falls der Aufruf der Basisklasse nicht gelingt, so wird auch die Erzeugung des neuen Objektes mit einem Fehler abgebrochen.

Write
Diese Funktion dient dazu, eine Zeichenkette an das Meßgerät zu übertragen. Sie sollte unbedingt überschrieben werden.

Read
Diese Funktion liest eine Zeichenkette als Antwort vom Meßgerät. Sie sollte unbedingt überschrieben werden.

Clear
Mittels dieser Funktion kann das Meßgerät bzw. die Schnittstelle in einen definierten Zustand versetzt werden.

GetPortName
Diese Funktion kann überschrieben werden, um einen aussagekräftigen Namen für die Schnittstelle zurückzuliefern, welcher in Fehlermeldungen verwendet wird.

Done
Diese Funktion kann zum Schließen der neuen Schnittstelle und Freigeben von eventuell verwendeten Speicher verwendet werden. Als letzter Aufruf in dieser Funktion sollte die Funktion der Basisklasse aufgerufen werden, damit auch diese ihre Variablen wieder freigeben kann. Ein Aufruf in dieser Funktion könnte folgenden Aufbau haben:

```
{ Basisklasse rufen }
inherited Done;
```

5.5.3 Einbinden in das Hauptprogramm

Das Einbinden der neuen Schnittstelle im Hauptprogramm erfolgt in der Funktion InitPort. Diese Funktion liest den Namen der Schnittstelle aus der aktuell verwendeten Initialisierungsdatei und legt ein Objekt je nach gefundenen Namen an. Somit wird für die Zeichenketten COM1, COM2, COM3 und COM4 ein Objekt vom Typ OCOMPORT angelegt, für die Zeichenkette IEEE wird ein Objekt vom Typ OI3EPORT und für die Zeichenfolge USB wird ein Objekt mit Type OUSBPORT angelegt. Die neue Schnittstelle hat beispielsweise den Namen FIREWIRE. Somit muß die Funktion InitPort derart erweitert werden, daß bei Auftreten dieser Zeichenfolge ein zugehöriges Objekt beispielsweise OFIRPORT angelegt wird. Dieser Abschnitt aus dieser Funktion wird im folgenden dargestellt.

```
.......
if ((v = PORT_COM1) OR (v = PORT_COM2) OR
    (v = PORT_COM3) OR (v = PORT_COM4) then begin
    { neuen COMPORT erzeugen }
    PCom := New(PComPort, Init(runitem^.ini));
end else if (v = PORT_IEEE) then begin
    { neuen IEEEPORT erzeugen }
    PCom := New(PI3EPort, Init(runitem^.ini));
end else if (v = PORT_USB) then begin
    { neuen USBPORT erzeugen }
    PCom := New(PUSBPort, Init(runitem^.ini));
end else if (v = PORT_FIREWIRE) then begin
    { neuen FIREWIREPORT erzeugen }
    PCom := New(PFIRPort, Init(runitem^.ini));
end;
.......
```

In diesem Ausschnitt aus der Funktion InitPort wird angenommen, daß die Konstanten PORT_XXXX ihre zugeordneten Zeichenkette enthalten, die auch in den Initialisierungsdateien auftauchen. Die Verwendung von Konstanten für Zeichenketten innerhalb des Programms ermöglicht ein rasches Anpassen an neue Bedürfnisse, da die Zeichenketten nur an einer gemeinsamen Stelle stehen und damit leicht und übersichtlich geändert werden können.

6. Beispiel

In diesem Kapitel soll ein komplettes Anwendungsbeispiel besprochen werden, wie es am Institut installiert ist.

6.1 Die Meßgeräte

Zur Verfügung stehen ein BELL Modell 9500 Gaußmeter und ein HP 54600 Oszilloskop, die über die serielle Schnittstelle angesprochen werden. Sie müssen über ein serielles Kabel mit dem Meßrechner verbunden werden, und eventuelle Übertragungsparameter sollten eingestellt werden.

6.2 Der Meßrechner

Als Meßrechner dient ein 80486-Rechner, auf dem Windows 98 Second Edition installiert wurde. Der Zugriff auf die Meßgeräte erfolgt über die serielle Schnittstelle. Der Meßrechner ist mit 2 Netzwerkfreigaben auf dem Webserver verbunden. Eine Freigabe beinhaltet die Initialisierungsdateien für die beiden Meßgeräte, die zweite Freigabe ist das Arbeitsverzeichnis für die Befehlsdateien. Diese Freigaben werden beim Starten des Rechners automatisch hergestellt. Das DOS-Programm MM.EXE selbst liegt auf dem Webserver und wird in der Autostart Gruppe des Betriebssystems automatisch bei jedem Start geladen. Der Meßrechner ist damit fertig konfiguriert.

6.3 Der Webserver

Als Server dient der UNIX-basierte Webserver des Institutes. Auf diesem wurden die HTML-Dateien und PHP3-Skriptdateien gespeichert. Ebenso enthält dieser Rechner die Initialisierungsdateien für die beiden verwendeten Meßgeräte.

6.3.1 Initialisierungsdatei GAUSS.INI

Das Gaußmeter wird durch die folgende Initialisierungsdatei komplett beschrieben.

```
; INI File for BELL Gaussmeter 9500
[METER]
```

```
NAME=Gaussmeter         ; use this name in CMD-Files
PORT=COM1               ; COM1, COM2, COM3, COM4
;BAUDRATE=9600          ; 50 - 115200
;PARITY=N               ; (N)ONE, (E)VEN, (O)DD, (M)ARK,
(S)PACE
;DATABITS=8             ; 5 - 8
;STOPBITS=1             ; 1, 2
;INT=4                  ; Interrupt for this comport
;TIMEOUT=200            ; Timeout in 10ms
;RETRIES=3              ; Number of how many times to try to
send
COMMANDDELAY=100        ; Delay between sent commands in 10ms
EOR=#13                 ; End of Receive Sequence
SOT=$1B                 ; Start of Transmit Sequence
EOT=#13                 ; End of Transmit Sequence
BUSY=#13                ; Meter is busy
READY=#11               ; Meter is ready

[STREAM:DATA]
HEAD=';Measurement Results#13#10;?,?'
LINE='?,?'

[PARAMETERS]
RANGE=0
SIGN=0
READING=0
TESLARANGE=0
A=0
B=0
C=0

[CONSTANTS]
;***************** Formulas
GRESULT=SIGN * READING * RANGE
TRESULT=SIGN * READING * TESLARANGE
;***************** Constants
; MODE Constants
GAUSSAC=1
GAUSSDC=2
TESLAAC=3
TESLADC=4
FILTEROFF=1
FILTERON=2
; RANGE Constants
AUTORANGE=8
; PEAK Constants
PH_OFF=1
PH_ON=2
PH_RESET=3
; COMFORMAT Constants
PARITY_NONE=1
```

```
PARITY_ODD=2
PARITY_EVEN=3
5BIT=1
6BIT=2
7BIT=3
8BIT=4
1STOP=1
1.5STOP=2
2STOP=3
BAUD110=1
BAUD150=2
BAUD300=3
BAUD600=4
BAUD1200=5
BAUD2400=6
BAUD4800=7
BAUD9600=8
BAUD19200=9
; LOCKOUT Constants
OFF=1
ON=2

[RANGE]
DEFAULT=1
1=1
2=10
3=100
4=1000
5=10000
6=100000

[SIGN]
DEFAULT=1
-=-1

[TESLARANGE]
DEFAULT=1
1=0.0001
2=0.001
3=0.01
4=0.1
5=1
6=10

[COMMANDS]
MODE=MO1?|?
MODE_RESPONSE=MO1%?|%%?
RANGE=RA1?
RANGE_RESPONSE=RA1%?
PEAK=PE1?
PEAK_RESPONSE=PE1%?
```

```
DISPLAYON=DI1222
DISPLAYON_RESPONSE=DI1222
DISPLAYOFF=DI1111
DISPLAYOFF_RESPONSE=DI1111
IEEEFORMAT=CO1??00
IEEEFORMAT_RESPONSE=CO1!??00
COMFORMAT=CO2?|?|?|?
COMFORMAT_RESPONSE=CO2!????
SETUPSAVE=SE1?
SETUPSAVE_RESPONSE=SE1!?
SETUPLOAD=SE2?
SETUPLOAD_RESPONSE=SE2!?
PROBEZERO=ZE10
PROBEZERO_RESPONSE=ZE10
PROBE=RE1220000000
PROBE_RESPONSE=RE122?|?|?????
MEASURE=ME100000000
MEASURE_RESPONSE=ME10?|?|?????
LOCKOUT=LO?
LOCKOUT_RESPONSE=LO%?
CALIBRATE=CA1
CALIBRATE_RESPONSE=CA1
```

6.3.2 Initialisierungsdatei OSCI.INI

Das Oszilloskop wird durch die folgende Initialisierungsdatei komplett beschrieben.

```
; INI File for HP 54600-Series Oscilloscopes
[METER]
NAME=Oscilloscope      ; use this name in CMD-Files
PORT=COM2              ; COM1, COM2, COM3, COM4
;BAUDRATE=9600         ; 50 - 115200
;PARITY=N              ; (N)ONE, (E)VEN, (O)DD, (M)ARK,
(S)PACE
;DATABITS=8            ; 5 - 8
;STOPBITS=1            ; 1, 2
;INT=4                 ; Interrupt for this comport
;TIMEOUT=200           ; Timeout in 10ms
;RETRIES=3             ; Number of how many times to try to
send
COMMANDDELAY=10        ; Delay between sent commands in 10ms
EOR=#10                ; End of Receive Sequence
EOT=#10                ; End of Transmit Sequence
BUFFERSIZE=8192

[STREAM:DATA]
HEAD=';Measurement Results#13#10;x,y,S,V'
LINE='?,?,?,?'
PARAMS=#COUNT #VALUE TIME VOLTAGE
```

```
[STREAM:INFO]
HEAD=';Measurement Results#13#10;Information'
LINE=';? = ? ?'

[PARAMETERS]
; Preamble Information
FORMAT=0
TYPE=0
POINTS=0
COUNT=0
XINC=0
XORG=0
XREF=0
YINC=0
YORG=0
YREF=0
X=0
Y=0
S=0
A=0
B=0
C=0

[CONSTANTS]
; Formulas
V/DIV=32 * YINC
OFFSET=(128 - YREF) * YINC + YORG
S/DIV=POINTS * XINC / 10
DELAY=(POINTS/2 - XREF) * XINC + XORG
VOLTAGE=(#VALUE - YREF) * YINC + YORG
TIME=(#COUNT - XORG) * XINC + XORG
; Parameters
AC='AC'
ASCII='ASCII'
AUTLEVEL='AUTLEVEL'
AUTO='AUTO'
AVERAGE='AVERAGE'
BYTE='BYTE'
CENTER='CENTER'
CHANNEL1='CHANNEL1'
CHANNEL2='CHANNEL2'
CHANNEL3='CHANNEL3'
CHANNEL4='CHANNEL4'
DC='DC'
DELAYED='DELAYED'
EXTERNAL='EXTERNAL'
FIELD1='FIELD1'
FIELD2='FIELD2'
FIFTY='FIFTY'
GND='GND'
HF='HF'
```

```
HIGH='HIGH'
LEFT='LEFT'
LF='LF'
LINE='LINE'
LOW='LOW'
LSBFIRST='LSBFIRST'
MANUAL='MANUAL'
MSBFIRST='MSBFIRST'
NEGATIVE='NEGATIVE'
NORMAL='NORMAL'
OFF='OFF'
ON='ON'
ONEMEG='ONEMEG'
PEAK='PEAK'
PLUS='PLUS'
PMEMORY1='PMEMORY1'
PMEMORY2='PMEMORY2'
POSITIVE='POSITIVE'
ROLL='ROLL'
SINGLE='SINGLE'
SUBSTRACT='SUBSTRACT'
TV='TV'
VERTICAL='VERTICAL'
WORD='WORD'
XY='XY'
X1='X1'
X10='X10'
X100='X100'

[COMMANDS]
; Common Commands
CLS=\*CLS
ESE=\*ESE *
OPC=\*OPC
RCL=\*RCL *
RST=\*RST
SAV=\*SAV *
SRE=\*SRE *
TRG=\*TRG
WAI=\*WAI
GETESE=\*ESE\?
GETESE_RESPONSE=*
GETESR=\*ESR\?
GETESR_RESPONSE=*
GETIDN=\*IDN\?
GETIDN_RESPONSE=*
GETLRN=\*LRN\?
GETLRN_RESPONSE=##800000121*
GETOPC=\*OPC\?
GETOPC_RESPONSE=*
GETOPT=\*OPT\?
```

```
GETOPT_RESPONSE=*,*
GETSRE=\*SRE\?
GETSRE_RESPONSE=*
GETSTB=\*STB\?
GETSTB_RESPONSE=*
GETTST=\*TST\?
GETTST_RESPONSE=*

; Root Level Commands
ASTORE=:ASTORE
AUTOSCALE=:AUTOSCALE
BLANK=:BLANK *
DIGITIZE=:DIGITIZE *
DITHER=:DITHER *
ERASE=:ERASE *
MENU=:MENU *
MERGE=:MERGE *
PRINT=:PRINT\?
RUN=:RUN
STOP=:STOP
VIEW=:VIEW *
GETDITHER=:DITHER\?
GETDITHER_RESPONSE=*
GETMENU=:MENU\?
GETMENU_RESPONSE=*
GETSTATUS=:STATUS\? *
GETSTATUS_RESPONSE=*
GETTER=:TER\?
GETTER_RESPONSE=*

; System Commands
SYSDSP=:SYSTEM:DSP "*"
SYSKEY=:SYSTEM:KEY *
SYSLOCK=:SYSTEM:LOCK *
SYSSETUP=:SYSTEM:SETUP ##800000121*
SYSGETERROR=:SYSTEM:ERROR\?
SYSGETERROR_RESPONSE=*
SYSGETKEY=:SYSTEM:KEY\?
SYSGETKEY_RESPONSE=*
SYSGETLOCK=:SYSTEM:LOCK\?
SYSGETLOCK_RESPONSE=*
SYSGETSETUP=:SYSTEM:SETUP\?
SYSGETSETUP_RESPONSE=##800000121*

; Acquire Commands
ACQCOMPLETE=:ACQUIRE:COMPLETE *
ACQCOUNT=:ACQUIRE:COUNT *
ACQTYPE=:ACQUIRE:TYPE *
ACQGETCOMPLETE=:ACQUIRE:COMPLETE\?
ACQGETCOMPLETE_RESPONSE=*
ACQGETCOUNT=:ACQUIRE:COUNT\?
```

```
ACQGETCOUNT_RESPONSE=*
ACQGETPOINTS=:ACQUIRE:POINTS\?
ACQGETPOINTS_RESPONSE=*
ACQGETSETUP=:ACQUIRE:SETUP\?
ACQGETSETUP_RESPONSE=ACQ:TYPE*;COUNT*;POINTS*;COMP*
ACQGETTYPE=:ACQUIRE:TYPE\?
ACQGETTYPE_RESPONSE=*

; Channel Commands
CHABWLIMIT=:*:BWLIMIT *
CHACOUPLING=:*:COUPLING *
CHAINPUT=:*:INPUT *
CHAINVERT=:*:INVERT *
CHAMATH=:*:MATH *
CHAOFFSET=:*:OFFSET *
CHAPMODE=:*:PMODE *
CHAPROBE=:*:PROBE *
CHAPROTECT=:*:PROTECT *
CHARANGE=:*:RANGE *
CHASKEW=:CHANNEL2:SKEW *
CHAVERNIER=:*:VERNIER *
CHAGETBWLIMIT=:*:BWLIMIT\?
CHAGETBWLIMIT_RESPONSE=*
CHAGETCOUPLING=:*:COUPLING\?
CHAGETCOUPLING_RESPONSE=*
CHAGETINPUT=:*:INPUT\?
CHAGETINPUT_RESPONSE=*
CHAGETINVERT=:*:INVERT\?
CHAGETINVERT_RESPONSE=*
CHAGETMATH=:*:MATH\?
CHAGETMATH_RESPONSE=*
CHAGETOFFSET=:*:OFFSET\?
CHAGETOFFSET_RESPONSE=*
CHAGETPMODE=:*:PMODE\?
CHAGETPMODE_RESPONSE=*
CHAGETPROBE=:*:PROBE\?
CHAGETPROBE_RESPONSE=*
CHAGETPROTECT=:*:PROTECT\?
CHAGETPROTECT_RESPONSE=*
CHAGETRANGE=:*:RANGE\?
CHAGETRANGE_RESPONSE=*
CHAGETSETUP=:*:SETUP\?
CHAGETSETUP_RESPONSE=CHAN*:RANGE*;OFFSET*;COUP*;BWLIMIT*;INVER
T*;VERNIER*;PROBE*
CHAGETSKEW=:CHANNEL2:SKEW\?
CHAGETSKEW_RESPONSE=*
CHAGETVERNIER=:*:VERNIER\?
CHAGETVERNIER_RESPONSE=*

; Display Commands
DISCOLUMN=:DISPLAY:COLUMN *
```

```
DISDATA=:DISPLAY:DATA *
DISGRID=:DISPLAY:GRID *
DISINVERSE=:DISPLAY:INVERSE *
DISLINE=:DISPLAY:LINE "*"
DISPIXEL=:DISPLAY:PIXEL *,*,*
DISROW=:DISPLAY:ROW *
DISSOURCE=:DISPLAY:SOURCE *
DISTEXT=:DISPLAY:TEXT BLANK
DISGETCOLUMN=:DISPLAY:COLUMN\?
DISGETCOLUMN_RESPONSE=*
DISGETDATA=:DISPLAY:DATA\?
DISGETDATA_RESPONSE=##800016256*
DISGETGRID=:DISPLAY:GRID\?
DISGETGRID_RESPONSE=*
DISGETINVERSE=:DISPLAY:INVERSE\?
DISGETINVERSE_RESPONSE=*
DISGETPIXEL=:DISPLAY:PIXEL\?*,*
DISGETPIXEL_RESPONSE=*
DISGETROW=:DISPLAY:ROW\?
DISGETROW_RESPONSE=*
DISGETSETUP=:DISPLAY:SETUP\?
DISGETSETUP_RESPONSE=DISP:ROW*;COL*;INVERSE*;GRID*;SOURCE*;PME
M*
DISGETSOURCE=:DISPLAY:SOURCE\?
DISGETSOURCE_RESPONSE=*

; Measure Commands
MEADUTYCYCLE=:MEASURE:DUTYCYCLE
MEAFALLTIME=:MEASURE:FALLTIME
MEAFREQUENCY=:MEASURE:FREQUENCY
MEANWIDTH=:MEASURE:NWIDTH
MEAPERIOD=:MEASURE:PERIOD
MEAPWIDTH=:MEASURE:PWIDTH
MEARISETIME=:MEASURE:RISETIME
MEASCRATCH=:MEASURE:SCRATCH
MEASHOW=:MEASURE:SHOW *
MEASOURCE=:MEASURE:SOURCE *
MEATSTART=:MEASURE:TSTART *
MEATSTOP=:MEASURE:TSTOP *
MEAVAVERAGE=:MEASURE:VAVERAGE
MEAVBASE=:MEASURE:VBASE
MEAVMAX=:MEASURE:VMAX
MEAVMIN=:MEASURE:VMIN
MEAVPP=:MEASURE:VPP
MEAVRMS=:MEASURE:VRMS
MEAVSTART=:MEASURE:VSTART *
MEAVSTOP=:MEASURE:VSTOP *
MEAVTOP=:MEASURE:VTOP
MEAGETALL=:MEASURE:ALL\?
MEAGETALL_RESPONSE=*
MEAGETDUTYCYCLE=:MEASURE:DUTYCYCLE\?
```

```
MEAGETDUTYCYCLE_RESPONSE=*
MEAGETFALLTIME=:MEASURE:FALLTIME\?
MEAGETFALLTIME_RESPONSE=*
MEAGETFREQUENCY=:MEASURE:FREQUENCY\?
MEAGETFREQUENCY_RESPONSE=*
MEAGETNWIDTH=:MEASURE:NWIDTH\?
MEAGETNWIDTH_RESPONSE=*
MEAGETPERIOD=:MEASURE:PERIOD\?
MEAGETPERIOD_RESPONSE=*
MEAGETPWIDTH=:MEASURE:PWIDTH\?
MEAGETPWIDTH_RESPONSE=*
MEAGETRISETIME=:MEASURE:RISETIME\?
MEAGETRISETIME_RESPONSE=*
MEAGETSHOW=:MEASURE:SHOW\?
MEAGETSHOW_RESPONSE=*
MEAGETSOURCE=:MEASURE:SOURCE\?
MEAGETSOURCE_RESPONSE=*
MEAGETTDELTA=:MEASURE:TDELTA\?
MEAGETTDELAT_RESPONSE=*
MEAGETTSTART=:MEASURE:TSTART\?
MEAGETTSTART_RESPONSE=*
MEAGETTSTOP=:MEASURE:TSTOP\?
MEAGETTSTOP_RESPONSE=*
MEAGETTVOLT=:MEASURE:TVOLT\? *,*
MEAGETTVOLT_RESPONSE=*
MEAGETVAVERAGE=:MEASURE:VAVERAGE\?
MEAGETVAVERAGE_RESPONSE=*
MEAGETVBASE=:MEASURE:VBASE\?
MEAGETVBASE_RESPONSE=*
MEAGETVDELTA=:MEASURE:VDELTA\?
MEAGETVDELTA_RESPONSE=*
MEAGETVMAX=:MEASURE:VMAX\?
MEAGETVMAX_RESPONSE=*
MEAGETVMIN=:MEASURE:VMIN\?
MEAGETVMIN_RESPONSE=*
MEAGETVPP=:MEASURE:VPP\?
MEAGETVPP_RESPONSE=*
MEAGETVRMS=:MEASURE:VRMS\?
MEAGETRMS_RESPONSE=*
MEAGETVSTART=:MEASURE:VSTART\?
MEAGETVSTART_RESPONSE=*
MEAGETVSTOP=:MEASURE:VSTOP\?
MEAGETVSTOP_RESPONSE=*
MEAGETVTIME=:MEASURE:VTIME\? *
MEAGETVTIME_RESPONSE=*
MEAGETVTOP=:MEASURE:VTOP\?
MEAGETVTOP_RESPONSE=*

; Time base Commands
TIMDELAY=:TIMEBASE:DELAY *
TIMMODE=:TIMEBASE:MODE *
```

```
TIMRANGE=:TIMEBASE:RANGE *
TIMREFERENCE=:TIMEBASE:REFERENCE *
TIMVERNIER=:TIMEBASE:VERNIER *
TIMGETDELAY=:TIMEBASE:DELAY\?
TIMGETDELAY_RESPONSE=*
TIMGETMODE=:TIMEBASE:MODE\?
TIMGETMODE_RESPONSE=*
TIMGETRANGE=:TIMEBASE:RANGE\?
TIMGETRANGE_RESPONSE=*
TIMGETREFERENCE=:TIMEBASE:REFERENCE\?
TIMGETREFERENCE_RESPONSE=*
TIMGETSETUP=:TIMEBASE:SETUP\?
TIMGETSETUP_RESPONSE=TIMEBASE:MODE*;RANGE*;DELAY*;REF*;VERN*
TIMGETVERNIER=:TIMEBASE:VERNIER\?
TIMGETVERNIER_RESPONSE=*

; Trigger Commands
TRICOUPLING=:TRIGGER:COUPLING *
TRIHOLDOFF=:TRIGGER:HOLDOFF *
TRILEVEL=:TRIGGER:LEVEL *
TRIMODE=:TRIGGER:MODE *
TRINREJECT=:TRIGGER:NREJECT *
TRIPOLARITY=:TRIGGER:POLARIY *
TRIREJECT=:TRIGGER:REJECT *
TRISLOPE=:TRIGGER:SLOPE *
TRISOURCE=:TRIGGER:SOURCE *
TRITVHFREJ=:TRIGGER:TVHFREJ *
TRITVMODE=:TRIGGER:TVMODE *
TRIGETCOUPLING=:TRIGGER:COUPLING\?
TRIGETCOUPLING_RESPONSE=*
TRIGETHOLDOFF=:TRIGGER:HOLDOFF\?
TRIGETHOLDOFF_RESPONSE=*
TRIGETLEVEL=:TRIGGER:LEVEL\?
TRIGETLEVEL_RESPONSE=*
TRIGETMODE=:TRIGGER:MODE\?
TRIGETMODE_RESPONSE=*
TRIGETNREJECT=:TRIGGER:NREJECT\?
TRIGETNREJECT_RESPONSE=*
TRIGETPOLARITY=:TRIGGER:POLARITY\?
TRIGETPOLARITY_RESPONSE=*
TRIGETREJECT=:TRIGGER:REJECT\?
TRIGETREJECT_RESPONSE=*
TRIGETSETUP=:TRIGGER:SETUP\?
TRIGETSETUP_RESPONSE=TRIG:MODE*;SOURCE*;LEVEL*;HOLD*;SLOPE*;CO
UP*;REJ*;NREJ*;TVMODE*;TVHF*
TRIGETSLOPE=:TRIGGER:SLOPE\?
TRIGETSLOPE_RESPONSE=*
TRIGETSOURCE=:TRIGGER:SOURCE\?
TRIGETSOURCE_RESPONSE=*
TRIGETTVHFREJ=:TRIGGER:TVHFREJ\?
TRIGETTVHFREJ_RESPONSE=*
```

```
TRIGETTVMODE=:TRIGGER:TVMODE\?
TRIGETTVMODE_RESPONSE=*

; Waveform Commands
WAVBYTEORDER=:WAVEFORM:BYTEORDER *
WAVFORMAT=:WAVEFORM:FORMAT *
WAVPOINTS=:WAVEFORM:POINTS *
WAVSOURCE=:WAVEFORM:SOURCE *
WAVGETBYTEORDER=:WAVEFORM:BYTEORDER\?
WAVGETBYTEORDER_RESPONSE=*
WAVGETDATA=:WAVEFORM:DATA\?
WAVGETDATA_RESPONSE=##8!????????[*,]
WAVGETFORMAT=:WAVEFORM:FORMAT\?
WAVGETFORMAT_RESPONSE=*
WAVGETPOINTS=:WAVEFORM:POINTS\?
WAVGETPOINTS_RESPONSE=*
WAVGETPREAMBLE=:WAVEFORM:PREAMBLE\?
WAVGETPREAMBLE_RESPONSE=*,*,*,*,*,*,*,*,*,*
WAVGETSOURCE=:WAVEFORM:SOURCE\?
WAVGETSOURCE_RESPONSE=*
WAVGETTYPE=:WAVEFORM:TYPE\?
WAVGETTYPE_RESPONSE=*
WAVGETXINC=:WAVEFORM:XINCREMENT\?
WAVGETXINC_RESPONSE=*
WAVGETXORG=:WAVEFORM:XORIGIN\?
WAVGETXORG_RESPONSE=*
WAVGETXREF=:WAVEFORM:XREFERENCE\?
WAVGETXREF_RESPONSE=*
WAVGETYINC=:WAVEFORM:YINCREMENT\?
WAVGETYINC_RESPONSE=*
WAVGETYORG=:WAVEFORM:YORIGIN\?
WAVGETYORG_RESPONSE=*
WAVGETYREF=:WAVEFORM:YREFERENCE\?
WAVGETYREF_RESPONSE=*
```

6.3.3 Konfigurationsdatei INIT.PHP für die Weboberfläche

Diese Datei stellt die Parameter für die Webseite ein.

```
<?php
define("WORKDIR", "./tmp");     // working directory for
                                // CMD-Files
define("INIDIR", ".");          // directory containing
                                // INI-Files
define("DIRSEP", "/");          // directory seperator
define("LOGFILE", "mm.log");    // current history file
define("PASSWORD", "*****");    // password for deleting
                                // results
?>
```

6.3.4 Konfigurationsdatei MM.CFG für das DOS-Programm

Die Datei MM.CFG stellt die Parameter für das DOS-Programm MM.EXE ein.

```
; mm.cfg Initialization file for Multi Measurements
[MM]
INIDIR=W:\HTTPD\html
CMDDIR=X:\
SERVERMODE=1
LOGFILE=mm.log

[ERROR]
LINE=MM: ?
```

6.3.5 Vordefinierte Befehlsdateien

Einige Befehlsdateien sind auf dem Webserver gespeichert und können in eigenen Dateien verwendet werden. Hier sollen zwei Beispiele den Ablauf einer Befehlsdatei verdeutlichen.

6.3.5.1 Include-Datei GAUSSAC.INC

```
[Gaussmeter]   ; Get Gauss AC
LOCKOUT ON
MODE GAUSSAC FILTERON
RANGE AUTORANGE
PEAK PH_OFF
WAIT 0
OUT DATA OPEN 'sec' 'G'
:MEASURE
MEASURE RANGE SIGN READING
SET A ROUND(T/100)
OUT DATA LINE A GRESULT
IF T > 6000 GOTO ENDE
WAIT 200
GOTO MEASURE
:ENDE
OUT DATA CLOSE
LOCKOUT OFF
```

Der Ablauf dieser Befehlsdatei soll im folgenden dargestellt werden. Auf der linken Seite stehen die einzelnen Zeilen der Befehlsdatei. Rechts werden die Aktionen beschrieben, die diese Zeilen auslösen.

Einträge in GAUSSAC.INC		**Zugriffe auf INI-Datei und Beschreibung**
[Gaussmeter]	→	Datei GAUSS.INI wird verwendet

LOCKOUT	→	[COMMANDS] ... LOCKOUT=LO? ...
	←	Ein Parameter (?) wird erwartet
ON	→	[CONSTANTS] ... ON=2 ...
		Befehl LO2 wird an das Meßgerät geschickt
		[COMMANDS] ... LOCKOUT_RESPONSE=LO%? ...
		Antwort wird überprüft, muß LO2 lauten
MODE	→	Befehl wird wie oben ausgeführt
...		
WAIT 0	→	Interner Befehl WAIT, mit Parameter 0 setzt Laufzeit auf 0 zurück
OUT DATA OPEN	→	Interner Befehl OUT entdeckt, mit Stream DATA und Action OPEN
		[STREAM:DATA] HEAD=xxxxx?xx?xxxxxxx
	←	2 Parameter (?) erwartet
'sec' 'G'		Werden eingefügt und in Datei geschrieben
MEASURE	→	ME100000000 wird an Meßgerät geschickt
		[COMMANDS] ... MEASURE_RESULT=ME10?|?|????? ...
	←	3 Parameter werden erwartet, um die zurückgelieferten Zeichen zu speichern
RANGE SIGN READING	→	Abschnitt [RANGE] existiert Zurückgelieferter Wert wird mittels angegebenen Umwandlungen transferiert
		[SIGN] DEFAULT=1 -=-1 Wenn für SIGN das Zeichen „-„ kommt, wird als Wert die Zahl –1 gespeichert
		READING wird direkt als die letzten 5 Zeichen der Antwort gespeichert
SET	→	Interner Befehl SET
	←	Parameter wird erwartet

A	→	[PARAMETERS] .. A=0 ..
	←	Formel oder Wert wird erwartet
ROUND(T/100)	→	Formelauswertung, ROUND ist Funktion, T ist Spezialparameter, ausgerechneter Wert wird in Parameter A gespeichert
OUT DATA LINE	→	Interner Befehl OUT mit Stream DATA und Action LINE
		[STREAM:DATA] LINE=xxxxx?xxxxxxxxxxxx?
	←	Es werden 2 Parameter (?) erwartet
A GRESULT	→	A ist Zahl und wird eingefügt
		GRESULT ist Formel und wird ausgewertet
		[CONSTANTS] ... GRESULT=RANGE * SIGN * READING ...
IF T > 6000 GOTO ENDE	→	Wenn Laufzeit größer als 60 Sekunden, wird zur Sprungmarke :ENDE verzweigt
WAIT 200	→	Interner Befehl WAIT wartet bis 2 Sekunden seit dem letzten Aufruf verstrichen sind.
GOTO MEASURE	→	Springt zur Sprungmarke :MEASURE
OUT DATA CLOSE	→	Interner Befehl OUT mit Stream DATA und Action CLOSE
		[STREAM:DATA] TAIL=xxxxxxxx
		Schließt die Ausgabedatei
LOCKOUT OFF	→	Schickt Befehl ans Meßgerät

Tabelle 4: Befehlsablauf

6.3.5.2 Include-Datei WAVE1.INC

```
[Oscilloscope] ; Get Channel1
OUT DATA OPEN
WAVSOURCE CHANNEL1
WAVFORMAT ASCII
WAVPOINTS 400
WAVGETPREAMBLE FORMAT TYPE POINTS COUNT XINC XORG XREF YINC
YORG YREF
OUT INFO LINE 'V/DIV' V/DIV 'V'
OUT INFO LINE 'Offset' OFFSET 'V'
OUT INFO LINE 'S/DIV' S/DIV 'S'
```

```
OUT INFO LINE 'Delay' DELAY 'S'
WAVGETDATA DATA
OUT DATA CLOSE
```

Einträge in WAVE1.INC		**Zugriffe auf INI-Datei und Beschreibung**
[Oscilloscope]	→	Datei OSCI.INI wird verwendet
OUT DATA OPEN	→	Interner Befehl OUT mit Stream DATA und Action OPEN
		[STREAM:DATA] HEAD=xxxxxxxxxxxxxxx
		Kein Parameter erwartet, Zeile wird direkt in Datei geschrieben
WAVSOURCE	→	[COMMANDS] ... WAVSOURCE=:WAVFORM:SOURCE * ...
	←	Ein Parameter wird erwartet
CHANNEL1	→	[CONSTANTS] ... CHANNEL1='CHANNEL1' ...
		Befehl :WAVEFORM:SOURCE CHANNEL1 wird an Meßgerät geschickt CHANNEL1_RESPONSE existiert nicht, deshalb wird auf keine Antwort gewartet
...		
WAVGETPREAMBLE	→	[COMMANDS] ... WAVGETPREAMBLE=:WAVEFORM:PREAMBLE\? ...
		Befehl wird ans Meßgerät geschickt
		[COMMANDS] ... WAVGETPREAMBLE_RESPONSE=*,*,*,*,*,*,*,*,*,*
	←	10 Parameter mit beliebiger Länge durch Beistriche getrennt werden zurückgeliefert
FORMAT TYPE POINTS COUNT XINC XORG XREF YINC	→	[PARAMETERS] FORMAT=0 ...
		Parameter werden gespeichert
OUT INFO LINE	→	Interner Befehl OUT mit Stream INFO

		und Action LINE
		[STREAM:INFO] LINE='; ? = ? ?'
		3 Parameter werden erwartet
'V/DIV'	→	Zeichenkette wird direkt übernommen
V/DIV	→	[CONSTANTS] V/DIV=32 * YINC
		Formel wird ausgewertet
'V'	→	Zeichenkette wird direkt übernommen
...		
WAVGETDATA	→	[COMMANDS] WAVGETDATA=:WAVEFORM:DATA\?
		Befehl wird ans Meßgerät geschickt
		[COMMANDS] WAVGETDATA_RESPONSE=##8!?????? ??[*,]
		Die Antwort beginnt mit der Zeichenkette ‚#8', danach folgen 8 Zeichen, die ignoriert werden sollen.
	←	Das Zeichen „[„ leitet einen Stream ein
DATA	→	[STREAM:DATA] LINE=?,?,?,? PARAMS=#COUNT #VALUE TIME VOLTAGE
		Der Stream erwartet 4 Parameter, diese werden durch PARAMS angegeben
		#COUNT steht für die aktuelle Nummer des empfangenen Zeichens
		#VALUE steht für den Wert des aktuell empfangenen Zeichens
		[CONSTANTS] TIME=(#COUNT-XREF)*XINC+XORG VOLTAGE=(#VALUE-YREF)*YINC+YORG
OUT DATA CLOSE	→	Ausgabedatei wird geschlossen

Tabelle 5: Befehlsablauf

7. Ausblick

Das realisierte System erlaubt die Bedienung eines einzelnen Meßgerätes auf DOS-Ebene mittels direkter Befehlseingabe ebenso wie das Verwenden von mehreren zusammenarbeitenden Meßgeräten und anderen Steuergeräten. Auf schnelle Weise können einfache Befehle direkt an ein Meßgerät geschickt werden. Ebenso ist der Einsatz als ferngesteuertes Labor denkbar. Man kann Spannungsquellen und Verbraucher zusammenschließen, die dann über die Webseite der Reihe nach aktiviert werden, womit ein komplexes Verhalten realisiert werden kann.

8. Verwendete Hilfsmittel

Programme:

Borland PASCAL Version 7.0, Copyright © 1983,92 Borland International, Inc.

Asyncronous Serial Communications Package, © 1989 Rising Edge Data Services

PHP (PHP: Hypertext Preprocessor) Version 3.0.17 von http://www.php.net/

Dokumentation:

PHP Online Manual und Diskussionsforum

PC Intern 3.0 Systemprogrammierung, Michael Tischer, 1992 DATA BECKER

DOS 5 für Programmierer, Arne Schäpers, 1991 Addison-Wesley

Microsoft MSDN Win32 API: Platform SDK

Wilbur HTML 3.2 von http://www.htmlhelp.com/reference/wilbur/

SelfHTML 7.0 von http://www.teamone.de/selfaktuell/

Client-Side JavaScript Reference von
http://developer.netscape.com/docs/manuals/javascript.html

Webbrowser:

Netscape Navigator 4.7

Microsoft Internet Explorer 5.0