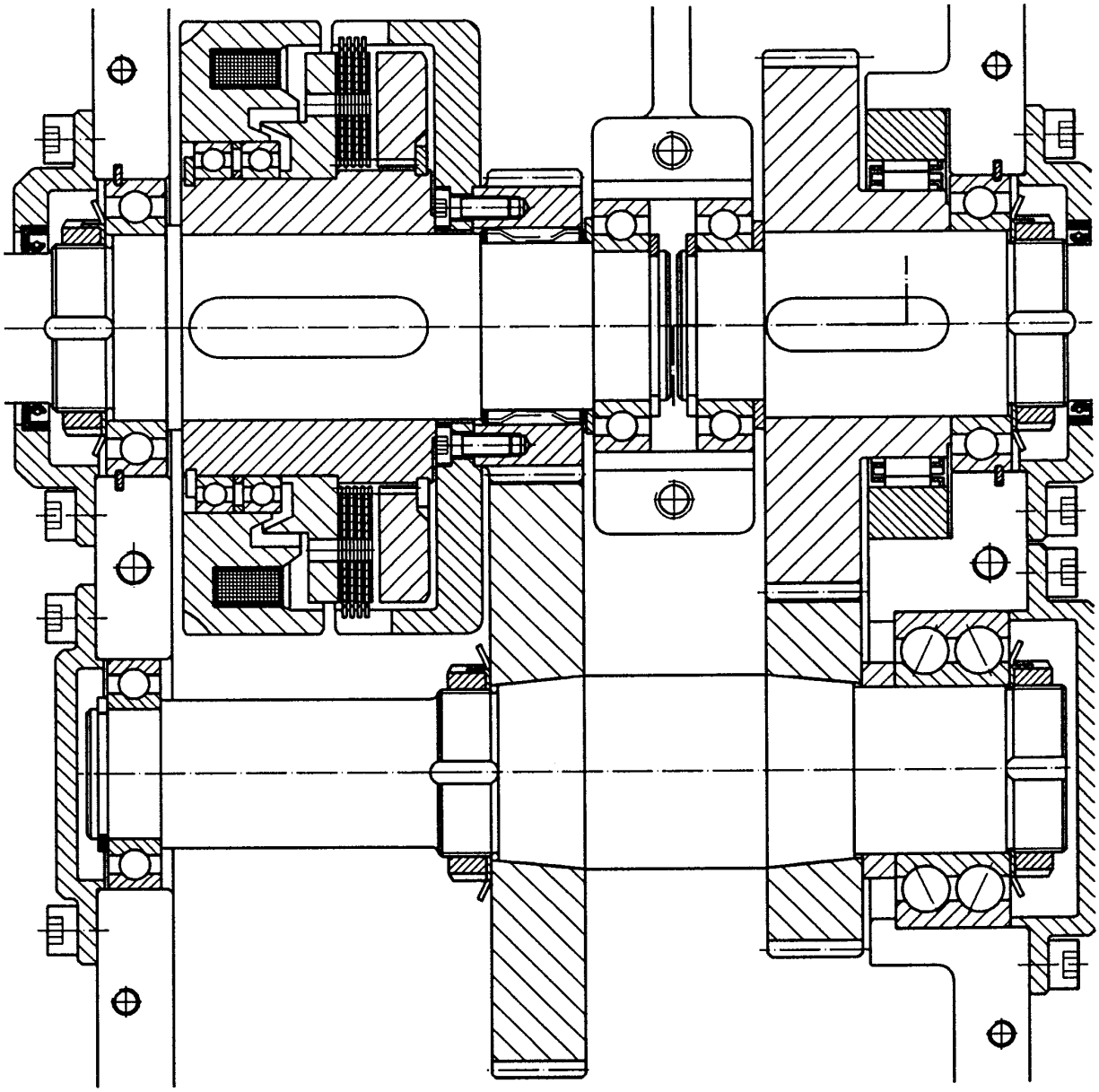
Dieter Fischer

Die Makrosprache des CAD-Programms HP-ME10



Die Makrosprache des CAD-Programms HP-ME10

Vorwort

Das vorliegende Skriptum ist lediglich zur Benutzung in Lehrveranstaltungen bestimmt. Für die Richtigkeit kann keine Gewähr übernommen werden. Eine Vervielfältigung oder Weitergabe ist nur mit Genehmigung des Autors erlaubt.

Kapitel 1 gibt einen Überblick über Makro- und Variantentechniken. Es werden darin grundsätzliche Begriffe erläutert. Kapitel 2 beschreibt das Arbeiten mit Makros, das Erstellen, Austesten, Verwalten von Makros sowie den Aufbau der Makrosprache. Kapitel 3 ist dem ME10-Texteditor gewidmet. Kapitel 4 zeigt den grundsätzlichen Aufbau von Makros. Insbesondere wird auf den später noch häufig verwendeten Begriff "Token" eingegangen. Kapitel 5 handelt von Variablen und deren Verwendung.

Dialoganweisungen, die eine Verbindung zwischen Makro und Benutzer herstellen, werden in Kapitel 6 beschrieben. Die Kapitel 7 und 8 behandeln Ausdrücke, Operatoren, Operanden und die in ME10 integrierten Funktionen. In Kapitel 9 findet ein kurzer, aber sehr nützlicher Ausflug in die Vektoralgebra statt. Kapitel 10 behandelt die Fallunterscheidung, die Schleifentechnik und die Unterprogrammtechnik. Das Arbeiten mit sequentiellen Dateien ist Thema von Kapitel 11. Kapitel 12 beschreibt, wie Anwendungsprogramme in Makros eingebunden werden können.

In Kapitel 13 werden Anweisungen zur Abfrage von Elementdaten, Systemeinstellungen und anderen interessanten Dingen beschrieben. Kapitel 14 behandelt die Systemanpassung, also das Anpassen der Systemumgebung, die Belegung der Funktionstasten und die Veränderung des Menüsystems. Logische Tabellen und Anzeigetabellen sind Themen des letzten Kapitels.

Anhang A1 enthält eine Zusammenstellung der wichtigsten ME10-Befehle und Interruptfunktionen, Anhang A2 eine Aufstellung von Fehlern, die erfahrungsgemäß besonders häufig gemacht werden. In Anhang A3 finden sich Übungsaufgaben, die sich in ihrer Numerierung an den zugehörigen Kapiteln orientieren. Aufgabe 10_1 gehört zum Beispiel zu Kapitel 10. Es wird dringend empfohlen, vor dem Lösen der Aufgaben die zugehörigen Kapitel durchzuarbeiten.

Zu Schluß noch eine Bitte: Das Skriptum ist sicher noch nicht perfekt. Es enthält bestimmt noch einige Schreibfehler und vielleicht auch sachliche Fehler. Wahrscheinlich ist es ist auch nicht in allen Punkten so verständlich, wie ich es mir vorstelle. Bitte notieren Sie sich Fehler oder Stellen, die Sie für unverständlich halten und informieren Sie mich darüber. Im CAD-Raum wird während des Praktikums ein Skriptum ausliegen, in dem Sie Mängel kennzeichnen können.

München, August 1997 Dieter Fischer

Inhaltsverzeichnis

1 Einführung ... 1- 1
1.1 CAD-Arbeitstechniken ... 1- 1
1.2 Variantenprogrammierung ... 1- 2

2 Allgemeines zu ME10-Makros ... 2- 1
2.1 Syntaxdiagramme ... 2- 1
2.1.1 Symbole ... 2- 1
2.1.2 Parameter |Datei| ... 2- 2
2.1.3 Parameter |Ausgabeziel| ... 2- 3
2.2 Arbeiten mit Makros ... 2- 4
2.3 Erstellen, Editieren, Speichern, Laden und Ausdrucken von Makros ... 2- 5
2.3.1 Erstellen und Editieren mit EDIT_FILE ... 2- 5
2.3.2 Erstellen und Editieren mit EDIT_MACRO ... 2- 7
2.3.3 Erstellen mit Hilfe einer ECHO-Datei ... 2- 7
2.3.4 Erstellen in der Dialogzeile ... 2- 8
2.3.5 Speichern ... 2- 9
2.3.6 Laden (INPUT, LOAD_MACRO) ... 2-10
2.3.7 Ausdrucken ... 2-11
2.4 Verwalten von Makros ... 2-12
2.4.1 Auflisten (LIST_MACRO_NAMES) ... 2-12
2.4.2 Sperren (SECURE_MACRO) ... 2-12
2.4.3 Löschen (DELETE_MACRO) ... 2-12
2.5 Formaler Sprachaufbau ... 2-13
2.6 Namen in Makros ... 2-13
2.7 Datentypen ... 2-14
2.8 Kommentare ... 2-14
2.9 Fehlerbehandlung ... 2-15
2.9.1 Fehlerarten, Systemreaktionen und Hilfsmittel ... 2-15
2.9.2 TRACE ... 2-16
2.9.3 PROMPT_LIST ... 2-17
2.9.4 ON_ERROR ... 2-17
2.9.5 TRAP_ERROR ... 2-18
2.9.6 CHECK_ERROR ... 2-19
2.9.7 ERROR_STR ... 2-19
2.9.8 ERROR_LOG ... 2-20
2.10 Koordinatensysteme ... 2-20
2.11 Makros und Objektfang ... 2-22

3 Der ME10-Texteditor ... 3- 1

4 Aufbau eines Makros ... 4- 1
4.1 Vereinbarungen ... 4- 1
4.1.1 DEFINE, END_DEFINE ... 4- 1
4.1.2 PARAMETER ... 4- 1
4.1.3 LOCAL ... 4- 2
4.2 Anweisungen ... 4- 4
4.2.1 Anweisungsarten und Syntax ... 4- 4
4.2.2 Formaler Aufbau ... 4- 6
5 Zuweisungen an Variablen ... 5- 1
5.1 Allgemeines zur Verwendung von Variablen ... 5- 1

5.2 Zuweisung mit LET 5- 1
5.3 Weitere Formen der Zuweisung 5- 2

6 Dialoganweisungen 6- 1
6.1 READ 6- 1
6.2 DISPLAY, DISPLAY_NO_WAIT, WAIT 6- 6
6.3 ENTER 6- 7
6.4 BEEP, TONE 6- 8

7 Ausdrücke, Operatoren und Operanden 7- 1
7.1 Aufbau von Ausdrücken 7- 1
7.2 Operatoren 7- 1
7.2.1 Arithmetische Operatoren 7- 1
7.2.2 Boolsche Operatoren 7- 2
7.2.3 Vergleichsoperatoren 7- 3
7.3 Operanden 7- 3
7.3.1 Konstanten 7- 4
7.3.1.1 Stringkonstanten (Text-Konstanten) 7- 4
7.3.1.2 Zahlenkonstanten 7- 5
7.3.1.3 Punkt_2D-Konstanten (2D-Vektor-Konstanten) 7- 6
7.3.1.4 Punkt_3D-Konstanten (3D-Vektor-Konstanten) 7- 6
7.4 Variablen 7- 7
7.5 Funktionsaufrufe 7- 7
7.6 Reihenfolge bei der Auswertung von Ausdrücken 7- 8

8 Integrierte Funktionen 8- 1
8.1 Arithmetische Funktionen 8- 1
8.2 Trigonometrische Funktionen 8- 2
8.3 Vektorfunktionen 8- 3
8.4 Stringfunktionen 8- 4
8.5 Sonstige Integrierte Funktionen 8- 5

9 Verwendung von Vektoren 9- 1

10 Steueranweisungen 10- 1
10.1 IF 10- 1
10.2 LOOP 10- 3
10.3 REPEAT 10- 4
10.4 WHILE 10- 5
10.5 Untermakroaufruf 10- 6
10.6 Shell-Aufruf 10- 8

11 Dateioperationen 11- 1
11.1 Öffnen von Dateien 11- 1
11.1.1 Öffnen zum Lesen (OPEN_INFILE) 11- 2
11.1.2 Öffnen zum Schreiben (OPEN_OUTFILE) 11- 2
11.2 Schließen von Dateien (CLOSE_FILE) 11- 3
11.3 Lesen von Dateien (READ_FILE) 11- 3
11.4 Schreiben in Dateien (WRITE_FILE) 11- 4
11.5 Ausgabe von Dateien 11- 4
11.6 Dateiverwaltung 11- 5
11.7 Anwendungsbeispiel 11- 5

12 Einbinden von Anwendungsprogrammen 12- 1
12.1 Grundlagen 12- 1
12.1.1 Benannte Pipes 12- 1
12.1.2 Vorder- und Hintergrundprozesse 12- 3
12.1.3 Umadressierung 12- 3
12.2 Anwendungsbeispiel 12- 5

13 Abfrageanweisungen 13- 1
13.1 Abfrage von Elementdaten 13- 2
13.1.1 Elementdatenermittlung für ein einzelnes Element 13- 2
13.1.2 Elementdatenermittlung für mehrere Elemente 13- 5
13.2 Abfrage von Teiledaten 13- 5
13.3 Abfrage von Systemeinstellungen 13- 6
13.4 Abfrage von Information zu Bildschirmmenüs und Anzeigetabellen 13- 9
13.5 Abfrage von Information zu vorausgegangenen Systemaktionen 13-11

14 Systemanpassung 14- 1
14.1 Anpassen der Systemumgebung 14- 2
14.2 Belegung der Funktionstasten 14- 3
14.3 Anpassen des Menüsystems 14- 5
14.3.1 Anpassung des Tablettmenüs 14- 6
14.3.2 Anpassung des Bildschirmmenüs 14-10
14.3.3 Benutzerdialog über das Menüsystem 14-18

15 Tabellen 15- 1
15.1 Eigenschaften 15- 1
15.2 Logische Tabellen 15- 4
15.2.1 Anlegen 15- 4
15.2.2 Erweitern 15- 5
15.2.3 Vollständiges Löschen 15- 5
15.2.4 Löschen einzelner Zeilen 15- 5
15.2.5 Sichern 15- 6
15.2.6 Speichern 15- 6
15.2.7 Abfragen der Tabellengröße 15- 7
15.2.8 Schreiben 15- 8
15.2.9 Lesen 15- 9
15.2.10 Sortieren 15-12
15.2.11 Selektieren 15-13
15.2.12 Einblenden, Ausblenden 15-16
15.2.13 Scrollen 15-16
15.2.14 Farbe ändern 15-17
15.2.15 Hervorheben 15-19
15.2.16 Anwendungsbeispiel für Logische Tabellen 15-20
15.3 Anzeigetabellen 15-27
15.3.1 Anzeigetabellen und Bildschirmmenüs im Vergleich 15-27
15.3.2 Anlegen und Festlegen des Layouts 15-28
15.3.3 Verknüpfen mit einer logischen Tabelle 15-33
15.3.4 Belegen der Kopffelder 15-34
15.3.5 Belegung der Datenfelder 15-37
15.3.6 Löschen 15-44
15.3.7 Sichern 15-44
15.9.8 Speichern 15-45

15.3.9 Ausdrucken 15-45
15.3.10 Einblenden, Ausblenden 15-46
15.3.11 Verändern der Schrittweite beim Blättern 15-46
15.3.12 Verändern der Größe 15-47
15.3.13 Verschieben 15-47
15.3.13 Festlegen und Ändern des Menüstatus 15-48
15.4 Anwendungsbeispiele 15-49
15.4.1 Darstellung von Matrizenkoeffizienten 15-49
15.4.2 Hilfesystem 15-50

Anhang

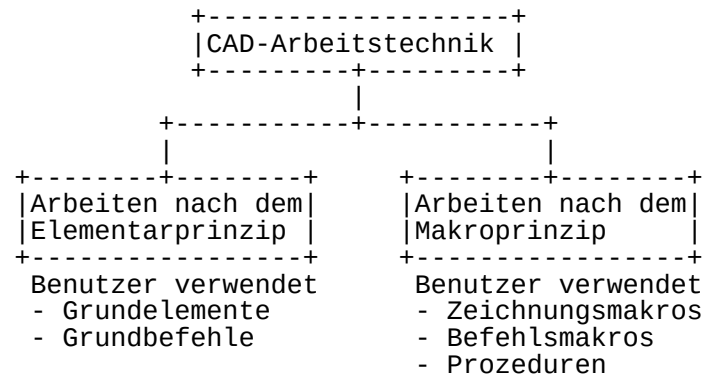
A1 Befehle und Interruptfunktionen
A2 Häufige Fehler
A3 Übungsaufgaben
A4 Musterlösungen
A5 Literatur
A6 Index (geplant)

1 Einführung

1.1 CAD-Arbeitstechniken

Manuell erstellte technische Zeichnungen erfordern meist umfangreiche, auf die Dauer lästige Wiederholungstätigkeiten. Da muß z.B. eine Schraube zehnmal gezeichnet werden (wer kann schon Schrauben richtig zeichnen?) oder Stücklisten sind in Normschriften auszufüllen (Sklavenarbeit!). Kein Wunder, wenn für diese Routinetätigkeiten schon sehr früh Hilfsmittel, wie z.B. Schablonen, transparente Folien oder ähnliches, entwickelt wurden.

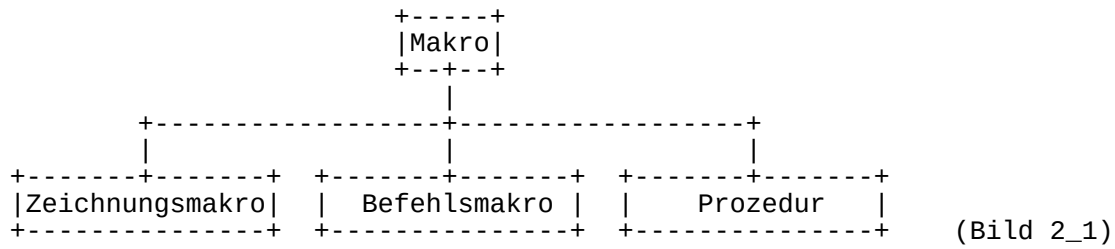
Hersteller von CAD-Systemen sind mit dem lobenswerten Vorsatz angetreten, Benutzern die Arbeit zu erleichtern, sie von Routinetätigkeiten zu befreien. Die gewonnene Zeit kann dann für sinnvollere Tätigkeiten genutzt werden, wie z.B. das Studium englischsprachiger Literatur (im allgemeinen als Manuals bezeichnet) oder für einen kleinen Plausch mit der Hotline. Die Arbeitstechnik, die das ermöglichen soll, kann als "Arbeiten nach dem Makroprinzip" bezeichnet werden. Arbeitstechniken des Rechnerunterstützten Konstruierens (CAD) lassen sich dann in "Arbeiten nach dem Elementarprinzip" und "Arbeiten nach dem Makroprinzip" einteilen (Bild 1_1).



(Bild 1_1)

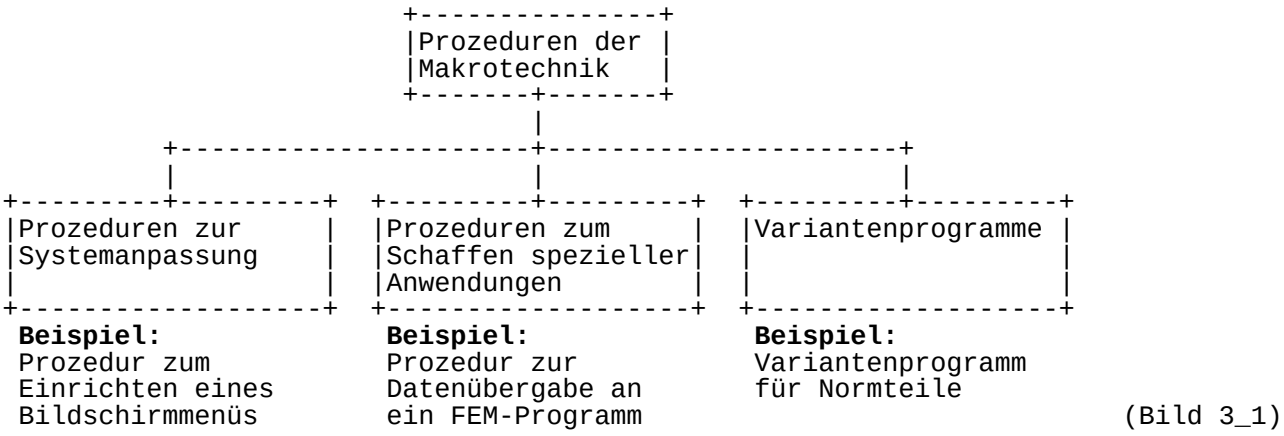
Arbeiten nach dem Elementarprinzip bedeutet, daß Zeichnungen aus den graphischen Grundelementen des CAD-Programms aufgebaut werden (z.B. aus Punkten, Strecken, Kreisbögen und Kreisen), und daß dabei nur die vorhandenen Grundbefehle Verwendung finden. Bestimmte Teile einer Zeichnung - und damit auch bestimmte Tätigkeiten des Benutzers - wiederholen sich jedoch häufig oder kommen in anderen Zeichnungen in ähnlicher Form wieder vor. In solchen Fällen bietet sich das Arbeiten nach dem Makroprinzip an.

Beim Arbeiten nach dem Makroprinzip werden bereits vorhandene Zeichnungen oder Teile davon (sogenannte Zeichnungsmakros), komplexe Befehle (sogenannte Befehlsmakros) oder Prozeduren (wird noch näher erläutert) verwendet. Ganz allgemein versteht man unter Makros also Zeichnungsmakros, Befehlsmakros und Prozeduren (Bild 2_1).



(Bild 2_1)

Prozeduren stellen die komplexeste Form der Makrotechnik dar. Je nach Verwendungszweck kann man sie einteilen in Prozeduren zur Systemanpassung, Prozeduren zum Schaffen spezieller Anwendungen und Variantenprogramme (Bild 3_1).



(Bild 3_1)

Variantenprogramme sind Thema des nächsten Unterkapitels.

1.2 Variantenprogrammierung

Für den Konstrukteur stellt sich oft die Aufgabe, bestehende Konstruktionen abzuwandeln, um sie veränderten Aufgaben anzupassen. So muß z.B. der Durchmesser einer Welle erhöht werden, damit ein größeres Drehmoment übertragen werden kann, oder die Lagersitze der Welle müssen einem anderen Lagertyp angepaßt werden. Auch bei der Entwicklung von Baureihen werden die einzelnen Glieder der Baureihe nicht immer wieder von Grund auf neu konstruiert, sondern aus einem Grundentwurf abgeleitet, indem Abmessungen verändert werden.

In allen Fällen entstehen sogenannte konstruktive Varianten (im folgenden kurz Varianten genannt), die sich vom Grundentwurf lediglich durch Abmessungen und Anzahl bestimmter Gestaltzonen unterscheiden.

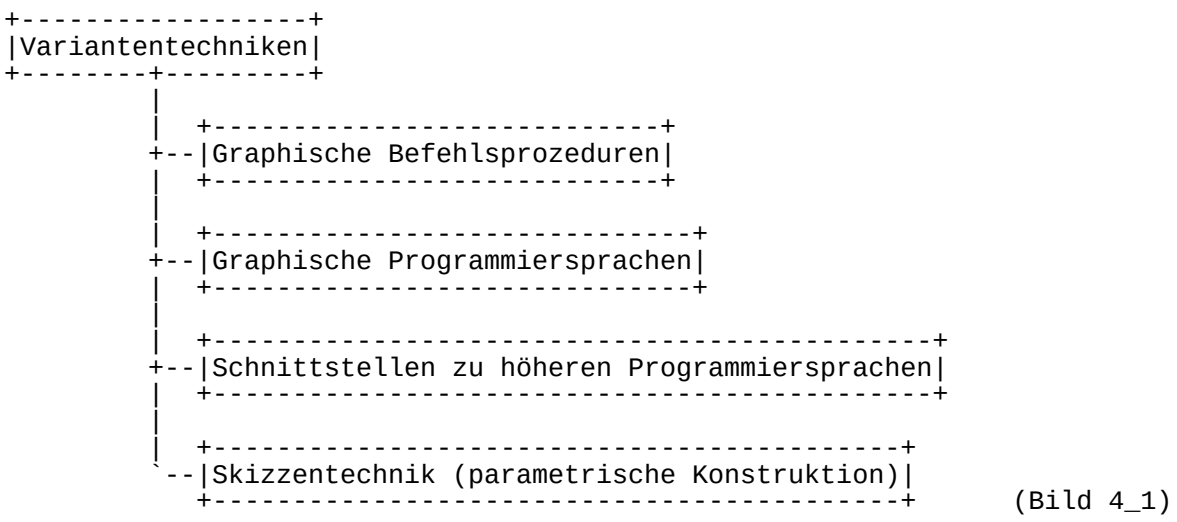
Eine Variante ist eine konstruktive Abwandlung eines Grundentwurfs. Die Abwandlung kann sowohl Abmessungen als auch Zahl und Anordnung einzelner Gestaltzonen betreffen.

Beim Rechnerunterstützten Konstruieren können Varianten auf zwei Weisen erzeugt werden:

- Durch Ändern vorhandener Objekte (z.B. Skalieren eines Zeichnungsmakros)
- Mit Hilfe der Variantentechnik (sofern das CAD-Programm über entsprechende Möglichkeiten verfügt)

Kennzeichnend für die Variantentechnik ist, daß der prinzipielle Aufbau der Konstruktion festliegt, und daß sich die spezielle Ausführung aus Daten bestimmt, die während der Laufzeit des Programms eingelesen werden.

Bekannte Variantentechniken (Bild 4_1, nach /5/):



(Bild 4_1)

Graphische Befehlsprozeduren können als erweiterte Befehlsketten bezeichnet werden. Ihre Sprachelemente entsprechen weitgehend dem Befehlsvorrat des CAD-Programms. Eine Einarbeitung ist in der Regel schnell möglich. Nachteilig ist meist ein im Vergleich zu höheren Programmiersprachen geringerer Leistungsumfang.

Graphische Programmiersprachen basieren meist auf höheren Programmiersprachen und sind um CAD-spezifische Sprachelemente erweitert. Sie sind sehr leistungsfähig. Der Benutzer muß allerdings eine spezielle Programmiersprache lernen.

Schnittstellen zu höheren Programmiersprachen. Es werden Programme in einer höheren Programmiersprache (z.B. FORTRAN oder C) geschrieben, die auf Funktionen des CAD-Programms zugreifen. Vorteilhaft sind eine sehr hohe Verarbeitungsgeschwindigkeit und die Möglichkeit, den gesamten Befehlsvorrat der höheren Programmiersprache zu nutzen. Nachteilig ist, daß der Anwender gute Kenntnisse in der höheren Programmiersprache besitzen muß.

Skizzentechnik (parametrische Konstruktion). Varianten werden durch Verändern von Maßangaben einer Mutterzeichnung erzeugt. Vorteilhaft ist die sehr einfache Anwendung. Nachteilig ist, daß bei einigen Systemen nur Maßvariationen möglich sind.

Neuerdings gewinnt die VDA-PS-Norm an Bedeutung. "VDA" steht hierbei für "Verband der deutschen Automobilindustrie", "PS" für "Programmschnittstelle". Nach dieser Norm können Variantenprogramme systemneutral in der Programmiersprache FORTRAN

erstellt und über entsprechende Schnittstellenprogramme in systemspezifische Variantenprogramme umgewandelt werden.

Das CAD-Programm HP-ME10 verfügt standardmäßig über eine Schnittstelle zur Programmiersprache "C" und über eine Makrosprache, die nach ihrem Aufbau eher den graphischen Befehlsprozeduren zuzuordnen ist. In ihrer Leistungsfähigkeit entspricht sie jedoch weitgehend einer graphischen Programmiersprache. Sie ermöglicht eine Programmierung von Befehlsmakros und Prozeduren. VDA-PS und Skizzentechnik sind als Erweiterungsmodule zu ME10 erhältlich.

Die nächsten Kapitel sollen einen einfachen, aber trotzdem gründlichen Einstieg in die Programmierung von ME10-Makros ermöglichen. Da ME10 voll im dreidimensionalen CAD-System HP-ME30 enthalten ist, lassen sich die Ausführungen auch gut für das Programmieren von ME30-Makros verwenden.

2 Allgemeines zu ME10-Makros

Dieses Kapitel soll darüber informieren, wie Makros entwickelt, ausgetestet und angewendet werden. Es enthält weiterhin Definitionen von Begriffen, die in zahlreichen nachfolgenden Kapiteln ohne nochmalige Erläuterung Verwendung finden.

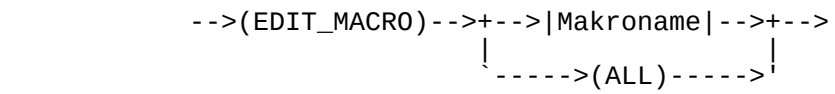
Zum vollen Verständnis aller Ausführungen müßten im Prinzip schon Kenntnisse vorhanden sein, die erst in späteren Kapiteln vermittelt werden. Hier ist eine pragmatische Vorgehensweise angebracht. Kapitel 2 sollte zunächst vorwiegend zur allgemeinen Information durchgelesen werden. Es ist nicht notwendig, jedes Detail zu verstehen. Wenn dann in späteren Kapiteln ein Querverweis auf Kapitel 2 erfolgt, ist die betreffende Passage erneut durchzulesen. Sie wird dann besser zu verstehen sein, da der Gesamtzusammenhang erkennbar ist. Auch bei Schwierigkeiten, die beim Schreiben von Makros auftreten, kann es nützlich sein, noch einmal kurz in Kapitel 2 zu blättern. Oft finden sich darin wertvolle Hinweise auf mögliche Fehler.

2.1 Syntaxdiagramme

2.1.1 Symbole

In den nachfolgenden Kapiteln werden Syntaxdiagramme verwendet. Die Syntax ist gewissermaßen die Grammatik der Makrosprache. Syntaxdiagramme zeigen in komprimierter Form alle Möglichkeiten auf, syntaktisch richtige Anweisungen zu bilden. Damit ist jedoch nichts über den Sinngehalt (die sogenannte Semantik) der Anweisungen ausgesagt. Es ist durchaus möglich, syntaktisch richtige und gleichzeitig semantisch falsche Anweisungen zu bilden. Ein analoges Beispiel ist die menschliche Sprache. Man kann völligen Unsinn grammatikalisch korrekt formulieren. Die in den Diagrammen enthaltenen Symbole haben folgende Bedeutung:

Zeichen in runden Klammern kennzeichnen **Schlüsselwörter** wie z.B. Namen für ME10-Befehle. Buchstaben sind hierbei zur besseren Unterscheidung groß geschrieben. Dies ist jedoch nicht unbedingt erforderlich, es können auch Kleinbuchstaben verwendet werden. Unzulässig ist hingegen eine gemischte Groß/Kleinschreibung. Im nachfolgenden Syntaxdiagramm handelt es sich bei "EDIT_MACRO" und "ALL" um Schlüsselwörter.



Die **in senkrechten Strichen** angegebenen Zeichenfolgen kennzeichnen Parameter. Sie werden im allgemeinen nicht in Großbuchstaben geschrieben. Parameter beschreiben die erforderlichen Eingaben wie z.B. |Zahl| oder |Datei|. Sie können sich auch auf andere Syntaxdiagramme beziehen. |Makroname| kennzeichnet also im obengenannten Beispiel den Namen des zu editierenden Makros.

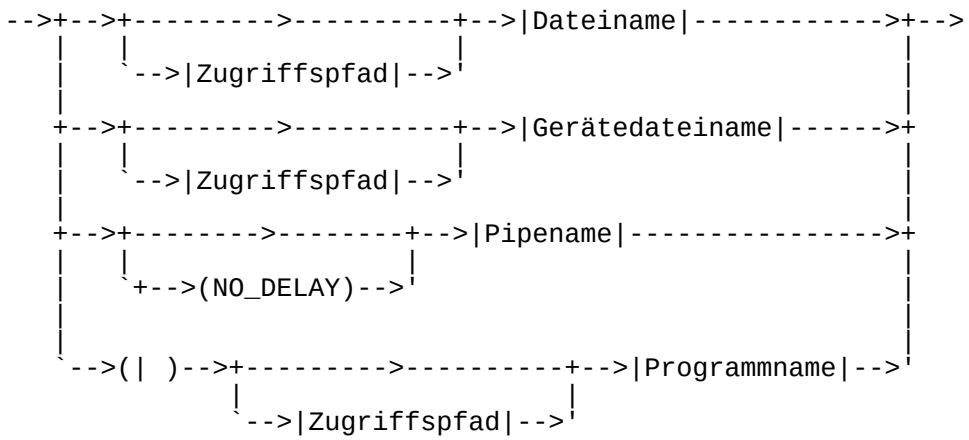
Linien und Pfeile kennzeichnen die Reihenfolge, in der die einzelnen Sprachelemente zu verwenden sind und geben an, ob und in welcher Weise Verzweigungen zu alternativen Sprachelementen oder Wiederholungen zulässig sind. Es wird allerdings nichts über die Anzahl der erlaubten Wiederholungen ausgesagt. Ein Diagrammzweig darf nur in

Pfeilrichtung und nicht entgegengesetzt durchlaufen werden. Beim vorgenannten Beispiel kann also entweder ein Makroname oder das Schlüsselwort "ALL" angegeben werden. Beide Angaben zusammen sind nicht zulässig.

2.1.2 Parameter |Datei|

Dieser Parameter kennzeichnet die Eingabequelle. Es kann sich dabei um eine normale Datei, um eine Gerätedatei, und in Sonderfällen auch um eine benannte Pipe oder um ein Betriebssystemprogramm handeln.

|Datei|



Bei normalen Dateien genügt die Angabe des Dateinamens, wenn sie sich im aktuellen Verzeichnis oder in dem mittels der Interruptfunktion PATH eingestellten Standard-Suchpfad befinden. Dateien in anderen Verzeichnissen müssen mit Zugriffspfad und Namen gekennzeichnet werden.

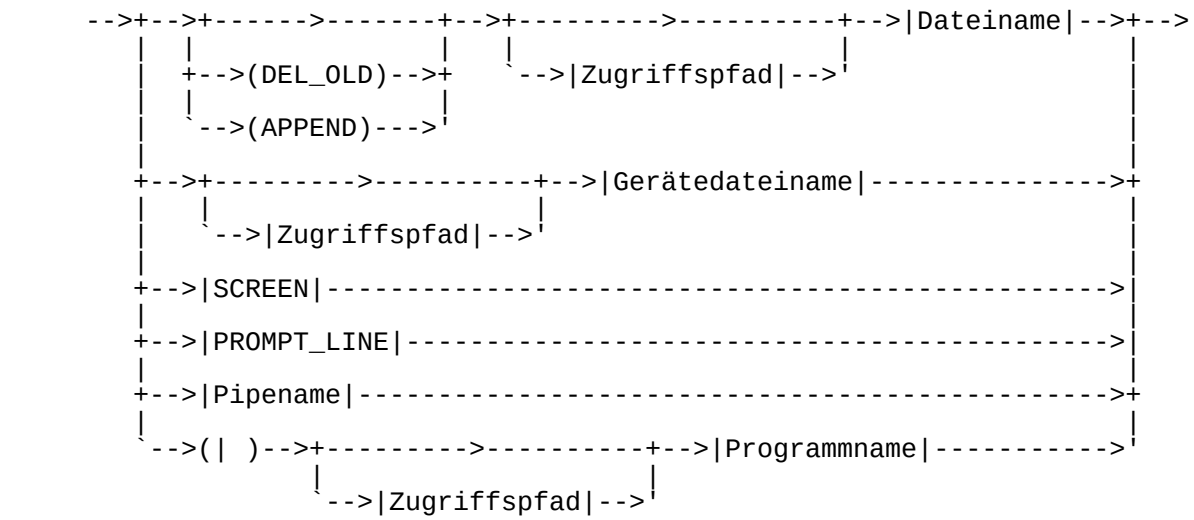
Gerätedateien sind normalerweise nicht im aktuellen Verzeichnis angelegt. Die Angabe des Zugriffspfades ist daher in der Regel erforderlich.

Benannte Pipes und Betriebssystemprogramme können nur bei HP-UX-Systemen als Eingabequellen angegeben werden. Auf diese Sonderfälle wird in separaten Kapiteln noch näher eingegangen. Den Betriebssystem-Programmnamen ist dabei immer das Pipe-Symbol "|" und ein Leerzeichen voranzustellen. Obwohl das von HP-UX bekannte Pipesymbol verwendet wird, sind an dieser Stelle nicht alle Operationen zulässig, die HP-UX erlaubt. Umgekehrt kann hier das Pipesymbol in einer Weise verwendet werden, die HP-UX nicht akzeptieren würde.

2.1.3 Parameter |Ausgabeziel|

Als Ausgabeziele können normale Dateien, Gerätedateien, der Textbildschirm, die Hinweiszeile, und in Sonderfällen auch benannte Pipes und Betriebssystemprogramme, angegeben werden.

|Ausgabeziel|



Bei normalen Dateien genügt die Angabe des Dateinamens, wenn sie im aktuellen Verzeichnis angelegt werden sollen. Dateien in anderen Verzeichnissen müssen mit Zugriffspfad und Namen gekennzeichnet werden. Die Optionen DEL_OLD und APPEND haben folgende Bedeutung:

Keine Option: Wenn die angegebene Datei existiert, wird eine Fehlermeldung angezeigt, ansonsten wird eine neue Datei erstellt.

DEL_OLD: Wenn die angegebene Datei bereits existiert, wird ihr alter Inhalt überschrieben.

APPEND: Wenn die angegebene Datei bereits existiert, wird die Information an das Ende der bestehenden Datei angefügt, ansonsten wird eine neue Datei erstellt.

Gerätedateien sind normalerweise nicht im aktuellen Verzeichnis angelegt. Die Angabe des Zugriffspfads ist daher in der Regel erforderlich.

Bei Angabe von |SCREEN| erfolgt die Ausgabe an den Bildschirmeditor (-> Kap.3). Sie kann danach mit allen Editorfunktionen bearbeitet werden.

|PROMPT_LINE| bestimmt die Hinweiszeile als Ausgabeziel. Die Ausgabe erfolgt zeilenweise. Das Betätigen einer beliebigen Taste oder des Tablettstifts schaltet eine Zeile weiter. Die Ausgabe wird mit der ESCAPE-Taste vorzeitig beendet.

Benannte Pipes und Betriebssystemprogramme können nur bei HP-UX-Systemen als Ausgabeziele angegeben werden. Auf diese Sonderfälle wird in separaten Kapiteln noch näher eingegangen. Einem Betriebssystem-Programmnamen ist dabei immer das Pipe-Symbol "|" und ein Leerzeichen voranzustellen. So kann z.B. ein Makro mit der Anweisung " SAVE_MACRO '| lp' " an den Drucker-Spooler des Betriebssystems geschickt werden. Obwohl das von HP-UX bekannte Pipesymbol verwendet wird, sind an dieser Stelle nicht

alle Operationen zulässig, die HP-UX erlaubt. Umgekehrt kann hier das Pipesymbol in einer Weise verwendet werden, die HP-UX nicht akzeptieren würde.

2.2 Arbeiten mit Makros

Die von ME10 verwendete Makrosprache ist eine Interpretersprache. Die Makros werden aus ME10 heraus aufgerufen und ihre Anweisungen werden vom Kommandointerpreter nacheinander übersetzt und ausgeführt. Während des Makrolaufs wird ME10 vom Makro gesteuert. Der Benutzer kann dann nur mittelbar über die vom Makro angeforderten Eingaben auf ME10 einwirken. Nach Beendigung des Makros wird die Steuerung wieder an den Benutzer übergeben. Ruft der Benutzer während eines Makrolaufs eine ME10-Interruptfunktion auf, wird diese ausgeführt und der Makrolauf fortgesetzt. Der Aufruf eines ME10-Befehls hingegen bricht den Makrolauf ab.

Der Start von bereits vorhandenen Makros geschieht in folgenden Schritten:

- Laden der den Makrotext enthaltenden Datei in den Speicher (-> Kap.2.3.6)
- Aufruf des Makros durch Eingabe seines Namens

Beispiel: Die Datei "normteil.mac" enthält den Text des Makros "Schraube". Zum Laden und Starten des Makros sind folgende Eingaben notwendig (Kommentare sind in geschweiften Klammen angegeben):

```
INPUT 'normteil.mac'      {Makrotext wird in Hauptspeicher geladen}
Schraube                  {Das Makro wird gestartet}
```

Der beim Laden anzugebende Dateiname muß in Hochkommas eingeschlossen werden. Die Datei kann auch die Texte mehrerer Makros enthalten. Da das Makro von ME10 wie ein Systembefehl behandelt wird, ist der Makroname beim Aufruf ohne Hochkommas einzugeben. Einem Makro können beim Aufruf Parameter übergeben werden. Dies ist jedoch nur dann zu empfehlen, wenn das Makro von einem anderen Makro aufgerufen wird. Die Parameterübergabe wird im Kapitel "Steueranweisungen" näher beschrieben.

Makros können im allgemeinen vorzeitig abgebrochen werden. Dies ist zum Beispiel notwendig, wenn sich ein Makro in einer Endlosschleife befindet. Wartet das Makro auf eine Benutzereingabe, so kann es durch Eingabe eines beliebigen ME10-Befehls (z.B. END oder LINE) oder durch Betätigen der ESCAPE-Taste abgebrochen werden. Wenn das Makro aber Aktionen ohne Benutzerdialog durchführt, haben ME10-Befehle oder die ESCAPE-Taste keine Wirkung. In diesem Fall kann ein BREAK-Signal das Makro beenden. BREAK-Signale werden bei HP-UX-Systemen mit der Tastenkombination <Shift>+<Pause> und bei MSDOS/WINDOWS-Systemen mit <Strg>+ <Pause> erzeugt.

Manchmal kann es sinnvoll sein, den Abbruch eines Makros durch ESCAPE oder BREAK zu unterbinden. ME10 erlaubt hierzu ein Abfangen von Abbruchsignalen.

IGNORE_BREAK (Interruptfunktion)

-->(IGNORE_BREAK)-->

ESCAPE und BREAK sind nach Aufruf dieser Funktion solange wirkungslos, bis sie mit Hilfe von ENABLE_BREAK wieder aktiviert werden oder bis ME10 erneut gestartet wird.

Nach Aufruf von IGNORE_BREAK kann mittels CHECK_BREAK geprüft werden, ob ESCAPE oder BREAK betätigt wurde.

CHECK_BREAK (integrierte Funktion)

-->(CHECK_BREAK)

CHECK_BREAK liefert den Wert 1, wenn ein Abbruchsignal nach Aufruf von IGNORE_BREAK erzeugt wurde und den Wert 0, wenn kein Abbruchsignal erzeugt wurde. Bei CHECK_BREAK handelt es sich um eine sogenannte integrierte Funktion, die eine Zahl als Ergebnis liefert. Die Zahl muß in einer Anweisung als Argument oder Parameter verwendet werden. Eine Anweisung, die nur aus einer integrierten Funktion besteht, ergibt keinen Sinn.

Da Abbruchsignale generell und nicht nur für ein bestimmtes Makro abgefangen werden, ist auf eine rechtzeitige Reaktivierung mittels ENABLE_BREAK zu achten.

ENABLE_BREAK (Interruptfunktion)

-->(ENABLE_BREAK)-->

2.3 Erstellen, Editieren, Speichern, Laden und Ausdrucken von Makros

Makrotexte können mit einem beliebigen ASCII-Texteditor geschrieben werden. ME10 bietet einen eigenen Texteditor an, der in Kapitel 3 beschrieben wird. Er wird mit EDIT_FILE oder EDIT_MACRO aufgerufen. ME10 bietet außerdem die Möglichkeit, Benutzereingaben in einer Datei zu protokollieren, um sie als Vorlage für Makros zu verwenden. Makros können wahlweise im ASCII- oder im Binärformat gespeichert werden. Das Ausdrucken von Makros ist im Prinzip ein Speichern, bei dem als Ausgabeziel der Drucker angegeben wird.

2.3.1 Erstellen und Editieren mit EDIT_FILE

EDIT_FILE wird beim Erstellen und Editieren (Verändern) von ASCII-Dateien (also nicht nur von Makros) verwendet. Dabei kommt der in Kapitel 3 beschriebene ME10-Texteditor zur Anwendung. Die Dateien dürfen nicht gesperrt sein (-> Kap.2.4.2) und müssen im ASCII-Format vorliegen.

EDIT_FILE (Interruptfunktion)

-->(EDIT_FILE)-->|Datei|-->

|Datei| kennzeichnet die zu bearbeitende Datei (-> Kap.2.1.2). Wenn die angegebene Datei bereits existiert, wird sie geöffnet und auf dem Textbildschirm dargestellt. Existiert noch keine Datei, wird eine neue Datei erstellt, und es erscheint der leere Textbildschirm.

Beispiel: Die im Verzeichnis "/users/user04/project1" befindliche Datei "platte.mac" soll editiert werden. Aktuelles Verzeichnis sei "user04". Es gibt mehrere Möglichkeiten:

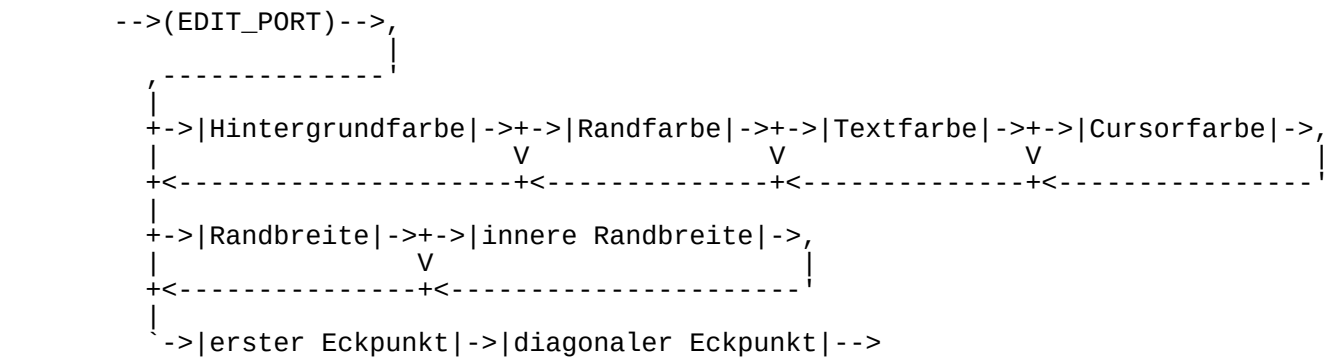
```
a)  EDIT_FILE 'projekt1/platte.mac'
b)  EDIT_FILE '/users/user04/project1/platte.mac'
c)  CURRENT_DIRECTORY 'project1'
    EDIT_FILE 'platte.mac'
```

Bei Möglichkeit c) wird zunächst Verzeichnis "project1" als aktuelles Verzeichnis eingestellt. Dieses Verfahren ist immer dann zu empfehlen, wenn mehrere in "project1" befindliche Dateien bearbeitet werden sollen.

Danach kann der Makrotext wie mit einem Textverarbeitungssystem eingegeben und mit Hilfe der in Kapitel 3 beschriebenen Editorbefehle bearbeitet werden. Die Datei kann den Text eines einzelnen oder auch mehrerer Makros enthalten. Bei gleichzeitigem Betätigen der Tasten <CTRL> und <D> wird der editierte Text unter dem beim Aufruf von EDIT_FILE angegebenen Dateinamen gespeichert. Ein Speichern unter einem anderen Namen ist mit Hilfe der Editorbefehle "$O" oder "$W" möglich (-> Kap.3).

Normalerweise wird nach Eingabe von EDIT_FILE auf den Textbildschirm umgeschaltet, und dem Benutzer steht die gesamte Bildschirmfläche für Editierzwecke zur Verfügung. Es besteht jedoch auch die Möglichkeit, mit EDIT_PORT ein Editierfenster im Grafikbildschirm einzurichten. Hierdurch lassen sich bei sehr umfangreichen Zeichnungsinhalten Rechenzeiten sparen, da der Grafikbildschirm nach Beendigung des Texteditors nicht wieder vollständig aufgebaut werden muß.

EDIT_PORT (Interruptfunktion)



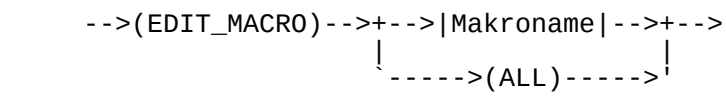
Die Eckpunkte des Fensters können mit Hilfe des Cursors bestimmt oder in Form von Pixelkoordinaten eingegeben werden. Wenn wieder der gesamte Bildschirm zum Editieren genutzt werden soll, müssen hier die für den jeweiligen Bildschirmtyp geltenden Werte der Pixelkoordinaten verwendet werden. Diese Werte sind normalerweise in den globalen Variablen "Graphic_area_low_left" und "Graphic_area_up_right" gespeichert. Die Anweisung könnte also wie folgt lauten:

```
EDIT_PORT Graphic_area_low_left Graphic_area_up_right
```

2.3.2 Erstellen und Editieren mit EDIT_MACRO

EDIT_MACRO wird speziell zum Erstellen und Editieren von Makros im Hauptspeicher verwendet und benutzt ebenfalls den ME10-Texteditor.

EDIT_MACRO (Interruptfunktion)



Der Makroname ist nicht in Hochkommas anzugeben. Befindet sich ein Makro mit dem angegebenen Namen im Hauptspeicher, wird es in den Editor geladen und auf dem Textbildschirm dargestellt. Existiert das Makro noch nicht, erscheint nur der leere Textbildschirm. Bei Angabe der Option ALL werden alle im Hauptspeicher befindlichen Makros in den Editor geladen.

Die Makros müssen im ASCII-Format vorliegen. Später wird gezeigt, daß es auch Makros im Binärformat gibt. Makros im Binärformat könnten zwar in den Editor geladen werden, sind aber für den Anwender nicht lesbar und damit nicht editierbar. Es besteht jedoch die Möglichkeit, sie mit SAVE_MACRO als ASCII-Dateien zu speichern, danach mit INPUT zu laden und nachfolgend zu editieren. Gesperrte Makros (-> SECURE_MACRO) können grundsätzlich nicht editiert werden, und es besteht auch keine Möglichkeit, die Sperre aufzuheben.

Der Makrotext kann wie mit einem Textverarbeitungsprogramm bearbeitet werden. Editorbefehle sind in Kapitel 3 beschrieben. Die Tastenkombination <CTRL>+<D> schließt den Texteditor ab. Da das Editieren im Hauptspeicher stattfindet, ist ein nachfolgendes Laden in den Hauptspeicher mit INPUT oder LOAD_MACRO nicht erforderlich. Es würde sogar zu unerwünschten Ergebnissen führen (-> Kap.2.3.6).

Ein auf diese Weise erstelltes oder verändertes Makro wäre beim Beenden von ME10 verloren, da es sich beim Hauptspeicher um einen sogenannten flüchtigen Speicher handelt. Es kann jedoch mit Hilfe der Editorbefehle "W" , "O" oder mit Hilfe von SAVE_MACRO bzw. STORE_MACRO dauerhaft auf einem Massenspeicher abgelegt werden.

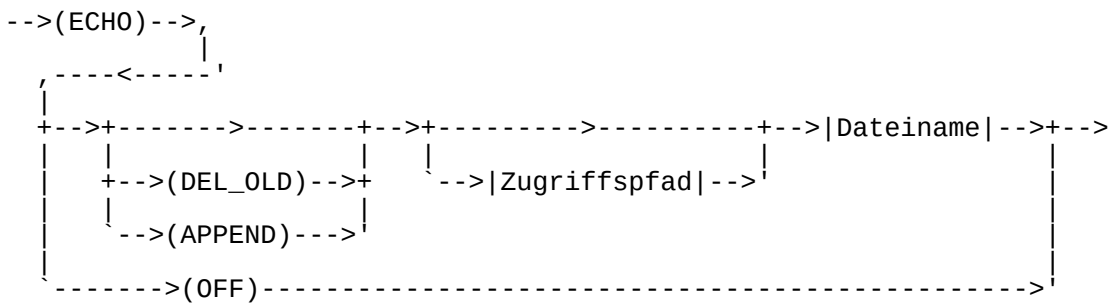
2.3.3 Erstellen mit Hilfe einer ECHO-Datei

ME10 ermöglicht die Protokollierung von Benutzereingaben in ECHO-Dateien, um Vorlagen für Makros zu erhalten. Es ist jedoch ein Irrtum, zu glauben, man müsse lediglich ein paar Dialoganweisungen hinzufügen und Konstanten durch Variablen ersetzten, um ein Makro zu erhalten.

Die notwendigen Änderungen sind vielmehr meist derart umfangreich, daß es in der Regel einfacher ist, ein Makro von Beginn an zu programmieren. Interaktives Arbeiten und die Vorgehensweise in Makros folgen nämlich unterschiedlichen Strategien. So werden z.B. Hilfslinientechnik und Änderungsfunktionen beim interaktiven Arbeiten sehr häufig verwendet, während in Makros davon kaum Gebrauch gemacht wird. Der Nutzen einer Echo-Datei dürfte für den Anfänger hauptsächlich darin liegen, zu erfahren, wie die ME10-Funktionen heißen, die nachher im Makro benötigt werden.

Die ECHO-Datei wird mit der Interruptfunktion ECHO geöffnet und geschlossen.

ECHO (Interruptfunktion)



Eine offene ECHO-Datei nimmt eine Kopie aller Benutzereingaben auf. Mit ECHO OFF wird die aktuelle ECHO-Datei geschlossen. Da aus Gründen der Eindeutigkeit immer nur eine ECHO-Datei offen sein kann, wird vor dem Öffnen eine andere geöffnete ECHO-Datei geschlossen.

Eine ECHO-Datei kann durch INPUT gelesen werden. Die darin enthaltenen ME10-Anweisungen werden beim Einlesen so ausgeführt, als wären sie vom Benutzer eingegeben. Das Einlesen einer Datei mit INPUT erbringt jedoch normalerweise nicht dasselbe Ergebnis wie die Benutzeraktionen, die zur ECHO-Datei geführt haben. Dies ist hauptsächlich darauf zurückzuführen, daß die Punkte in der Datei keinerlei Angaben über das Fenster beinhalten, in dem sie digitalisiert wurden. Da digitalisierte Punkte beim Fangen umgewandelt werden, die mit INPUT gelesenen Punkte jedoch standardmäßig nicht (ein Fangen kann jedoch auch hier erzwungen werden, -> Kap. 2.11), entstehen häufig voneinander abweichende Ergebnisse. Ein weiterer Grund ist die Tatsache, daß auch vom Benutzer verursachte Fehler in die ECHO-Datei geschrieben werden. Bei dem Lesen einer solchen ECHO-Datei mit INPUT wird der Lesevorgang aufgrund der Fehler abgebrochen.

2.3.4 Erstellen in der Dialogzeile

Prinzipiell können Makrotexte zeilenweise in die Dialogzeile eingegeben werden. Sie werden dabei unmittelbar in den Hauptspeicher geschrieben und sind sofort ohne einen separaten Ladevorgang ausführbar. Zur dauerhaften Speicherung müssen sie allerdings mit SAVE_MACRO oder STORE_MACRO auf einem Massenspeicher abgelegt werden. Das Verfahren ist jedoch nicht sehr übersichtlich und nur bei sehr viel Erfahrung und bei kurzen Makros zu empfehlen.

2.3.5 Speichern

Beim Speichern von Makros sind drei Fälle zu unterscheiden:

a) EDIT_FILE ist noch aktiv

Während des Arbeitens mit EDIT_FILE wird die aktuelle Datei durch gleichzeitiges Betätigen der Tasten <CTRL> und <D> unter der beim Aufruf von EDIT_FILE

angegebenen Bezeichnung oder mit Hilfe des Editorbefehls "W" bzw. "O" unter einer anderen Bezeichnung als ASCII-Datei auf einem Massenspeicher abgelegt. Die Datei kann ein einzelnes oder auch mehrere Makros enthalten.

b) EDIT_MACRO ist noch aktiv

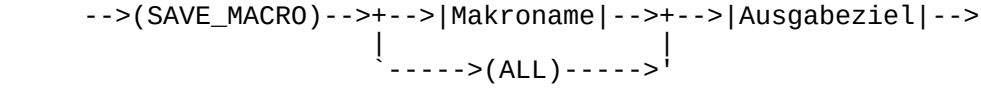
Während des Arbeitens mit EDIT_MACRO kann die aktuelle Datei durch gleichzeitiges Betätigen der Tasten <CTRL> und <D> unter dem beim Aufruf von EDIT_MACRO angegebenen Namen im Hauptspeicher oder mit Hilfe des Editorbefehls "W" bzw. "O" unter Angabe einer Datei auf einem Massenspeicher abgelegt werden. Das Speichern erfolgt im ASCII-Format.

c) Ein im Hauptspeicher befindliches Makro soll dauerhaft gespeichert werden

Makros, die sich im Hauptspeicher befinden, gehen beim Beenden von ME10 verloren. Eine dauerhafte Speicherung ist wahlweise als ASCII-Datei oder als Binärdatei möglich. ASCII-Dateien sind für den Benutzer lesbar und daher auch editierbar. Binärdateien sind für den Benutzer nicht lesbar, haben aber den Vorteil, weniger Speicherplatz zu benötigen.

Das Speichern eines im Hauptspeicher befindlichen Makros als ASCII-Datei erfolgt mit SAVE_MACRO.

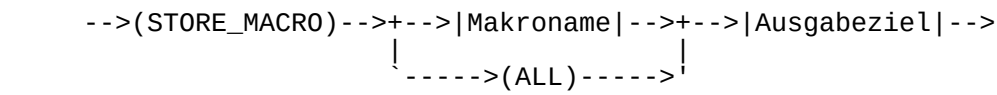
SAVE_MACRO (Interruptfunktion)



SAVE_MACRO überträgt das genannte Makro oder alle (ALL) Makros an das angegebene Ausgabeziel (-> Kap.2.1.3). Die Ausgabe erfolgt im ASCII-Format. Der Name des zu speichernden Makros ist nicht in Hochkommas anzugeben.

Das Speichern eines im Hauptspeicher befindlichen Makros als Binärdatei erfolgt mit STORE_MACRO.

STORE_MACRO (Interruptfunktion)



STORE_MACRO überträgt das angegebene Makro oder alle (ALL) Makros in Binärform an das angegebene Ausgabeziel (-> Kap.2.1.3). Da hierbei ein internes, systemspezifisches Format verwendet wird, sind Binärmakros nicht zwischen ME10-Versionen austauschbar, wenn sie auf verschiedenen Betriebssystemen installiert sind. Ein Austausch von ASCII-Makros ist jedoch problemlos möglich.

STORE_MACRO kann sowohl gesperrte als auch nicht gesperrte Makros sichern. Dateien, die mit STORE_MACRO erstellt werden, können nicht mit INPUT, sondern ausschließlich mit der Funktion LOAD_MACRO geladen werden.

2.3.6 Laden (INPUT, LOAD_MACRO)

Ein in einer Datei gespeichertes Makro muß vor seiner Ausführung in den Hauptspeicher geladen werden. Hierbei werden alle Kommentare (-> Kap.2.8) entfernt, und es findet eine grobe syntaktische Prüfung statt. ASCII-Dateien werden mit INPUT, Binärdateien mit LOAD_MACRO geladen.

INPUT (Interruptfunktion)

```
-->(INPUT)-->|Datei|-->
```

Die spezifizierte ASCII-Datei wird in den Hauptspeicher geladen. Ein Fehler in einer INPUT-Datei bewirkt, daß alle nachfolgenden Informationen nicht mehr aufgenommen werden. Anweisungen, die nicht in "DEFINE *Makroname*" und "END_DEFINE" eingeschlossen und somit als Makro gekennzeichnet sind (-> Kap.4), werden sofort ausgeführt. Sie verbleiben auch nicht im Hauptspeicher. INPUT-Dateien ohne DEFINE und END_DEFINE sind daher im Prinzip Batchdateien, welche beim Laden abgearbeitet werden.

Anweisungen, die in "DEFINE *Makroname*" und "END_DEFINE" eingeschlossen sind, werden von ME10 als Makros angesehen. Sie stehen nach dem Laden zur Ausführung bereit und bleiben solange im Hauptspeicher, bis sie durch Makros mit gleichen Namen überschrieben oder mit DELETE_MACRO gelöscht werden, oder bis ME10 beendet wird.

INPUT-Dateien dürfen geschachtelt strukturiert werden. Das bedeutet, daß eine INPUT-Datei Anweisungen zum Laden weiterer Dateien mit INPUT enthalten kann. INPUT oder LOAD_MACRO dürfen hingegen nicht in Makros verwendet werden.

Hinweis: INPUT in Makros führt zu keiner Fehlermeldung und im allgemeinen auch zu keinen Laufzeitfehlern. Nicht vorhersehbare Wirkungen sind aber trotzdem nicht ganz auszuschließen. Daher ist von einer Verwendung abzuraten.

Beispiel: Im aktuellen Verzeichnis befindet sich die Datei "uebung.mac". Inhalt dieser Datei ist ein Makro mit dem Namen "Pfeil". (Dateinamen haben mit den Namen der darin enthaltenen Makros also nichts zu tun, sie können völlig verschieden sein.)

Die Datei wird nun zunächst in den Hauptspeicher geladen:

```
INPUT 'uebung.mac'
```

Jetzt steht das Makro zur Verfügung und kann durch Eingabe seines Namens gestartet werden:

```
Pfeil
```

Das Makro wird ausgeführt.

LOAD_MACRO (Interruptfunktion)

```
-->(LOAD_MACRO)-->|Datei|-->
```

LOAD_MACRO lädt alle Makros, die in der angegebenen Datei enthalten sind. Die Datei muß mit STORE_MACRO erstellt worden sein, also im Binärformat vorliegen. LOAD_MACRO wird im Prinzip wie INPUT angewendet.

Auf einen von Anfängern gerne gemachten Fehler soll an dieser Stelle hingewiesen werden. Häufig wird der Makrotext zunächst mit EDIT_FILE geschrieben, auf Platte gespeichert, mit INPUT geladen und ausgetestet. Der Benutzer stellt einen Fehler fest, ruft EDIT_MACRO auf, verbessert den Fehler und speichert das Makro mit Hilfe der Tastenkombination <CTRL>+<D> ab. Die verbesserte Version befindet sich jetzt im Hauptspeicher und könnte ohne vorheriges Laden ausgetestet werden. Auf der Platte befindet sich noch die alte, fehlerhafte Version. Der Benutzer ist sich aber über diese Tatsache nicht im klaren und lädt den auf der Platte befindlichen Makrotext in den Hauptspeicher. Dadurch wird das zwischenzeitlich verbesserte Makro vom fehlerhaften alten Makro überschrieben.

2.3.7 Ausdrucken

ME10 kennt keine besondere Anweisung zum Ausdrucken von Makrotexten. Im Prinzip läßt sich dazu jede Anweisung zum Speichern im ASCII-Format verwenden, wenn als Ausgabeziel der Drucker angegeben werden kann. Um das Überschneiden verschiedener Druckaufträge zu vermeiden, sollte bei HP-UX-Mehrplatzsystemen mit gemeinsamem Drucker keine Gerätedatei, sondern das Spoolprogramm "lp" angegeben werden.

Beispiele

Ein im Hauptspeicher befindliches Makro "Test_mac" kann bei HP-UX-Systemen mit folgender Anweisung ausgedruckt werden:

```
SAVE_MACRO Test_mac '| lp'
```

Der Drucker wird bei MS-DOS-Systemen als Gerätedatei "prn" angegeben. Die Anweisung müßte hier wie folgt lauten:

```
SAVE_MACRO Test_mac 'prn'
```

Die Übergabe des Textes an den Druckerspooler PRINT kann hier zu Problemen führen.

Ein im ME10-Texteditor geladener Makrotext kann bei HP-UX-Systemen mit folgendem Editorbefehl (-> Kap.3) ausgedruckt werden:

```
$W '| lp'
```

MS-DOS-Systeme verfügen zusätzlich noch über die in Kapitel 11 beschriebene Interruptfunktion COPY_TO_DEVICE, mit der Dateien gezielt auf einem an einer parallelen oder seriellen Schnittstelle angeschlossenen Gerät ausgegeben werden können.

2.4 Verwalten von Makros

2.4.1 Auflisten (LIST_MACRO_NAMES)

Mit LIST_MACRO_NAMES können die Namen aller momentan im Hauptspeicher geladenen Makros aufgelistet sein. Gesperrte Makros sind dabei durch einen Stern gekennzeichnet. Da ME10 viele Systemmakros benutzt, ist die Ausgabeliste sehr lang.

LIST_MACRO_NAMES (Interruptfunktion)

```
-->(LIST_MACRO_NAMES)-->|Ausgabeziel|-->
```

2.4.2 Sperren (SECURE_MACRO)

Im Hauptspeicher befindliche Makros können mit SECURE_MACRO gegen Editieren, Speichern als ASCII-Datei und Ablaufverfolgung gesperrt werden. Ein gesperrtes Makro läßt sich verwenden und mit STORE_MACRO auch als Binärdatei speichern. Da die Sperrung nicht mehr rückgängig gemacht werden kann, sollte stets auch noch eine nicht gesperrte Version des Makros aufbewahrt werden.

SECURE_MACRO (Interruptfunktion)

```
-->(SECURE_MACRO)-->+-->|Makroname|-->+-->
                    |                 |
                    `----->(ALL)----->'
```

Die Option ALL ermöglicht eine Sperrung aller im Hauptspeicher befindlichen Makros.

2.4.3 Löschen (DELETE_MACRO)

DELETE_MACRO löscht das angegebene Makro oder alle (ALL) Makros aus dem Hauptspeicher. Die Option ALL sollte mit Vorsicht verwendet werden, da auch das Menüsystem von Makros gebildet wird und danach nicht mehr zur Verfügung steht. Es läßt sich jedoch notfalls durch einen Neustart von ME10 wieder einrichten.

DELETE_MACRO (Interruptfunktion)

```
-->(DELETE_MACRO)-->+-->|Makroname|-->+-->
                    |                 |
                    `----->(ALL)----->'
```

2.5 Formaler Sprachaufbau

Makrotexte können die Standardschriftzeichen der ASCII-Tabelle sowie die ASCII-Zeichen Nr. 161 bis 254 enthalten. Die Makrosprache ist nicht zeilengebunden. Anweisungen werden durch ein Leerzeichen voneinander getrennt. Sie können auf mehrere Zeilen verteilt sein, und es können auch mehrere Anweisungen in einer Zeile stehen. Ein Zeilenwechsel wirkt wie ein Leerzeichen. Zur besseren Lesbarkeit empfiehlt es sich, Makrotext durch Leerzeilen und Einrückungen optisch zu strukturieren. Solche Maßnahmen richten sich jedoch nur an den Benutzer, sie sind formal nicht notwendig.

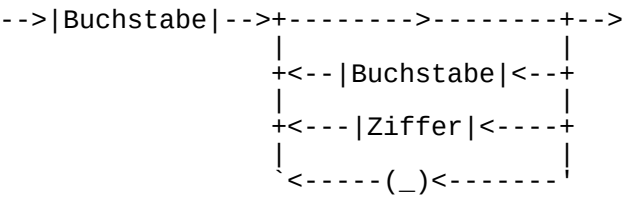
Leerzeichen zwischen Zeichenfolgen sind erforderlich, wenn aus Gründen der Eindeutigkeit Zeichenfolgen von anderen Zeichenfolgen abgegrenzt werden müssen und eine eindeutige Abgrenzung nicht bereits durch Klammern, Kommas oder Operatoren aus Sonderzeichen erfolgt. Leerzeichen sind hingegen unzulässig in zusammengehörigen Zeichenfolgen (reservierte Schlüsselwörter, Namen, Operatoren). Bei Strings (= Zeichenketten) ist zu beachten, daß Leerzeichen als Text interpretiert werden.

Schlüsselwörter (ME10-Anweisungen bestehen aus Schlüsselwörtern und Parametern) müssen einheitlich in Kleinbuchstaben oder Großbuchstaben geschrieben werden. Eine gemischte Groß/Kleinschreibung ist hier nicht zulässig. Benutzerdefinierte Variablen- und Makronamen hingegen können prinzipiell in einer beliebigen Kombination aus Groß- und Kleinbuchstaben geschrieben werden. Wie im nächsten Kapitel noch näher begründet wird, ist es jedoch dringend zu empfehlen, diese Namen generell mit einem Großbuchstaben beginnen zu lassen und weitere Buchstaben klein zu schreiben.

In Programmbeispielen dieses Skripts wird für Makronamen und Variablennamen die Groß/Kleinschreibung verwendet. Schlüsselwörter werden zur besseren Unterscheidung mit Großbuchstaben geschrieben.

2.6 Namen in Makros

|Name|



Benutzerdefinierte Namen für Variable und Makros müssen mit einem Buchstaben beginnen. Die nachfolgenden Zeichen können Buchstaben, Ziffern oder Unterstreichungszeichen "_" sein. Die Anzahl der Zeichen ist nicht begrenzt.

Reservierte Schlüsselwörter dürfen auf keinen Fall als benutzerdefinierte Namen verwendet werden. Die von ME10 beanspruchten Schlüsselwörter lassen sich mit LIST_KEYWORDS ermitteln.

LIST_KEYWORDS (Interruptfunktion)

```
-->(LIST_KEYWORDS)-->|Ausgabeziel|-->
```

Die Liste aller von ME10 verwendeten Schlüsselwörter wird in das angegebene Ausgabeziel (Datei, Gerätedatei, Programm) geschrieben.

Im vorangegangenen Kapitel wurde bereits empfohlen, benutzerdefinierte Namen mit Großbuchstaben beginnen zu lassen und weitere Buchstaben klein zu schreiben. Der Grund für diese Empfehlung ist die Tatsache, daß ME10 sehr viele Schlüsselwörter besitzt und daher die Gefahr besteht, eine Variable oder ein Makro versehentlich mit einem Schlüsselwort zu benennen. Diese Gefahr besteht jedoch nicht, wenn benutzerdefinierte Namen generell in Groß/Kleinschreibung angegeben werden. ME10 erkennt nämlich Namen nur dann als Schlüsselwörter, wenn sie einheitlich groß oder klein geschrieben sind.

Beispiel: Der Benutzer weiß nicht, daß "TEXT" ein ME10-Schlüsselwort ist und gibt einer Variablen diesen Namen. Dabei ist es unerheblich, ob er die Variable mit "TEXT" oder mit "text" benennt. Das Makro reagiert mit einer Fehlermeldung. Hätte er die Variable mit "Text" bezeichnet, wäre der Fehler nicht aufgetreten.

2.7 Datentypen

ME10 unterscheidet die Datentypen Befehl (COMMAND), Interruptfunktion (FUNCTION), Option (QUALIFIER), Pseudokommando (PSEUDO_COMMAND), Integrierte Funktion (ARITHM_FUNCTION), Operator-Symbol (SYMBOL), Makro (MACRO), Text (STRING), Zahl (NUMBER), 2D-Vektor (PNT) und 3D-Vektor (PNT3). Es ist streng darauf zu achten, daß Daten verschiedenen Typs nicht in unzulässiger Weise miteinander verknüpft werden. So ist es beispielsweise möglich, eine Zahl mit einem Vektor zu multiplizieren. Eine Addition von Zahl und Vektor ist hingegen (wie in der Mathematik) nicht möglich. Gerade dieser Fehler wird aber beim Schreiben von Makros sehr häufig gemacht.

2.8 Kommentare

Ein in geschweiften Klammern {..} oder in die Zeichenfolgen (* und *) eingeschlossener Text wird beim Laden eines Makros entfernt und daher auch nicht ausgeführt. Auf diese Weise ist es möglich, Kommentare in den Makrotext einzufügen oder in der Testphase einzelne Programmbereiche zunächst nicht zur Ausführung zuzulassen. Kommentare dürfen allerdings selbst keine weiteren Kommentare enthalten. Dies gilt auch, wenn unterschiedliche Kommentarsymbole verwendet werden.

|Kommentar|

```
-->+-->( { )--->|beliebige Zeichenfolge ohne } oder *) |-->( } )--->+-->
   |                                                               |
   `-->( (* )-->|beliebige Zeichenfolge ohne } oder *) |-->( *) )-->'
```

Das Einfügen von Kommentaren ist nicht möglich, wenn das Makro mit EDIT_MACRO erstellt wird. Die Kommentare erscheinen zwar zunächst auf dem Bildschirm, werden aber bereits beim Speichern mittels der Tastenkombination <CTRL>+<D> aus dem Text entfernt und sind daher verloren.

2.9 Fehlerbehandlung

2.9.1 Fehlerarten, Systemreaktionen und Hilfsmittel

Fehler können als Syntaxfehler, Laufzeitfehler und logische Fehler auftreten. Syntaxfehler sind Verstöße gegen die Sprachkonventionen und werden bei der beim Interpretieren der Anweisungen stattfindenden Syntaxprüfung erkannt. Laufzeitfehler treten auf, wenn in formal richtigen Anweisungen unzulässige Parameter zur Anwendung kommen. Sie werden beim Ausführen der Anweisungen erkannt. So ist zum Beispiel die Anweisung "LET A (B / C)", bei der eine Division der Variablen "B" durch die Variable "C" stattfindet, formal richtig. Sie führt aber zu einem Laufzeitfehler, wenn "C" den Wert "0" annimmt.

Logische Fehler liegen vor, wenn formal richtige Anweisungen mit zulässigen Parametern zu falschen Ergebnissen führen. Man kann sie auch als "formal richtig programmierter Unsinn" bezeichnen. Ein logischer Fehler wäre es zum Beispiel, wenn eine Rechenvorschrift die Addition zweier Zahlen verlangt, der Benutzer aber versehenlich eine Subtraktion programmiert. Logische Fehler können nur durch Austesten des Programms gefunden werden.

Unzulässige Benutzereingaben wie zum Beispiel die Eingabe einer Zahl, wenn Text erforderlich ist, führen in der Regel zu keinem Fehler, sondern der Benutzer wird wie beim interaktiven Arbeiten durch einen Signalton gewarnt und zur erneuten Eingabe aufgefordert.

Eine Syntaxfehlerprüfung findet in gewissem Umfang schon beim Laden von Makros statt. Der Benutzer erhält allerdings kaum Hinweise auf die Art der Fehler, meist erfolgt nur ein akustisches Signal. Wenn der Ladevorgang mit der Meldung "Makro-Definition eingeben" endet, wurde die ein Makro abschließende Anweisung "END_DEFINE" (-> Kap.4) nicht gefunden. Dies kann folgende Ursachen haben:

- END_DEFINE fehlt
- Der Unterstrich in END_DEFINE fehlt (--> "END DEFINE")
- END_DEFINE wurde infolge einer nicht geschlossenen Kommentarklammer beim Laden entfernt

Die meisten Fehler werden erst beim Interpretieren des Makros erkannt. Sie führen normalerweise zum Abbruch. Auch hier sind die ausgegebenen Fehlermeldungen sehr dürftig. Der Benutzer wird häufig nicht über die Art eines Fehlers informiert (es ertönt nur ein akustisches Signal), und er bekommt auch nicht automatisch angezeigt, an welcher Stelle des Makrotextes der Fehler aufgetreten ist. In solchen Fällen muß der Fehler eingegrenzt werden. Das kann dadurch geschehen, daß Teile des Makros, in denen er vermutet wird, zunächst mit den Kommentarsymbolen { } bzw. (* *) wirkungslos gemacht werden. Die Teile dürfen dann allerdings keine weiteren Kommentarklammern enthalten.

Tritt der Fehler nachher nicht mehr auf, befindet er sich im wirkungslos gemachten Teil und kann weiter eingegrenzt werden.

Die Eigenheit von Interpretersprachen bedingt auch, daß Fehler nur in den tatsächlich während des Programmlaufs interpretierten Programmteilen gefunden werden. Beim Austesten von Makros ist also darauf zu achten, daß wirklich alle Programmteile durchlaufen werden. Aber selbst in noch so gut ausgetesteten Makros können immer noch unerwartete Benutzereingaben zu Fehlern führen. Werden Makros nicht nur von ihrem Autor benutzt, sollten zumindest die wahrscheinlichsten Bedienungsfehler vom Makro selbst abgefangen werden.

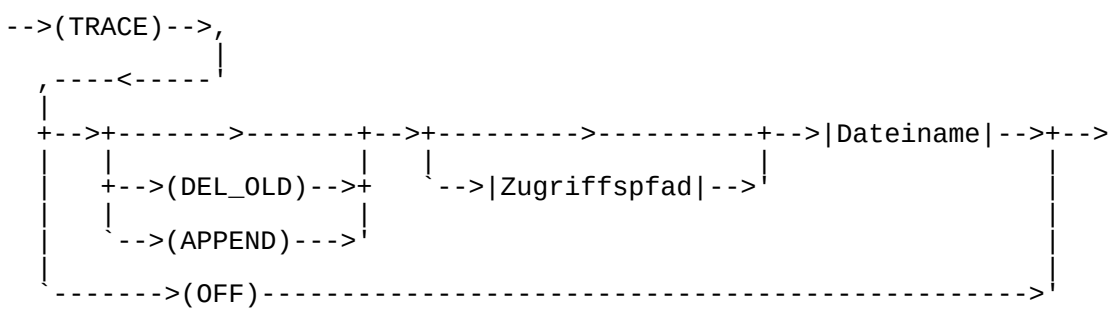
Fehlerhafte Makros können in manchen Fällen zu einem Aufhängen des Programms führen. Man erkennt diesen Zustand daran, daß das System auf keine normalen Benutzereingaben reagiert. In diesem Fall läßt sich der Makrolauf mit Hilfe der ESCAPE-Taste abbrechen. Voraussetzung ist allerdings, daß BREAK-Signale nicht abgefangen werden, -> Kap.2.2).

Neben der bereits beschriebenen Möglichkeit, Teile eines Makros als Kommentar zu kennzeichnen und dadurch wirkungslos zu machen, steht als wichtigste Hilfe zur Fehlersuche und zum Austesten die TRACE-Funktion zur Verfügung. Die Funktion PROMPT_LIST ermöglicht außerdem eine nochmalige Ausgabe von zunächst übersehenen Fehlermeldungen. Fehlerhafte Benutzereingaben können mit ON_ERROR abgefangen werden. ME10 verfügt ab Version 4 über Anweisungen, mit denen eine Fortsetzung des Makrolaufs bei einigen Fehlern möglich ist. Es handelt sich hierbei um die Funktionen TRAP_ERROR, CHECK_ERROR, ERROR_LOG und ERROR_STR. Ein wirkungsvoller Debugger fehlt allerdings noch immer.

2.9.2 TRACE

TRACE ermöglicht eine Protokollierung aller vom System verarbeiteten Anweisungen.

TRACE (Interruptfunktion)



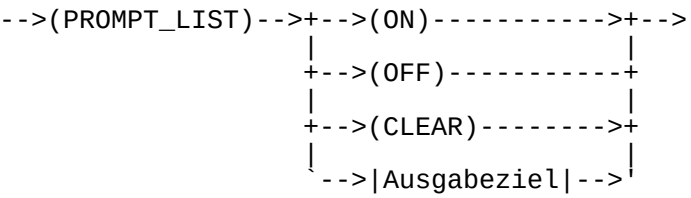
TRACE eröffnet und schließt die TRACE-Datei. Eine offene TRACE-Datei nimmt eine Kopie aller vom System verarbeiteten Anweisungen (Aufrufe von Befehlen und Interruptfunktionen, Punkt-, Wert- und Stringeingaben, Ergebnisse von Ausdrücken) auf. Diese Anweisungen können vom Benutzer oder von Makros kommen. Die Anweisung TRACE OFF schließt die aktuelle TRACE-Datei. Da aus Gründen der Eindeutigkeit immer nur eine TRACE-Datei offen sein kann, wird vor dem Öffnen eine andere geöffnete TRACE-Datei geschlossen.

Die TRACE-Datei gibt also Aufschluß darüber, welche Angaben das System in welcher Weise verarbeitet hat. Beim vorzeitigen Abbruch eines Makros ist der dafür verantwortliche Fehler in den letzten Zeilen der in der Datei protokollierten Makroanweisungen zu suchen.

2.9.3 PROMPT_LIST

ME10 speichert automatisch die letzten 500 Systemmeldungen in einem internen Speicher. Sie werden mit Hilfe der Interruptfunktion PROMPT_LIST verwaltet. Manchmal geben diese Meldungen Aufschluß über Fehler, die zwar gemeldet, aber vom Benutzer zunächst nicht beachtet wurden.

PROMPT_LIST (Interruptfunktion)

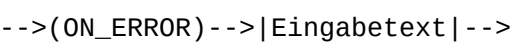


Die Optionen ON und OFF bestimmen, ob die automatische Speicherung der Systemmeldungen ein- oder ausgeschaltet ist. Standardmäßig ist die Speicherung eingeschaltet. PROMPT_LIST CLEAR löscht den internen Speicher. Bei Angabe eines Ausgabeziels (-> Kap.2.1.3) wird sein Inhalt ausgegeben.

2.9.4 ON_ERROR

Bei fehlerhaften Benutzereingaben wird normalerweise die Eingabe erneut angefordert. ON_ERROR erlaubt die automatische Verwendung von Ersatzeingaben ohne erneute Benutzereingabe.

ON_ERROR (Interruptfunktion)



Der nach ON_ERROR angegebene Eingabetext wird statt der fehlerhaften Systemeingabe als Ersatzeingabe verwendet und anschließend gelöscht. Die Ersatzeingabe kann alle in ME10 zugelassenen Datentypen aufweisen. Sie muß jedoch grundsätzlich in Textform (also als Stringkonstante, Stringvariable oder Stringausdruck) angegeben werden. Die Textform verhindert eine sofortige Ausführung der Ersatzeingaben, sie wird quasi "maskiert". Stringvariablen dürfen in diesem Fall jedoch nicht als lokal vereinbart werden, es sind hier also nur globale Variablen zulässig. (Der Unterschied zwischen globalen und lokalen Variablen wird in Kapitel 4 erläutert.)

Die Ersatzeingabe wird nach ihrer Verwendung gelöscht. Sie ist erneut zu definieren, wenn sie beim nächsten Fehler wieder verwendet werden soll. ON_ERROR '' deaktiviert die Ersatzeingabe.

Beispiele

```
DEFINE On_error_test_1
  LOCAL A
  LOCAL R
  ON_ERROR '10,10'
  READ PNT 'Punkt A =?' A
  ON_ERROR '20'
  READ NUMBER 'Radius R =?' R
  CIRCLE CENTER A R END
  ON_ERROR ''
END_DEFINE
```

```
DEFINE On_error_test_2
  LOCAL A
  LOCAL B
  ON_ERROR '0'
  READ NUMBER 'Zahl A =?' A
  ON_ERROR '0'
  READ NUMBER 'Zahl B =?' B
  DISPLAY ('A + B = ' + STR (A + B))
  ON_ERROR ''
END_DEFINE
```

Beim Makro On_error_test_1 muß der Benutzer zunächst einen Punkt und dann einen Zahlenwert eingeben. Mit diesen Werten wird anschließend ein Kreis gezeichnet. Normalerweise würden die Eingaben bei einer fehlerhaften Benutzereingabe erneut angefordert werden. Fehlerhafte Benutzereingaben wären z.B. die Eingabe eines Zahlenwertes, wenn ein Punkt gefordert ist oder die Angabe eines nicht vorhandenen Schnittpunktes bei der Punktbestimmung. Die ON_ERROR-Anweisungen bewirken, daß im Fehlerfall die darin spezifizierten Werte als Eingabe verwendet werden. Der Variablen "A" würde also bei einer falschen Benutzereingabe der Punkt "10,10" zugewiesen, der Variablen "R" der Zahlenwert "20".

Beim Makro On_error_test_2 soll den Variablen A und B im Fehlerfall jeweils der Zahlenwert "0" zugewiesen werden. Die dafür vorgesehene Anweisung ON_ERROR '0' muß zweimal verwendet werden, da sie nur für einen Fehler gilt.

ON_ERROR ist nicht in der Lage, den Abbruch eines Makros aufgrund eines auftretenden Fehlers zu verhindern. Die Art des Fehlers entscheidet, ob der Makrolauf fortgesetzt werden kann. Ist die Fortsetzung nach einer korrigierten manuellen Benutzereingabe möglich, so läßt sich mit ON_ERROR lediglich eine automatische Ersatzeingabe ohne Zutun des Benutzers erreichen. Wird der Makrolauf hingegen selbst nach einer korrigierten manuellen Eingabe abgebrochen, kann auch ON_ERROR den Abbruch nicht verhindern.

2.9.5 TRAP_ERROR

Ein Makro wird normalerweise beim ersten erkannten Fehler abgebrochen. Fehlerhafte Benutzereingaben sind unkritisch, auch beim Makro führt eine falsche Eingabe (z.B. die Eingabe einer Zahl, wenn ein Text angefordert wird) lediglich zu einer Wiederholung der Eingabeanforderung und nicht zum Programmabbruch.

Manchmal ist es wünschenswert, den Makrolauf auch bei anderen Fehlern zunächst fortzusetzen. ME10 verfügt ab Version 4 mit der Funktion TRAP_ERROR über die Möglichkeit, Fehlermeldungen und Programmabbrüche bei bestimmten Fehler zu unterdrücken. Die Fehler werden dadurch nicht behoben, sondern ein begonnener Prozeß wird unter Inkaufnahme der Fehler soweit wie möglich fortgesetzt. TRAP_ERROR wird ohne Parameter aktiviert und durch CHECK_ERROR deaktiviert.

```
TRAP_ERROR (Interruptfunktion)

-->(TRAP_ERROR)-->
```

In der Programmbeschreibung heißt es, daß TRAP_ERROR auf Fehler anwendbar ist, die auf einer falschen Eingabe beruhen.

Beispiele

- Eine nicht vorhandene Datei soll zum Lesen geöffnet werden
- Eine nicht vorhandene logische Tabelle soll einer Änderung unterzogen werden
- Eine nicht vorhandene Anzeigetabelle soll gelöscht werden

Es zeigt sich jedoch, daß auch Syntaxfehler, die vom Interpreter eindeutig abgegrenzt werden können, nach Aktivierung von TRAP_ERROR nicht mehr zum Abbruch eines Makros führen. Einige Fehlerreaktionen lassen sich jedoch auch mit TRAP_ERROR nicht unterdrücken. So führt z.B. der Aufruf eines nicht vorhandenen Makros auf jeden Fall zum Abbruch, da es ja danach völlig offen ist, wie das Makro fortgesetzt werden soll.

TRAP_ERROR beeinflußt das Fehlerverhalten von ME10 insgesamt, also auch das Fehlerverhalten beim interaktiven Arbeiten. Die Funktion sollte daher auf jeden Fall wieder deaktiviert werden, wenn eine Unterdrückung der Fehlerreaktionen nicht mehr erforderlich ist, da ME10 sonst praktisch keine Fehler mehr ausgibt.

2.9.6 CHECK_ERROR

CHECK_ERROR (integrierte Funktion)

```
-->(CHECK_ERROR)-->
```

CHECK_ERROR prüft, ob seit dem letzten Aufruf von TRAP_ERROR ein oder mehrere Fehler registriert wurden. Falls dies der Fall ist, wird die Zahl "1", andernfalls die Zahl "0" zurückgegeben. Der Fehlerzustand kann beliebig oft abgefragt werden, da die Zustandskennung durch TRAP_ERROR und nicht durch CHECK_ERROR zurückgesetzt wird. CHECK_ERROR deaktiviert TRAP_ERROR.

Bei CHECK_ERROR handelt es sich um eine sogenannte integrierte Funktionen, die eine Zahl als Ergebnis liefert. Die Zahl muß in einer Anweisung als Argument oder Parameter verwendet werden. Eine Anweisung, die nur aus einer integrierten Funktion besteht, ergibt keinen Sinn.

2.9.7 ERROR_STR

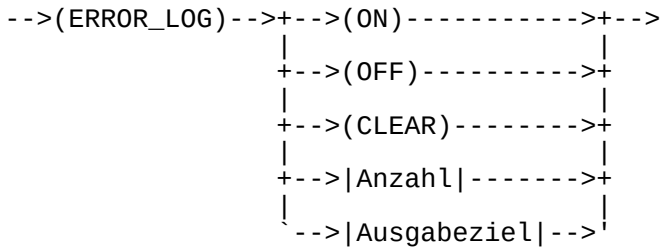
ERROR_STR (integrierte Funktion)

```
-->(ERROR_STR)-->
```

Nach Aktivierung von TRAP_ERROR werden die meisten Fehler lediglich registriert. Das System erzeugt zwar noch Fehlermeldungen, sie werden aber nicht mehr ausgegeben. ERROR_STR liefert die nach der Aktivierung von TRAP_ERROR zuerst erzeugte Fehlermeldung. Auch bei ERROR_STR handelt es sich um eine integrierte Funktion, die nur als Argument oder Parameter verwendet werden kann.

2.9.8 ERROR_LOG

ERROR_LOG (Interruptfunktion)



Nach Aktivierung von TRAP_ERROR werden leichtere Fehler lediglich registriert. Das System erzeugt hierbei Warnungen oder Fehlermeldungen, die aber nicht ausgegeben werden. ERROR_LOG bewirkt ein Speichern der Warnungen und Fehlermeldungen in einem internen Speicher. Bei schweren Fehlern erfolgt generell ein Makroabbruch und der erkannte Fehler wird angezeigt.

Mit den Optionen ON und OFF kann das Speichern ein- bzw. ausgeschaltet werden. Standardmäßig ist ON eingestellt. CLEAR löscht den internen Fehlerberichtsspeicher. Mit "Anzahl" kann die Anzahl der zu speichernden Meldungen eingestellt werden. Falls mehr Meldungen als hierdurch angegeben gespeichert wurden, werden die ältesten Meldungen überschrieben. Durch Angabe einer negativen Zahl läßt sich die Speicherung bis zur erneuten Angabe einer positiven Zahl unterbrechen. Der Inhalt des Fehlerberichtsspeichers kann durch Angabe eines Ausgabeziels (-> Kap.2.1.3) ausgegeben werden.

2.10 Koordinatensysteme

ME10 arbeitet mit einem internen Koordinatensystem und mit einem Eingabekoordinatensystem. Das interne Koordinatensystem ist ortsfest und seine X- und Y-Achsen verlaufen horizontal und vertikal. Es dient zur internen Darstellung von Geometriedaten und wird auch als absolutes Koordinatensystem bezeichnet. Koordinateneingaben, Längeneingaben und Winkelangaben, die beim interaktiven Arbeiten oder in Makros erfolgen, beziehen sich grundsätzlich auf das Eingabekoordinatensystem mit Ausnahme folgender Sonderfälle:

- Die Eingaben erfolgen durch Digitalisierung
- Die Eingaben erfolgen in Bildpunktkoordinaten oder in Tablettkoordinaten

Das Eingabekoordinatensystem wird auch als relatives Koordinatensystem bezeichnet. Alle Benutzereingaben werden vom System automatisch in absolute Koordinaten umgerechnet und gespeichert. Bei den in der Statuszeile angezeigten oder mit Hilfe einer Meßfunktion ermittelten Koordinaten handelt es sich ebenfalls um relative Koordinaten.

Beim Systemstart sind beide Koordinatensysteme gleich. Das Eingabekoordinatensystem kann weitgehend den Bedürfnissen des Benutzers angepaßt werden. Es läßt sich verschieben, um seinen Ursprung drehen und mitlaufend einrichten. Die Achsen können beliebige Winkel zueinander und zur Horizontalen bzw. Vertikalen haben und mit beliebiger Einteilung versehen werden.

ME10 verfügt über keine direkte Möglichkeit, absolute Koordinaten einzugeben oder zu ermitteln. Sollte dies einmal notwendig sein, so muß das Eingabekoordinatensystem so eingerichtet werden, daß es wieder mit dem absoluten Koordinatensystem zusammenfällt. Das Eingabekoordinatensystem wird mit Interruptfunktionen verändert, deren Namen mit "CS" beginnen. Die dafür geltenden Option ABSOLUT bewirkt, daß sich Angaben auf das absolute Koordinatensystem beziehen. In der nachfolgenden Tabelle sind die zum Verändern des Eingabekoordinatensystems vorgesehenen Interruptfunktionen aufgeführt. Nähere Angaben können den ME10-Handbüchern und der Help-Datei entnommen werden.

Funktion	Wirkung
CS_AXIS	Verändert die Achsenrichtungen und Eingabeeinheiten des Eingabekoordinatensystems. Die Lage des Ursprungs bleibt gleich.
CS_MIRROR	Spiegelt eine Achse des Eingabekoordinatensystems an der anderen Achse.
CS_REF_PT	Verschiebt den Ursprung des Eingabekoordinatensystems.
CS_ROTATE	Dreht das Eingabekoordinatensystem um seinen Ursprung.
CS_SET	Verändert den Ursprung und die Achsenrichtungen des Eingabekoordinatensystems. Die Eingabeeinheiten werden beibehalten.
UNITS	Legt die dem absoluten Koordinatensystem und dem Eingabekoordinatensystem zugrundeliegenden Maßeinheiten fest.

Beispiel: Das Eingabekoordinatensystem wurde durch vorangegangene Benutzereingaben verändert und soll nun wieder so eingerichtet werden, daß es mit dem absoluten Koordinatensystem zusammenfällt. Die Anweisung könnte wie folgt lauten:

```
CS_SET THREE_PTS ABSOLUTE 0,0 ABSOLUTE 1,0 ABSOLUTE 0,1
```

Die nachfolgenden Anweisungen hätten die gleiche Wirkung:

```
CS_REF_PT ABSOLUTE 0,0
CS_AXIS ABSOLUTE 0 1 ABSOLUTE 90 1
```

Die Option ABSOLUTE bestimmt, daß sich die Angaben zum Verändern des Eingabekoordinatensystems auf das absolute Koordinatensystem beziehen. Sie hätte auch wegfallen können, da sie standardmäßig gilt.

Weiteres Beispiel: Der Ursprung des Eingabekoordinatensystems befindet sich im Punkt X=100/Y=150 des absoluten Koordinatensystems. Abszisse und Ordinate stehen rechtwinklig zueinander, sind aber gegenüber den Achsen der Horizontalen bzw. Vertikalen um +15 Grad gedreht.

Es werden nun folgende Anweisungen eingegeben:

```
CS_REF_PT ABSOLUTE 10,20
CR_ROTATE ABSOLUTE 5
```

Der Ursprung des Eingabekoordinatensystems befindet sich danach im Punkt X=10/Y=20 des absoluten Koordinatensystems. Abszisse und Ordinate sind um +5 Grad gegenüber der Horizontalen bzw. Vertikalen verdreht. Die Option ABSOLUTE hätte auch wegfallen können, da sie standardmäßig gilt. Bei Verwendung der Option RELATIV würden sich die

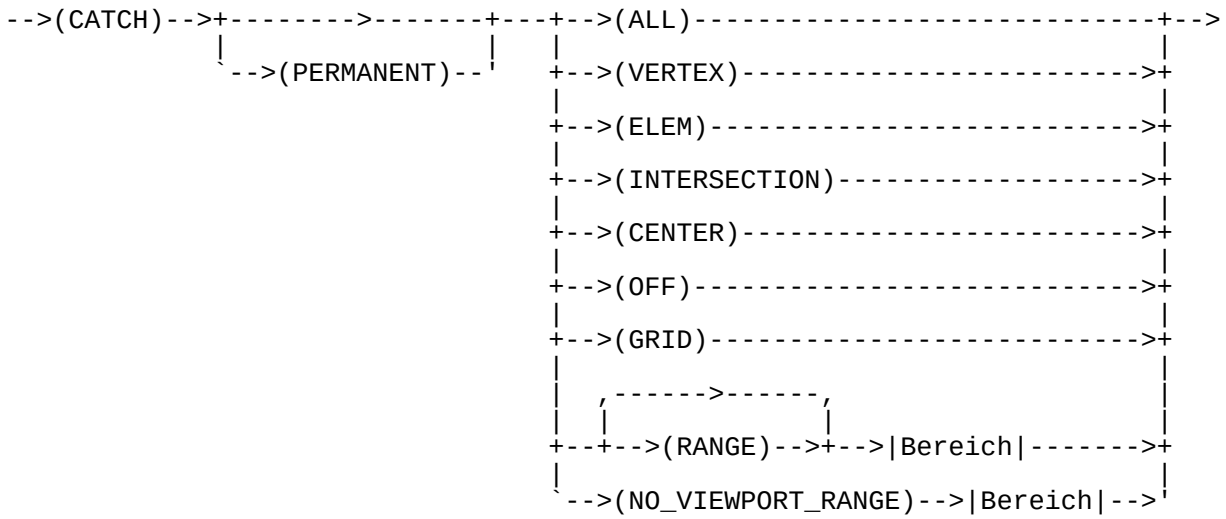
Angaben auf das bisher noch geltende Eingabekoordinatensystem beziehen. Nach Ausführung der Anweisung hätte der Ursprung die Absolutkoordinaten X=110/Y=170, und die Achsen hätten einen Winkel von 20 Grad zur Horizontalen bzw. Vertikalen.

2.11 Makros und Objektfang

ME10 verfügt über eine Fangfunktion, die ein Einfangen von Punkten im Bereich des Fadenkreuzes bewirkt, wenn der Benutzer zur Bestimmung eines Punktes oder zur Auswahl eines Objekts aufgefordert wird. Diese Fangfunktion ist nur bei einer Punktbestimmung oder Objektauswahl mit Hilfe des Fadenkreuzes wirksam, bei anderen Verfahren (z.B. Punktbestimmung oder Objektauswahl durch Angabe von Koordinaten) existiert standardmäßig kein Fangbereich. Dies gilt sowohl für das interaktive Arbeiten mit ME10 als auch für Makros.

Während beim interaktiven Arbeiten eine Punktbestimmung oder eine Objektauswahl durch Angabe von Koordinaten nur sehr selten vorkommt, wird sie bei Makros häufiger verwendet. So könnte z.B. eine Anweisung zum Löschen eines Elements wie folgt lauten: DELETE 100,100. Sie wirkt standardmäßig nur auf ein Objekt, das einen Punkt mit genau diesen Koordinaten besitzt. Ein Objekt in noch so geringem Abstand wird nicht erfaßt. Es ist jedoch möglich, auch für andere Arten von Punktbestimmungen und Objektauswahlverfahren einen Fangbereich größer Null einzurichten. Hierzu dient die Option NO_VIEWPORT_RANGE der Interruptfunktion CATCH.

CATCH (Interruptfunktion)



Die nach NO_VIEWPORT_RANGE einzugebende Zahl kennzeichnet den Fangbereich in aktuellen Einheiten (UNITS, z.B. Millimetern). Eine nachfolgende Änderung der aktuellen Einheiten ändert nichts an einem zuvor definierten Fangbereich.

3 Der ME10-Texteditor

Bei dem in ME10 integrierten Texteditor handelt es sich um einen Bildschirmeditor. Er wird von allen Befehlen und Interruptfunktionen verwendet, die ein Editieren auf dem Textbildschirm beinhalten. Einfache Editierarbeiten können damit nahezu ohne Einarbeitung durchgeführt werden. Die Pfeiltasten bewegen den Cursor an beliebige Stellen des Textes. Standardmäßig wird vorhandener Text überschrieben, mit der EINFÜGE-TASTE läßt sich zwischen Überschreibe- und Einfügemodus umschalten. Einzelne Zeichen können mit der ENTFERNEN-Taste gelöscht werden. Die Wirkung weiterer Tasten kann den nachfolgenden Tabellen entnommen werden. Alle Angaben beziehen sich auf PC-101-Tastaturen mit deutscher Beschriftung.

A) HP-UX-Systeme

Taste(n)	Wirkung
<Esc>	Bricht den Editiervorgang ab (ohne Sichern)
<Strg> + <D>	Sichert und beendet danach den Editiervorgang
<Pfeil nach oben>	Bewegt den Cursor nach oben
<Pfeil nach unten>	Bewegt den Cursor nach unten
<Pfeil nach rechts>	Bewegt den Cursor nach rechts
<Pfeil nach links>	Bewegt den Cursor nach links
<Pos 1>	Bewegt den Cursor zum Zeilenanfang
<Ende>	Bewegt den Cursor zum Zeilenende
<Shift> + <Pfeil n.oben>	Blättert eine Bildschirmseite nach oben
<Shift> + <Pfeil n.unten>	Blättert eine Bildschirmseite nach unten
<Einfg>	Schaltet zwischen Überschreib- und Einfügemodus um
<Entf>	Löscht ein einzelnes Zeichen
<Shift> + <Einfg>	Fügt eine Leerzeile vor der Cursorposition ein
<Shift> + <Entf>	Löscht eine Zeile vollständig
<Strg> + <Entf>	Löscht den Inhalt einer Zeile (Leerzeile bleibt)
<Alt> + <Entf>	Löscht den Inhalt einer Zeile ab der Cursorposition
<Bild rauf>	siehe nachfolgende Anmerkung
<Bild runter>	" "

B) MSDOS/WINDOWS-SYSTEME

Taste(n)	Aktion
<Esc>	Bricht den Editiervorgang ab (ohne Sichern)
<Strg>+<D>	Sichert und beendet danach den Editiervorgang
<Pfeil nach oben>	Bewegt den Cursor nach oben
<Pfeil nach unten>	Bewegt den Cursor nach unten
<Pfeil nach rechts>	Bewegt den Cursor nach rechts
<Pfeil nach links>	Bewegt den Cursor nach links
<Pos 1>	Bewegt den Cursor an den Anfang der aktuellen Zeile
<Ende>	Bewegt den Cursor ans Ende der aktuellen Zeile
<Strg>+<Seite nach oben>	Blättert eine Bildschirmseite nach oben
<Strg>+<Seite nach unten>	Blättert eine Bildschirmseite nach unten
<Einfg>	Schaltet zwischen Überschreib- und Einfügemodus um
<Entf>	Löscht ein einzelnes Zeichen
<Alt>+<F5>	Fügt eine Zeile ein
<Alt>+<F6>	Löscht eine Zeile
<Alt>+<F7>	Löscht den Rest einer Zeile ab der Cursorposition
<Alt>+<F8>	Löscht den Inhalt einer Zeile
<Bild rauf>	siehe nachfolgende Anmerkung
<Bild runter>	" "

Wirkung der Tasten <Bild rauf> und <Bild runter>:

ME10 speichert Tastatureingaben bis zu einer Obergrenze von 63 Zeilen. Diese Zeilen können durch (u.U.) mehrmaliges Betätigen der Tasten <Bild rauf> oder <Bild runter> erneut ausgegeben werden. Dabei bewirkt die Taste <Bild rauf> eine Ausgabe entgegen der Eingabereihenfolge und die Taste <Bild runter> eine Ausgabe in der Reihenfolge der ursprünglichen Eingabe. Der Speichervorgang läßt sich mit Hilfe der Interruptfunktion "RECALL_BUFFER" steuern und kontrollieren.

Der Texteditor verfügt weiterhin über leistungsfähige Editorbefehle. Normalerweise werden die vom Benutzer eingegebenen Zeichen als Text interpretiert. Um einen Editorbefehl eingeben zu können, ist zunächst in den Befehlseingabemodus umzuschalten. Die Umschaltung wird durch Eingabe eines Umschaltszeichens bewirkt und gilt nur für einen Befehl. Sollen mehrere Editorbefehle nacheinander eingegeben werden (dies kann auch in einer einzigen Zeile geschehen), so ist jedesmal ein Umschaltzeichen voranzustellen. Als Umschaltzeichen dient standardmäßig "$", es kann jedoch umdefiniert werden. Bei Eingabe des Umschaltzeichens springt der Cursor in die letzte Zeile des Textbildschirms. Die danach eingegebenen Zeichen definieren den jeweiligen Editorbefehl, erscheinen also nicht im Text. Die Editorbefehle beziehen sich meist auf bestimmte Stellen im Text, welche durch Marker definiert werden. In der nachfolgenden Tabelle sind Marker und Editorbefehle zusammengestellt. Die in rechteckigen Klammern angegebenen Befehlszusätze sind optional. Als Umschaltzeichen wird hierbei "$" verwendet.

Marker	Bedeutung
^	Anfang der ersten Zeile
.	Anfang der aktuellen Zeile
!	Anfang der letzten Zeile
0..9	Vom Benutzer definierter Marker

Editorbefehle Mk = Marker Str. = String	Wirkung
$Mk	Cursor in gekennzeichnete Zeile positionieren
$'Zeichenfolge'	Cursor auf die nächste 'Zeichenfolge' positionieren
$? ['Schlüsselwort']	Erklärung (zum Schlüsselwort) anzeigen
$C Mk1 Mk2	Textblock ab aktueller Zeile bis einschließlich der durch Marker 1 gekennzeichneten Zeile kopieren und nach der durch Marker 2 gekennzeichneten Zeile einfügen
$D Mk	Textblock ab aktueller Zeile bis einschließlich der durch den Marker gekennzeichneten Zeile löschen
$H ['Schlüsselwort']	Erklärung (zu Schlüsselwort) anzeigen
$L 'Datei'	Die gekennzeichnete Datei nach der aktuellen Zeile einfügen
$M Mk1 Mk2	Textblock ab aktueller Zeile bis einschließlich der durch Marker1 gekennzeichneten Zeile löschen und nach der durch Marker 2 gekennzeichneten Zeile einfügen
$N	Letzten Suchlauf wiederholen
$O 'Ausgabeziel' [Mk]	Textblock ab aktueller Zeile bis einschließlich der durch den Marker gekennzeichneten Zeile in das Ausgabeziel, alten Inhalt gegebenefalls überschreiben

Editorbefehle Mk = Marker Str. = String	Wirkung
$R [V] 'St1' 'St2' [Mk]	Ab aktueller Zeile bis einschließlich der durch den Marker gekennzeichneten Zeile 'String1' durch 'String2' ersetzen. Ohne Angabe eines Markers erfolgt nur nur eine Ersetzung. Ersetzen mit Bestätigung durch Option "V" (Verify) möglich.
$SE 'character'	Neues Umschaltzeichen setzen (Default $)
$SM 0..9	Marker setzen
$W 'Ausgabeziel' [Mk]	Wie $O 'Ausgabeziel', nur ohne Überschreibung

Weitere, hier nicht beschriebene Editorbefehle dienen zur Textformatierung (z.B. zum Erzeugen eines Blocksatzes) und dürfen nicht in Makros verwendet werden. Sie werden im Systemhandbuch zu ME10 erläutert.

Beispiele

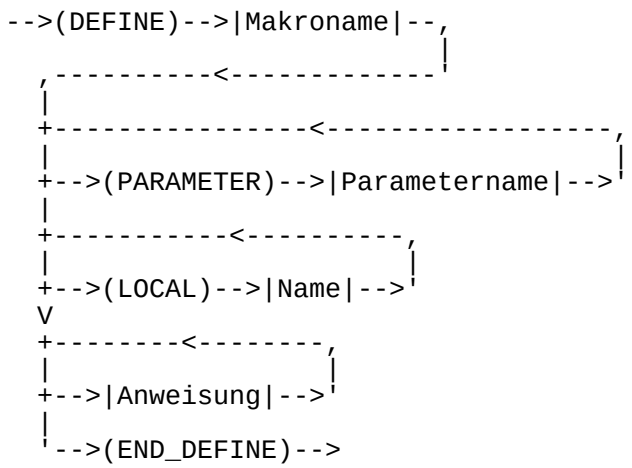
Gewünscht wird	Aktion (Tastenkombination)
Die Textdatei "test1.mac" soll editiert werden.	EDIT_FILE 'test1.mac'
Cursor soll auf die nächste Zeichenfolge "CIRCLE" positioniert werden.	$ 'CIRCLE'
In der aktuellen Zeile soll ein Marker mit der Nummer 5 gesetzt werden.	$ SM 5
Cursor soll an den Anfang des Textes positioniert werden.	$ ^
Corsor soll an den Anfang der durch Marker Nr. 5 gekennzeichnetet Zeile positioniert werden.	$ 5
Ab der aktuellen Cursorpostion bis zum Ende des Textes soll "Line" durch "LINE" ersetzt werden. Das Ersetzen ist zu bestätigen.	$ R V 'LINE' 'Line' !
Der editierte Text soll wieder unter dem beim Laden der Datei angegebenen Namen gespeichert werden.	Tastenkombination <CTRL>+<D> Der Texteditor wird dabei verlassen. Neustart mit EDIT_FILE 'test1.mac'
Der Inhalt der bereits vorhandenen Datei "test2.mac" soll an das Ende des aktuellen Textes angefügt werden.	$! $ L 'test2.mac'
Der aktuelle Text soll unter dem Namen "test3.mac" abgespeichert werden, eine bereits existierende Datei dieses Namens sollte nicht überschrieben werden.	$ W 'test3.mac"
Der Texteditor soll ohne Speichern verlassen werden.	ESCAPE-Taste betätigen

Anmerkung: Die bei den Aktionen angegebenen Leerzeichen können auch weggelassen werden. Sie dienen nur der besseren Lesbarkeit.

Nachdem nun der Texteditor bekannt ist, können Sie mit der ersten Übung beginnen. Zuvor sollten Sie sich jedoch eine zweckmäßige Arbeitsumgebung schaffen. ME10 bietet die Möglichkeit, die Funktionstasten F1 bis F8 mit oft benötigten Benutzereingaben zu belegen. In Kapitel 14.2 wird eine für die Entwicklung von Makros zweckmäßige Belegung beschrieben. Es wird Ihnen sicher viel Arbeit ersparen, wenn Sie diese Tastaturbelegung jetzt vornehmen, auch wenn Sie die einzelnen Anweisungen dazu im Moment noch nicht alle verstehen.

4 Aufbau eines Makros

Ein Makro hat folgenden Aufbau:



DEFINE, END_DEFINE, PARAMETER und LOCAL werden auch als Vereinbarungen bezeichnet.

4.1 Vereinbarungen

4.1.1 DEFINE, END_DEFINE

Durch DEFINE wird vereinbart, daß alle bis END_DEFINE folgenden Anweisungen Bestandteile des mit "Makroname" bezeichneten Makros sind und unter Angabe des Makronamens ausgeführt werden können.

4.1.2 PARAMETER

Mit PARAMETER können Parameter vereinbart werden. Die Parameter sind mit einem Namen gekennzeichnet. Ihnen werden beim Aufruf des Makros die Werte zugewiesen, die dem Makroaufruf folgen. Die Reihenfolge der Parametervereinbarungen und die Reihenfolge der dem Makroaufruf folgenden Parameter ergeben die gegenseitige Zuordnung. Dem ersten vereinbarten Parameter wird also der erste, dem zweiten vereinbarten Parameter der zweite Wert zugeordnet (usw.), welcher dem Makroaufruf folgt. Parameter gelten in dem Makro, in dem sie vereinbart wurden sowie in allen in einer Aufrufkette hierarchisch tieferstehenden Makros. In sonstigen Makros sowie auf Systemebene (also beim interaktiven Arbeiten mit ME10) sind sie nicht bekannt. Die Anweisung PARAMETER wird normalerweise dann verwendet, wenn das Makro von einem anderen Makro aufgerufen wird und wenn Werte vom aufrufenden an das aufgerufene Makro übergeben werden sollen. Parameter können aber auch verwendet werden, wenn der Benutzer Makros selbst aufruft.

Beispiel: Makro Addiere addiert die seinem Aufruf folgenden Zahlenwerte und weist das

Ergebnis der globalen Variablen "C zu. Der Aufruf "Addiere 7 5" führt also dazu, daß Parameter "A" den Wert 7, Parameter "B" der Wert "5" und die Variable "C" den Wert 12 erhält.

```
DEFINE Addiere
  PARAMETER A
  PARAMETER B
  LET C (A + B)
END_DEFINE
```

Eine Rückgabe von Werten an das aufrufende Makro oder an das System ist mit Hilfe von Parametern nicht möglich.

4.1.3 LOCAL

Mit LOCAL werden lokale Namen vereinbart. Die Namen können danach Variablen oder Makros kennzeichnen. (Genaugenommen kennt ME10 keine Variablen im üblichen Sinne. Variablen sind im Prinzip Makros, die einen einzigen Wert zum Inhalt haben.) Solange Namen lediglich vereinbart, aber noch nicht definiert sind, ist ihr Inhalt unbestimmt. Wird eine noch nicht definierte Variable verwendet, erscheint die Fehlermeldung "Das Makro *Makroname* ist nicht definiert".

Ein lokaler Name gilt in dem Makro, in dem er vereinbart wurde und in allen in einer Aufrufkette hierarchisch tieferstehenden Makros, sofern er dort nicht erneut als lokal vereinbart wird. Er ist in sonstigen Makros oder auf Systemebene nicht bekannt. Wenn z.B. ein lokaler Name in Makro M1 vereinbart wird, Makro M1 Makro M2 aufruft, Makro M2 Makro M3 aufruft, Makro M3 Makro M4 aufruft, und wenn in den aufgerufenen Makros der Name nicht erneut als lokal vereinbart wird, ist der Name in den Makros M1, M2, M3 und M4 bekannt und in allen sonstigen Makros sowie auf Systemebene unbekannt.

Wird ein lokaler Name in einem aufgerufenen Makro erneut als lokaler Name vereinbart, so wird dadurch dessen ursprünglicher Inhalt verdeckt, aber nicht überschrieben. Der ursprüngliche Inhalt steht nach dem Verlassen des aufgerufenen Makros wieder zur Verfügung. Lokale Namen können dadurch in unterschiedlichen Makros unterschiedliche Inhalte haben. ME10 kennt neben lokalen auch globale Namen. Ein globaler Name entsteht durch Definition ohne vorherige Vereinbarung. Er kann in allen Makros und auch beim interaktiven Arbeiten verwendet und verändert werden.

Beispiel 1

Im nachstehend aufgelisteten Makro H_mac werden den lokalen Variablen X1 und X2 die Werte 1 und 2 und den globalen Variablen X3 und X4 die Werte 3 und 4 zugewiesen. Diese Werte werden danach von H_mac in der Hinweiszeile ausgegeben. Dann erfolgt der Aufruf des Untermakros U_mac. In U_mac werden lokale Variablen X1 und X3 vereinbart und mit den Werten 100 und 300 belegt. X1 und X3 verdecken in U_mac die lokale Variable X1 des Makros H_mac und die globale Variable X3. Sie haben dadurch in U_mac einen anderen Inhalt als in H_mac. Die lokale Variable X2 des aufrufenden Makros H_mac und die globale Variable X4 sind in U_mac bekannt und können dort benutzt und verändert werden. Die DISPLAY-Anweisungen des Untermakros bewirken dadurch folgende Ausgaben in der Hinweiszeile:

```
Ausgabe  |Bedeutung
---------+-----------------------------------------------------------
X1 = 100 |Lokale Variable in U_mac, verdeckt X1 aus H_mac
X2 = 2   |Lokale Variable in H_mac, in U_mac bekannt
X3 = 300 |Lokale Variable in U_mac, verdeckt die globale Variable X3
X4 = 4   |Globale Variable, in allen Makros bekannt
```

Danach wird in U_mac der lokalen Variablen X2 des Makros H_mac der Wert 200 und der globalen Variablen X4 der Wert 400 zugewiesen. Die auf den Untermakroaufruf folgenden Display-Anweisungen des Makros H_mac bewirken dann folgende Ausgaben:

```
Ausgabe  |Bedeutung
---------+-----------------------------------------------
X1 = 1   |Ursprünglicher Wert von X1, nun wieder sichtbar
X2 = 200 |X2 hat in U_mac einen neuen Wert erhalten
X3 = 3   |Ursprünglicher Wert von X3, nun wieder sichtbar
X4 = 400 |X4 hat in U_mac einen neuen Wert erhalten
```

```
DEFINE H_mac
  LOCAL X1
  LOCAL X2
  LET X1 1
  LET X2 2
  LET X3 3
  LET X4 4
  DISPLAY ('X1 = ' + STR X1)
  DISPLAY ('X2 = ' + STR X2)
  DISPLAY ('X3 = ' + STR X3)
  DISPLAY ('X4 = ' + STR X4)
  U_mac
  DISPLAY ('X1 = ' + STR X1)
  DISPLAY ('X2 = ' + STR X2)
  DISPLAY ('X3 = ' + STR X3)
  DISPLAY ('X4 = ' + STR X4)
END_DEFINE
```

```
DEFINE U_mac
  LOCAL X1
  LOCAL X3
  LET X1 100
  LET X3 300
  DISPLAY ('X1 = ' + STR X1)
  DISPLAY ('X2 = ' + STR X2)
  DISPLAY ('X3 = ' + STR X3)
  DISPLAY ('X4 = ' + STR X4)
  LET X2 200
  LET X4 400
END_DEFINE
```

Beispiel 2

In Makro Mainmac_1 wird Locmac_1 als lokaler Name vereinbart. Danach wird in Mainmac_1 das lokale Makro Locmac_1 definiert und aufgerufen. In der Hinweiszeile erscheint die Meldung "Locmac_1 aufgerufen". Locmac_1 läßt sich nur von Mainmac_1 aufrufen, in anderen Makros oder auf Systemebene ist es nicht bekannt. Mainmac_2 unterscheidet sich von Mainmac_1 dadurch, daß Locmac_2 nicht als lokales Makro vereinbart wird. Locmac_2 ist ein globales Makro und kann daher auch von anderen Makros und vom Benutzer aufgerufen werden.

```
DEFINE Mainmac_1
  LOCAL Locmac_1
  DEFINE Locmac_1
    DISPLAY 'Locmac_1 aufgerufen!'
   END_DEFINE
  Locmac_1
END_DEFINE
```

```
DEFINE Mainmac_2
  DEFINE Locmac_2
    DISPLAY 'Locmac_2 aufgerufen!'
  END_DEFINE
  Locmac_2
 END_DEFINE
```

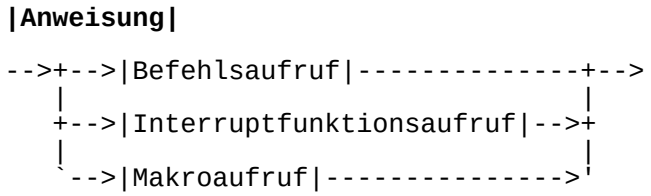
4.2 Anweisungen

4.2.1 Anweisungsarten und Syntax

Anweisungen bilden den Hauptteil eines Makros. Sie bewirken das, was der Anwender eigentlich vom Makro erwartet. In Makros sind mit Ausnahme von INPUT und TINPUT alle ME10-Befehle und ME10-Interruptfunktionen zulässig. Die Erfahrung zeigt allerdings, daß die meisten Befehle und Interruptfunktionen auf die Bedürfnisse des interaktiven Arbeitens ausgerichtet sind und daß nur ein relativ geringer Teil davon in Makros benötigt wird. Eine vollständige Beschreibung aller Befehle und Interruptfunktionen befindet sich im ME10-Systemhandbuch. In den folgenden Kapiteln werden die beim interaktiven Arbeiten verwendeten Befehle und Interruptfunktionen als bekannt vorausgesetzt und daher in der Regel nicht beschrieben. Der Schwerpunkt liegt auf den bei interaktiven Arbeiten normalerweise nicht verwendeten Anweisungen. Sie können in folgende Gruppen unterteilt werden:

- Zuweisungen an Variablen
- Dialoganweisungen
- Steueranweisungen
- Abfrageanweisungen
- Dateioperationen
- Anweisungen zur Systemanpassung
- Anweisungen zum Arbeiten mit Tabellen

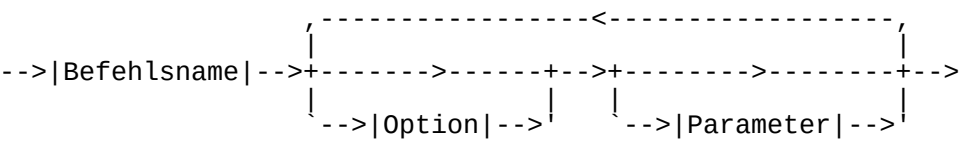
Anweisungen müssen entweder ME10-Befehle, ME10-Interruptfunktionen oder Makros aufrufen.



Befehle haben höchste Priorität und brechen jede andere Systemaktion ab. Sie werden, sofern dies einen Sinn ergibt, nach ihrer Ausführung solange wieder angeboten, bis ein anderer Befehl aufgerufen oder die BREAK- bzw. die ESC-Taste betätigt wird. (Diese Tasten rufen den ME10-Befehl CANCEL auf). Interruptfunktionen führen zu keinem Abbruch, sondern zu einem Unterbrechen einer gerade aktiven Systemaktion. Die unterbrochene Systemaktion wird nach dem Abarbeiten der Interruptfunktion fortgesetzt. Interruptfunktionen werden nach ihrem Aufruf nur einmal abgearbeitet, also nicht zur wiederholten Ausführung angeboten. Makros brechen andere Systemaktionen ab, wenn in Ihnen Befehle aufgerufen werden und unterbrechen die Systemaktionen, wenn sie Interrupfunktionen abarbeiten.

Befehle, Interruptfunktionen und Makros werden wie folgt aufgerufen:

|Befehlsaufruf|

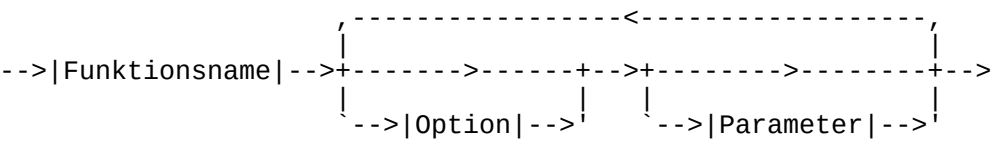


Beispiel

```
LINE RECTANGLE 100,100 150,200
```

```
LINE      : Schlüsselwort für ME10-Befehl zum Zeichnen einer Linie
RECTANGLE : OPTION für Sonderform Rechteck
100,100   : Erster Parameter = Koordinaten der linken unteren Ecke
150,200   : Zweiter Parameter = Koordinaten der rechten oberen
            Ecke
```

|Interruptfunktionsaufruf|

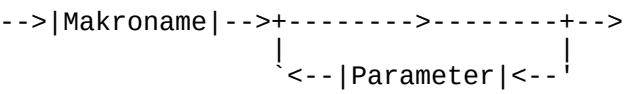


Beispiel

```
WINDOW ZOOM 1.5
```

```
WINDOW : Schlüsselwort für Interruptfunktion zum Verändern des
         Bildausschnitts
ZOOM   : OPTION für Zoomen
1.5    : Parameter = Zoomfaktor
```

|Untermakroaufruf|



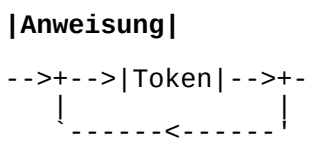
Beispiel

```
Addiere 7 5
```

```
Addiere : Name des Makros
7       : Erster Übergabeparameter
5       : Zweiter Übergabeparameter
```

4.2.2 Formaler Aufbau

Anweisungen werden vom Kommandointerpreter zeichenweise gelesen. Sobald ein oder mehrere unmittelbar aufeinanderfolgende Zeichen einen Sinn ergeben, wird daraus ein Anweisungselement, ein sogenanntes Token, gebildet. Ein Token ist z.B. die Zeichenfolge LINE, es wird vom Kommandointerpreter als Aufruf des ME10-Befehls "LINE" interpretiert. Vollständige Anweisungen können aus einem einzigen oder aus mehreren aufeinanderfolgenden Token bestehen. Die Anweisung zum Zeichnen eines Kreises "CIRCLE 100,100 50" besteht beispielsweise aus den Token "CIRCLE", "100,100" und "50".



Anweisungselemente werden vom Kommandointerpreter normalerweise von links beginnend solange zusammengefaßt, bis sie eine vollständige Anweisung bilden. Manchmal ist jedoch diese Reihenfolge des Zusammenfassens nicht gewünscht. Nachfolgendes Beispiel soll dies erläutern.

Ein Kreis soll bei x/y = 20/50 mit einem Radius r = 15*sin 30° gezeichnet werden. Der Programmierer schreibt dazu folgende Anweisung:

```
CIRCLE 20,50 15 * SIN 30
```

ME10 interpretiert die Anweisung wie folgt:

```
Anweisungselement 1 : CIRCLE -->  Aufruf des Befehls "Kreis"
Anweisungselement 2 : 20,50  -->  Kreismittelpunkt bei x = 20 und
                      y = 50
Anweisungselement 3 : 15     -->  Radius 15
```

Damit liegt bereits eine vollständige Anweisung vor. Es wird ein Kreis mit Radius 15 gezeichnet und der Befehl "Kreis" bleibt aktiv. Das System erwartet nun entweder die Bestimmung eines neuen Mittelpunkts oder die Angabe eines neuen Radius. Die restlichen Zeichen der Anweisung lauten aber "* SIN 30". Das System kann mit diesen Angaben nichts anfangen und antwortet daher mit einer Fehlermeldung.

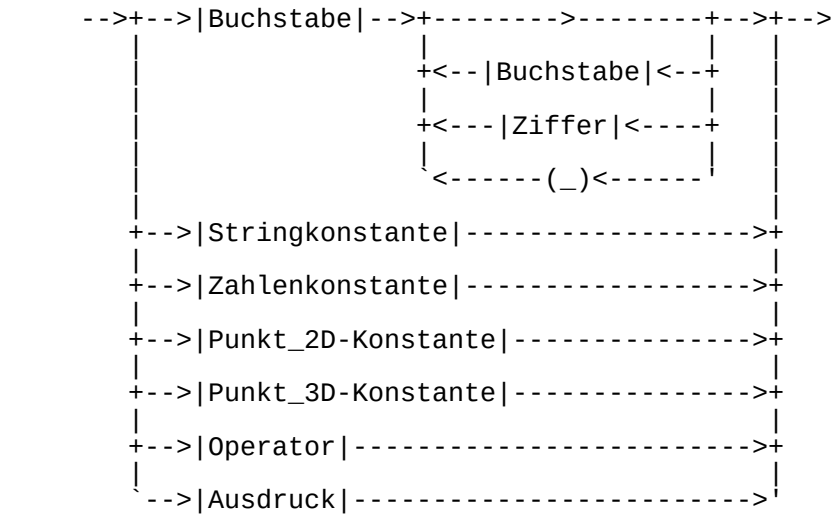
ME10 bietet deshalb die Möglichkeit, die Reihenfolge bei der Auswertung von Anweisungen mit Hilfe von Klammern zu steuern. Das Verfahren ist im Prinzip von der Algebra bekannt. Findet der Kommandointerpreter ein Paar runder Klammern, so wird das Zusammenfassen der Token unterbrochen und zunächst der Klammerinhalt ausgewertet. Wenn das Ergebnis einem in formaler und semantischer Hinsicht zulässigen Token entspricht, wird die Auswertung der Anweisung mit diesem resultierenden Token fortgesetzt. Klammern können dabei in beliebiger Tiefe geschachtelt werden. Sie müssen stets vollständig geschlossen sein, d.h. die Anzahl der öffnenden Klammern muß der Anzahl der schließenden Klammern entsprechen.

Im Beispiel hätte die korrekte Anweisung wie folgt lauten müssen:

```
CIRCLE 20,50 (15 * sin 30)
Token 1 : CIRCLE; Token 2 : 20,50; Token 3 : (15 * SIN 30) = 7.5
```

Nachfolgendes Syntaxdiagramm zeigt, wie ein Token formal aufgebaut sein muß. Das heißt aber nicht, daß jede damit gebildete Zeichenfolge im konkreten Fall auch ein zulässiges Token ist.

|Token|



Beispiele

Token	Erläuterung
H_mac	Makroname
X_wert	Variablenname
LINE	ME10-Befehl
RECTANGLE	ME10-Option zu LINE
SIN	Integrierte Funktion "Sinus"
'A = '	Stringkonstante
15.7	Zahlenkonstante
100,150	Punkt_2D-Konstante
20,40,50	Punkt_3D-Konstante
+	Additionsoperator
(5 * B2)	Arithmetischer Ausdruck

Kein zulässiges Token wäre	Begründung	Zulässig wäre
H mac als Makroname	Leerzeichen unzulässig	Hmac oder H_mac
X-wert als Variablenname	Minuszeichen wird als Operator interpretiert	Xwert oder X_wert
100 , 50 als Punktkonstante	Leerzeichen unzulässig	100,50
B - C als arithm.Ausdruck	Ausdrücke müssen geklammert sein	(B - C)
SIN 45 als trigonom.Ausdr.	Ausdrücke müssen geklammert sein	(sin 45)

5 Zuweisungen an Variablen

5.1 Allgemeines zur Verwendung von Variablen

ME10 kennt keine Variablen in dem bei anderen Programmiersprachen üblichen Sinne. Werte werden vielmehr generell in Form von Makros gespeichert. Wenn im folgenden doch von Variablen die Rede ist, so hat das rein didaktische Gründe. Unter einer Variablen soll ein Makro verstanden werden, das einen einzigen Wert zum Inhalt hat.

ME10 unterscheidet lokale und globale Variablen. Globale Variablen ermöglichen zwar auf sehr einfache Weise, Daten in verschiedenen Makros gemeinsam zu nutzen, führen aber zu sehr unübersichtlichen und unstrukturierten Programmen. Sie sollten, wenn irgend möglich, vermieden werden. Für Größen, die nur in einem Makro benötigt werden, sind generell lokale Variablen vorzuziehen. Auf diese Weise ist es möglich, umfangreiche, aus vielen Makros bestehende Programme zu schreiben, ohne Gefahr zu laufen, daß es zu Fehlern aufgrund zufällig gleicher Variablennamen in unterschiedlichen Programmoduln kommt. Wenn Daten in mehreren Makros verwendet werden sollen, sind globale Variablen zur Not vertretbar, aber nicht unbedingt notwendig, wie im Kapitel "Untermakroaufruf" noch gezeigt wird.

ME10 kennt leider nur einfache und keine indizierten Variablen. Es können also keine Felder vereinbart werden. Ab ME10 Version 4 besteht die Möglichkeit, zweidimensionale logische Tabellen einzurichten. Sie können auch als Felder verwendet werden. Felder mit drei und mehr Dimensionen sind aber nach wie vor nicht möglich. Einige technische Berechnungen können dadurch nur sehr umständlich programmiert werden. Dieser Nachteil wird aber durch die Möglichkeit ausgeglichen, beliebige, in einer anderen Programmiersprache geschriebene Programme vom Makro aufzurufen.

5.2 Zuweisung mit LET

LET (Interruptfunktion)

```
-->(LET)-->|Variablenname|-->|Token|-->
```

Ein dem Variablennamen folgendes Anweisungselement (Token) wird ausgewertet und der durch den Namen gekennzeichneten Variablen zugewiesen. Der Begriff Token wurde in Kapitel 4 näher beschrieben. Normalerweise handelt es sich hierbei um eine Konstante oder einen Ausdruck.

Beispiele

```
LET X 7          {weist X den Wert 7 zu}
LET Y (x + 5)    {weist Y den Wert x+5 zu, hier also 12}
LET Z CIRCLE     {weist Z den ME10-Befehl CIRCLE zu}
```

Ausdrücke müssen in Klammern stehen, wenn sie nicht lediglich aus einem einzigen Wert (Zahlenwert, Vektor, String) bestehen.

Wie oben erwähnt, besteht kein prinzipieller Unterschied zwischen einem Makro und einer Variablen. Dies soll an einem kleinen Beispiel verdeutlicht werden. Die Wertzuweisung

```
LET X 7
```

führt dazu, daß ein Makro mit dem Inhalt "7" definiert wird. Man kann sich das Makro mit dem Texteditor anschauen. Bei Eingabe von "EDIT_MACRO X" wird folgendes Makro am Textbildschirm angezeigt:

```
DEFINE X
 7
END_DEFINE
```

Eine Wertzuweisung an eine Variable kann daher auch wie folgt erfolgen:

```
DEFINE Variablennname
 Token
END_DEFINE
```

5.3 Weitere Formen der Zuweisung

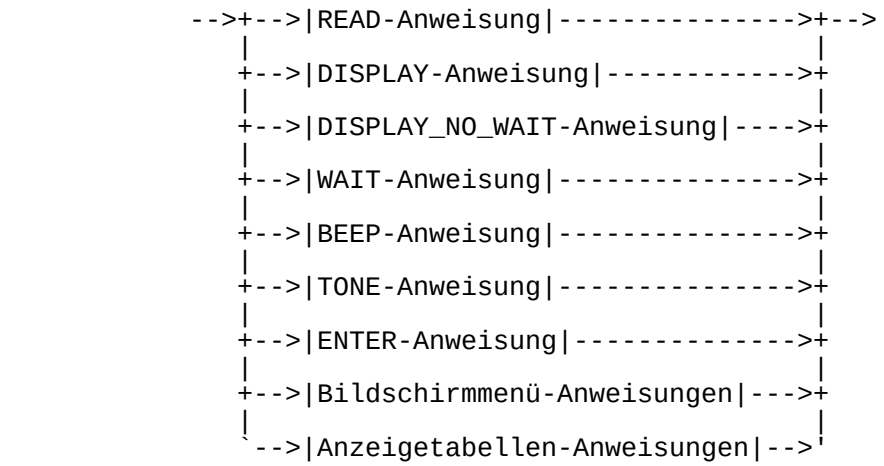
Zuweisungen an Variablen können weiterhin bei bestimmten Dialoganweisungen und Dateioperationen erfolgen.

6 Dialoganweisungen

Dialoganweisungen ermöglichen eine Ausgabe von Information durch das System und eine Eingabe von Daten durch den Benutzer. Bei der vom System auszugebenden Information handelt es sich um Text, um akustische Signale oder um Piktogramme. Für die Ausgabe stehen die Hinweiszeile, ein Tongenerator, Bildschirmmenüs und Anzeigetabellen zur Verfügung. Benutzereingaben können über die Eingabezeile, über das Menüsystem, über Anzeigetabellen oder durch Digitalisierung auf der Zeichenfläche erfolgen. Ein Maskendialog, wie er bei einigen anderen CAD-Programmen stattfindet, ist standardmäßig nicht vorgesehen. Er könnte jedoch durch Einbinden von Programmen in anderen Programmiersprachen realisiert werden.

In diesem Kapitel werden die Ausgabe von Meldungen in der Hinweiszeile, die Ausgabe von akustischen Meldungen sowie Benutzereingaben über die Eingabezeile und durch Digitalisierung beschrieben. Auf die Verwendung von Bildschirmmenüs und Anzeigetabellen für Dialog wird in den Kapiteln "Systemanpassung" und "Tabellen" eingegangen.

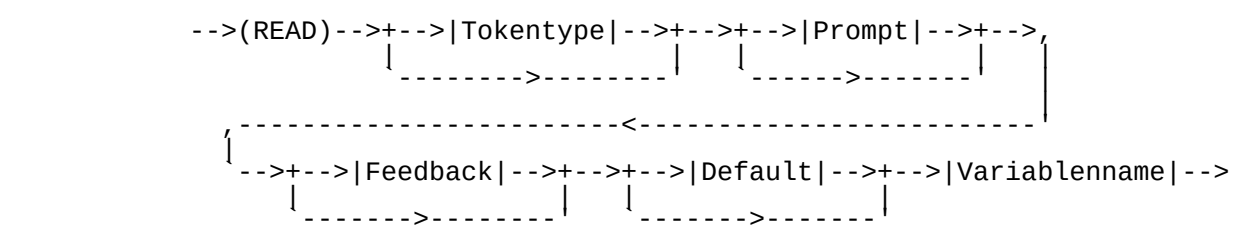
|Dialoganweisung|



6.1 READ

READ dient zum Anfordern und Einlesen von Benutzereingaben. Die Makroausführung wird nach Ausgabe des Anforderungstextes unterbrochen und erst nach der Benutzereingabe fortgesetzt.

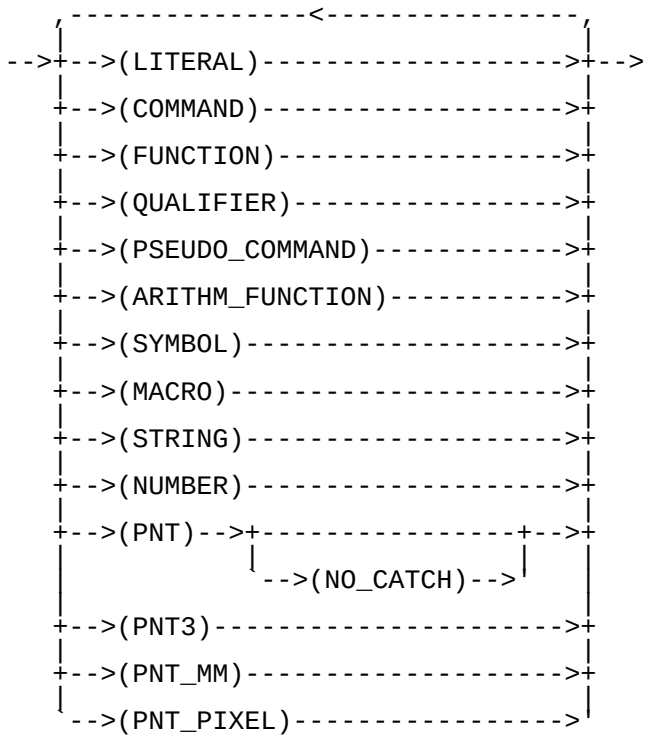
|READ| (Interruptfunktion)



Parameter "Tokentyp"

Tokentyp spezifiziert die für die Benutzereingabe zugelassenen Datentypen. Folgende Tokentypen können angegeben werden:

|Tokentyp|



Tokentyp	Zugelassener Datentyp
Keine Angabe	Text, Zahl, 2D-Vektor, 3D-Vektor
LITERAL	Alle (Benutzereingabe wird ohne Prüfung übernommen)
COMMAND	ME10-Befehle
FUNCTION	ME10-Interruptfunktion
QUALIFIER	Option
PSEUDO_COMMAND	Pseudokommando, z.B. REPEAT, UNTIL, WHILE, END_WHILE
ARITHM_FUNCTION	Integrierte Funktion (z.B. ABS, SIN, ANG, CHR, DATE)
SYMBOL	Operator-Symbole + - * / ** ^ < <= = >= <>
MACRO	Makro
STRING	Text
NUMBER	Zahl

(Tabelle wird fortgesetzt)

Tokentyp	Zugelassener Datentyp
PNT	2D-Vektor in akt.Einheiten, bezogen auf d.Koordinatenursprung NO_CATCH schaltet die automatische Fangfunktion aus
PNT3	3D-Vektor *) in akt.Einheiten, bezogen auf d.Koordinatenursp.
PNT_MM	2D-Vektor in mm, bezogen auf die linke untere Ecke der aktiven Tablettfläche
PNT_PIXEL	2D-Vektor in Pixel, bezogen auf die linke untere Ecke des aktuellen Fensters

*) Ein 3D-Vektor ist ein Punkt mit X-, Y- und Z-Koordinatenwerten. Er kann nur explizit eingegeben, nicht digitalisiert werden.

Ohne Angabe eines Tokentyps werden Texte, Zahlen, 2D-Vektoren und 3D-Vektoren als Benutzereingabe akzeptiert und der Variablen zugewiesen. Andere Datentypen führen entweder zu einem Abbruch des Makros oder zu einer Wiederholung der READ-Anweisung. Nachstehende Tabelle zeigt die Wirkung aller möglichen Benutzereingaben, wenn kein Tokentyp angegeben wird.

Datentyp der Eingabe	Wirkung, wenn kein Tokentyp spezifiziert wurde
Befehl	Abbruch des Makros und Ausführung des Befehls
Interruptfunktion	Ausführung der Interruptfunktion und Wiederholung der READ-Anweisung
Makro	Ausführung des eingegebenen Makros, danach Abbruch des aktuellen Makros, wenn das durch die Eingabe aufgerufene Makro einen Befehl verwendet, ansonsten Wiederholung der READ-Anweisung.
Text Zahlenwert 2D- u. 3D-Vektor	Das Ergebnis wird der Variablen der READ-Anweisung zugewiesen
Sonstiger Datentyp	Warnton und Wiederholung der READ-Anweisung

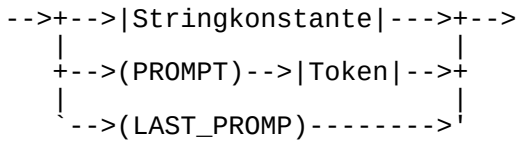
Bei Angabe eines oder mehrerer Tokentypen werden nur die dadurch spezifizierten Datentypen akzeptiert und der Variablen zugewiesen. Andere Datentypen führen entweder zu einem Abbruch des Makros oder zu einer Wiederholung der READ-Anweisung. Durch diese Möglichkeit lassen sich Eingabefehler, die zu unerwünschten Ergebnissen oder zum Programmabbruch führen können, vermeiden. Nachstehende Tabelle zeigt die Wirkung aller möglichen Benutzereingaben, wenn ein Datentyp spezifiziert wurde, die Eingabe aber einen anderen Datentyp aufweist.

Datentyp der Eingabe	Wirkung, wenn ein anderer Datentyp spezifiziert wurde
Befehl	Abbruch des Makros und Ausführung des Befehls
Interruptfunktion	Ausführung der Interruptfunktion und Wiederholung der READ-Anweisung
Makro	Ausführung des eingegebenen Makros, danach Abbruch des aktuellen Makros, wenn das durch die Eingabe aufgerufene Makro einen Befehl verwendet, ansonsten Wiederholung der READ-Anweisung.
Sonstiger Datentyp	Warnton und Wiederholung der READ-Anweisung

Parameter "Prompt"

Prompt bezeichnet die Eingabeanforderung durch das System. Normalerweise steht hier eine Textkonstante wie z.B. 'Bitte Wert eingeben'. Es sind jedoch alle Arten von Token wie z.B. Textvariablen zulässig. In diesen Fällen muß aus Gründen der Eindeutigkeit der Eingabeanforderung das Schlüsselwort PROMPT vorangestellt werden.

|Prompt|



Beispiel: In einem Makro soll ein Kreis mit Radius 5 mm an einer vom Benutzer zu bestimmenden Stelle der Zeichnung gezeichnet werden. Hierzu könnten folgende Anweisungen dienen:

```
READ PNT 'Bitte Position für Kreis bestimmen' X
CIRCLE X 5
```

Eine solche Lösung hätte allerdings den Nachteil, sprachabhängig zu sein. Wenn das Makro später in einem anderssprachigen Land verwendet werden soll, müßten alle Eingabeanforderungen übersetzt werden. Folgende Lösung wäre hier flexibler:

```
READ PNT PROMPT Satz1 X
CIRCLE X 5
```

Der Variablen Satz1 müßte vorher ein entsprechender Text zugewiesen worden sein. Für Östereich würde sich zum Beispiel folgender Text empfehlen:

```
LET Satz1 'Küß die Hand. Wo soll bitteschön der Kreis hin?'
```

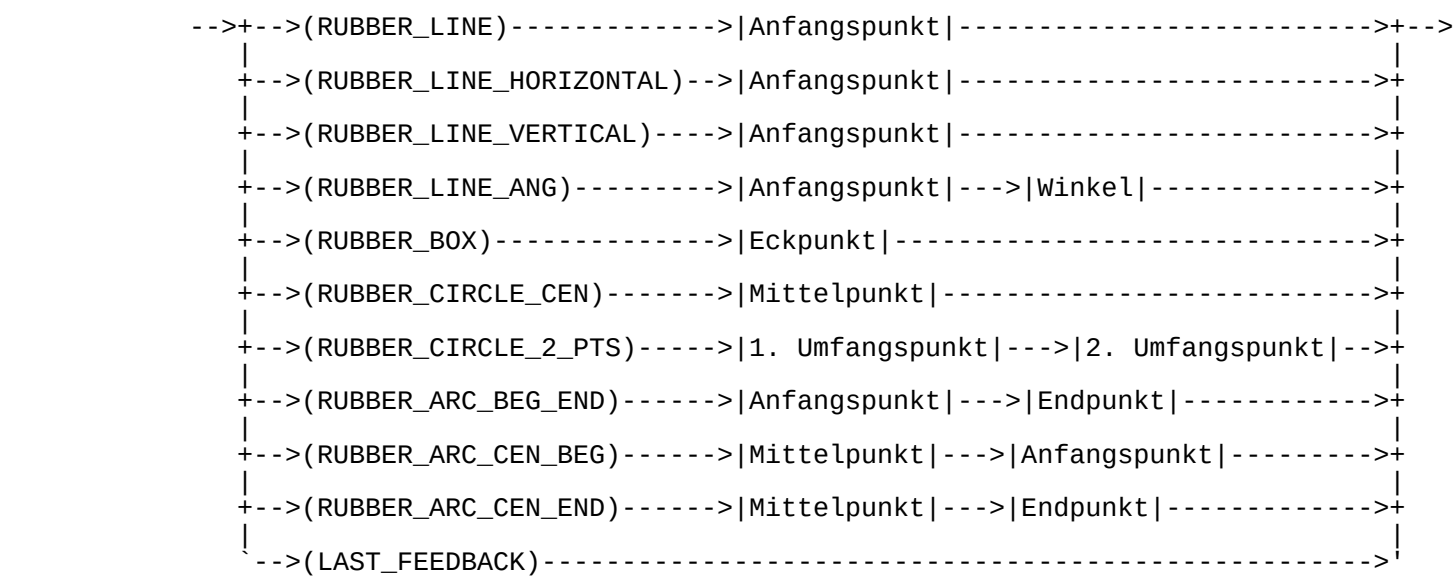
Der Text könnte sich auch zusammen mit weiteren Eingabeanforderungstexten in einer sprachspezifischen Datei befinden und mit Hilfe der READ_FILE-Anweisung der Variablen "Satz1" zugewiesen werden.

Bei Verwendung der Option LAST_PROMPT wird die vor dem Aufruf der READ-Funktion aktive Eingabeaufforderung des Systems als Eingabeaufforderung der READ-Funktion verwendet.

Parameter "Feedback"

Punktbestimmungen werden häufig durch die vom interaktiven Arbeiten bekannte Gummibandfunktion erleichtert. Die Gummibandfunktion bewirkt z.B. beim Befehl LINE, daß der Cursor wie durch ein Gummiband mit dem Anfangspunkt der Linie verbunden bleibt. Durch Angabe eines Feedbacks kann bestimmt werden, welche Art von Gummiband verwendet und von welchem Punkt aus das Gummiband gezogen wird.

|Feedback|

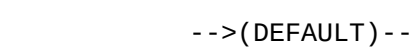


Bei Verwendung der Option LAST_FEEDBACK wird die zum Zeitpunkt des Aufrufs der READ-Funktion aktive Feedbackfunktion des Systems benutzt. Bei nichtaktiver Feedbackfunktion des Systems kommt es zu einer Fehlermeldung.

Parameter "Default"

Default ermöglicht die Ausgabe eines Standardwertes in der Eingabezeile. Der Standardwert kann durch Betätigung der Eingabetaste unverändert übernommen oder mit Hilfe der Editiertasten verändert werden.

|Default|



Normalerweise werden Texte, Zahlenwerte oder Punkte als Standardwert verwendet. Es sind jedoch alle Arten von Anweisungselementen (Token) zulässig.

Beispiel: Der Benutzer wird aufgefordert, einen Punkt zu bestimmen. Alternativ wird der Koordinatenursprung angeboten:

```
READ PNT 'Bitte Punkt bestimmen' DEFAULT 0,0 P1
```

Parameter "Variablenname"

Variablenname bezeichnet die Variable, der die Benutzereingabe zugewiesen werden soll.

Anwendungsbeispiel

Makro "Vierkant" ermöglicht das Zeichnen eines Rechtecks. Die erste Ecke wird durch Eingabe eines Punkts bestimmt, die zweite Ecke wahlweise durch Eingabe eines Punkts oder durch Eingabe von Winkel und Länge der Diagonalen. In der IF-Abfrage erfolgt eine Prüfung des Datentyps der Eingabe mit Hilfe der integrierten Funktion TYPE. Wenn der Datentyp NUMBER (Zahl) vorliegt, wird die Eingabe als Winkel interpretiert, die Länge abgefragt und der zweite Eckpunkt mit Hilfe von Vektoroperationen berechnet. Danach wird das Rechteck gezeichnet. (IF-Abfrage, integrierte Funktionen und Vektoroperationen werden in späteren Kapiteln noch ausführlich behandelt.)

```
DEFINE Vierkant
  LOCAL P1
  LOCAL P2
  LOCAL L_dia
  READ PNT 'Erster Eckpunkt?' P1
  READ PNT NUMBER 'Zweiter Eckpunkt? (Punkt bestimmen oder Winkel eingeben)'
  RUBBER_BOX P1 P2
  IF (TYPE P2 = NUMBER)
    READ NUMBER 'Länge der Diagonalen?' L_dia
    LET P2 (P1 + (PNT_RA L_dia P2))
  END_IF
  LINE RECTANGLE P1 P2
  END
END_DEFINE
```

6.2 DISPLAY, DISPLAY_NO_WAIT, WAIT

DISPLAY ermöglicht eine Ausgabe von Meldungen in der Hinweiszeile.

|DISPLAY| (Interruptfunktion)

-->(DISPLAY)-->|Token|-->

DISPLAY wertet das nachfolgende Anweisungslement (Token) aus, zeigt das Ergebnis in der Hinweiszeile an und fordert den Benutzer auf, eine beliebige Taste oder den Tablettstift zu betätigen. Bei DISPLAY_NO_WAIT hingegen wird der Makrolauf nach der Ausgabe der Meldung selbständig fortgesetzt.

|DISPLAY_NO_WAIT| (Interruptfunktion)

-->(DISPLAY_NO_WAIT)-->|Token|-->

Bei DISPLAY_NO_WAIT wird wie bei DISPLAY das nachfolgende Anweisungselement (Token) ausgewertet und in der Hinweiszeile angezeigt. Das System wartet jedoch nicht auf eine Reaktion des Benutzers, sondern überschreibt die Meldung mit der nächsten Eingabeaufforderung. DISPLAY_NO_WAIT ist daher nur sinnvoll, wenn die nächste Eingabeaufforderung nicht sofort erfolgt. Ein typischer Anwendungsfall besteht darin, den

Anwender bei längerdauernden Systemaktionen ohne entsprechende Meldung über deren Durchführung zu informieren.

Beispiel: Vor dem Start einer längeren Berechnung kann folgende Anweisung sinnvoll sein:

```
DISPLAY_NO_WAIT 'Führe Berechnung durch. Bitte etwas Geduld.'
```

Um dem Benutzer genügend Zeit zum Lesen einer der Meldung zu geben, kann auch eine WAIT-Anweisung oder eine TONE-Anweisung folgen.

|WAIT| (Interruptfunktion)

```
-->(WAIT)-->|Zahl|-->
```

WAIT veranlaßt das System, während des in Sekunden angegebenen Zeitraums zu warten.

6.3 ENTER

ENTER wertet ein Anweisungselement aus und trägt das Ergebnis in die Eingabezeile ein.

|ENTER| (Interruptfunktion)

```
-->(ENTER)-->|Token|-->
```

Der Benutzer kann den Eintrag durch Betätigung der Enter-Taste unverändert übernehmen, mit Hilfe der Editiertasten verändern oder durch Betätigung der ESCAPE-Taste verwerfen. Bei dem Anweisungselement handelt es sich häufig um einen Ausdruck, um einen Befehl oder um einen Makronamen. Gegenüber der vorher beschriebenen Dialogfunktion "DISPLAY" besteht ein wesentlicher Unterschied. DISPLAY führt lediglich zu einer Ausgabe des Anweisungselements in der Hinweiszeile. Bei Betätigung der Enter-Taste erfolgt keine Übernahme als Eingabe, sondern lediglich ein Löschen der Anzeige. Das mit ENTER in die Eingabezeile eingetragene Anweisungselement kann hingegen unmittelbar als Eingabe übernommen werden.

Beispiel: Nachstehende Anweisung gibt den Befehlsnamen LINE in der Eingabezeile aus. Eine Betätigung der Entertaste führt zu einem Aufruf des Befehls.

```
ENTER LINE
```

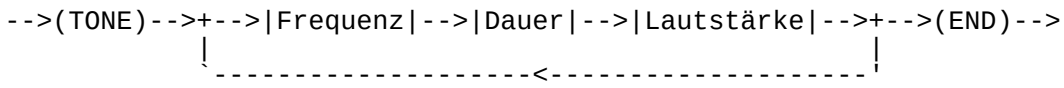
6.4 BEEP, TONE

BEEP erzeugt ein akustisches Signal.

|BEEP| (Interruptfunktion)

TONE erzeugt akustische Signale nach bestimmten Parametern. Die Frequenz wird in Hertz, die Dauer in Sekunden und die Lautstärke im Bereich von 0 bis 1 angegeben. Wirkung, Bereich und Auflösung der Parameter hängen von der jeweils verwendeten Hardware ab.

|TONE| (Interruptfunktion)



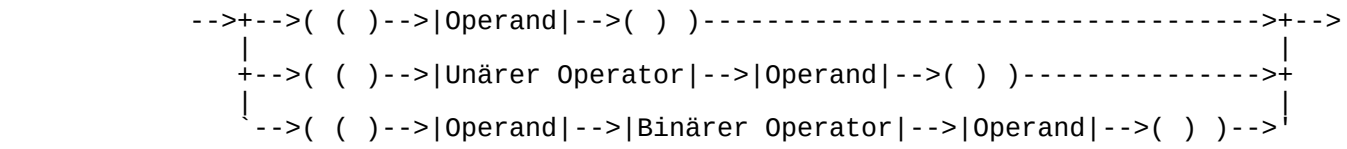
Beispiel: Die Anweisung "TONE 800 2 0.5 END" erzeugt einen Ton mit 800 Hertz, 2 Sekunden Dauer und Lautstärke 0,5.

7 Ausdrücke, Operatoren und Operanden

7.1 Aufbau von Ausdrücken

Kombinationen von Operatoren und Operanden, die bei einer Auswertung einen Wert ergeben, werden als Ausdrücke bezeichnet. Ausdrücke müssen grundsätzlich in Klammern stehen. Sie können wie folgt aufgebaut sein:

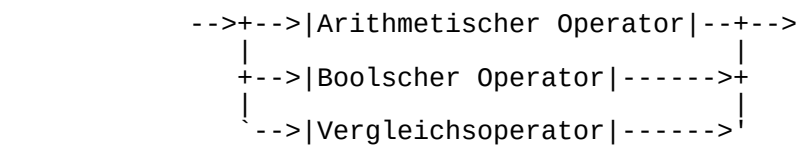
|Ausdruck|



7.2 Operatoren

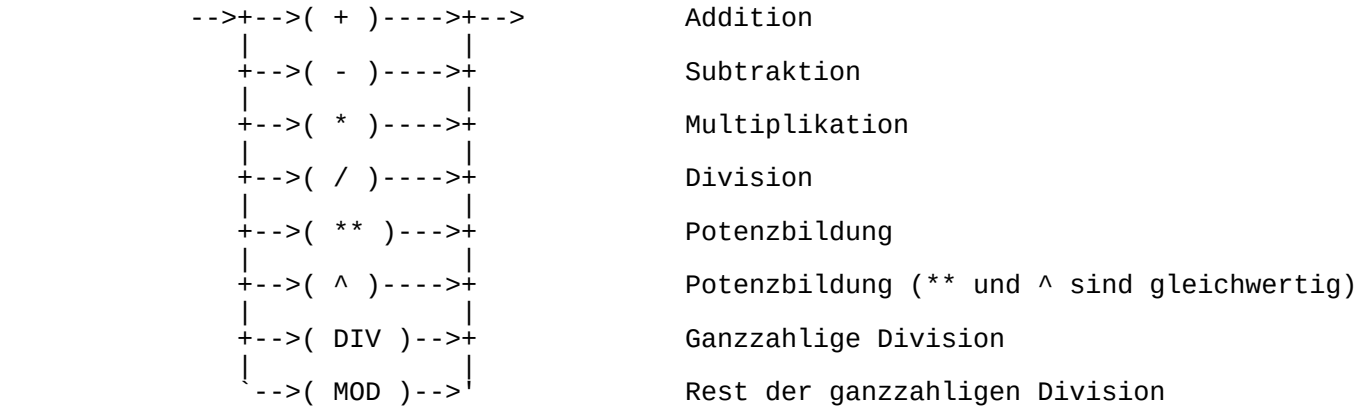
ME10 kennt arithmetische Operatoren, boolsche Operatoren und Vergleichsoperatoren.

|Operator|



7.2.1 Arithmetische Operatoren

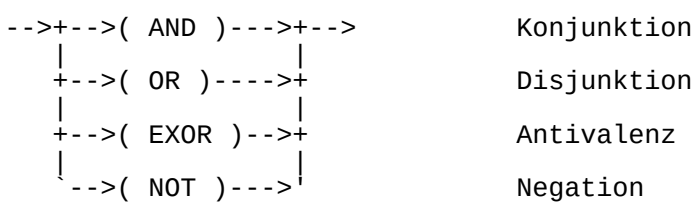
|Arithmetischer Operator|



Operator	Verwendung (Z = Zahl; V = Vektor (= Punkt, Punkt_3D); S = String)
+	Identität, Addition von Zahlen und Vektoren, Verkettung von Strings +Token --> Token; Z + Z --> Z; V + V --> V; S + S --> S
-	Negation, Subtraktion von Zahlen und Vektoren -Z --> Z * (-1); -V --> V * (-1); Z - Z --> Z; V - V --> V
*	Multiplikation von Zahlen, Skalierung von Vektoren Z * Z --> Z; Z * V --> V; V * Z --> V;
/	Division von Zahlen, Skalierung von Vektoren Z / Z --> Z; V / Z --> V;
** bzw.^	Potenzbildung für Zahlen (Basis muß positiv sein) Z ** Z --> Z; Z ^ Z --> Z
DIV	Division ohne Rest für Zahlen Z DIV Z --> Z
MOD	Rest der ganzzahligen Division von Zahlen Z MOD Z --> Z

7.2.2 Boolsche Operatoren

|Boolscher Operator|



ME10 beherrscht die boolschen Operationen Konjunktion, Disjunktion, Antivalenz und Negation. Als boolsche Operanden finden Zahlen Verwendung. Hierbei gilt folgende Vereinbarung:

```
Zahl gleich 0   : logisch unwahr
Zahl ungleich 0 : logisch wahr
```

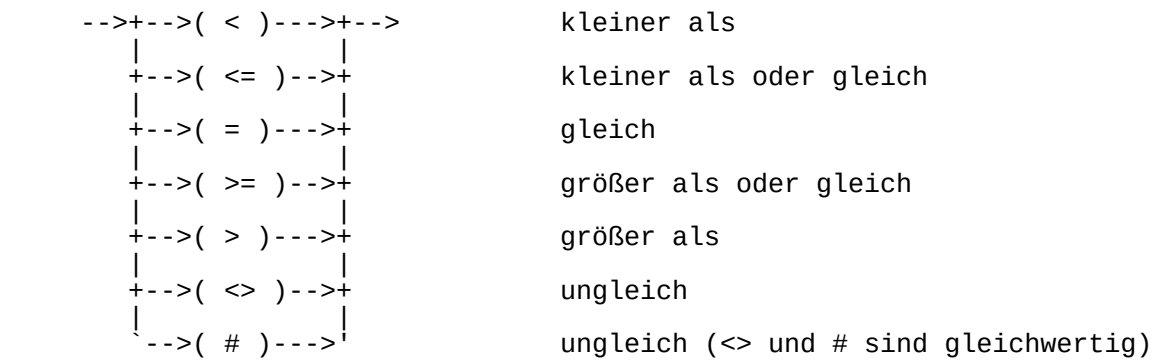
Die symbolischen Konstanten TRUE und FALSE sind mit den Zahlenwerten 1 und 0 vorbesetzt.

Rechenregeln (0 = unwahr, 1 = wahr):

a	b	a AND b	a OR b	a EXOR b	NOT a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

7.2.3 Vergleichsoperatoren

|Vergleichsoperator|



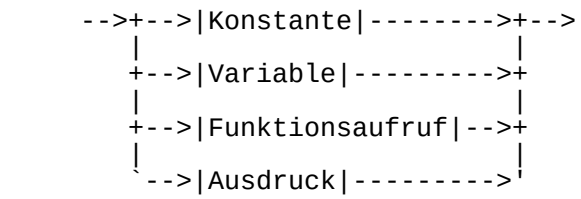
Verglichen werden können Zahlen oder Texte. Ergebnis von Vergleichen sind die logischen Aussagen "wahr" und "unwahr", die mit 1 und 0 dargestellt werden.

Eine Verwendung des Vergleichsoperator "#" ist bei Texten nicht zu empfehlen, da er in Kombination mit einem weiteren Zeichen auch als Steuerzeichen interpretiert wird (-> Kap.7.3.1.1). Nur schwer nachvollziehbare Fehlermeldungen können die Folge sein. Der Vergleichsoperator "<>" ist hingegen unproblematisch und daher nach Möglichkeit vorzuziehen.

7.3 Operanden

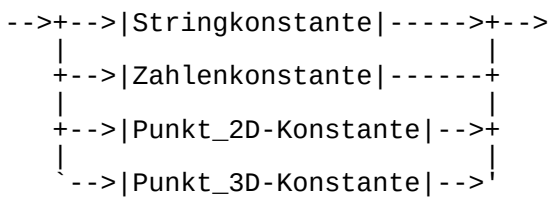
Operanden in Ausdrücken können Konstanten, Variablen, Funktionsaufrufe und Ausdrücke sein. Bei den in Ausdrücken aufrufbaren Funktionen handelt es sich um arithmetische Funktionen, trigonometrische Funktionen, Vektorfunktionen, Stringfunktionen und Abfragefunktionen. Sie dürfen nicht mit ME10-Interruptfunktionen verwechselt werden.

|Operand|



7.3.1 Konstanten

|Konstante|



ME10 kennt weiterhin die symbolischen Konstanten "PI", "TRUE" und "FALSE". Sie sind mit folgenden Zahlenkonstanten vorbelegt:

```
PI    --> Kreiszahl (3.14156....)
TRUE  --> 1
FALSE --> 0
```

7.3.1.1 Stringkonstanten (Text-Konstanten)

|Stringkonstante|

Eine Textkonstante ist eine Zeichenfolge, die aus einer beliebigen Kombination der folgenden Zeichen besteht:

- Eine Folge von druckbaren ASCII-Zeichen in Apostrophen oder Anführungszeichen
- # und ein weiteres Zeichen
- # und eine ganze Zahl, die die ASCII-Ordnungszahl des gewünschten Zeichens darstellt

Bei den druckbaren Zeichen zwischen Hochkommas oder Anführungszeichen handelt es sich um die ASCII-Zeichen mit den Ordnungsnummern 32 bis 126 sowie 161 bis 254. Aus Gründen der Eindeutigkeit müssen Hochkommas in Zeichenfolgen, die mit Hochkommas begrenzt sind, stets doppelt geschrieben werden. Anführungszeichen in Zeichenfolgen, die mit Anführungszeichen begrenzt sind, müssen ebenfalls doppelt geschrieben werden.

Wenn sich eine Textkonstante über mehrere Zeilen des Editors erstreckt, muß vor dem Zeilentrennungszeichen ein umgekehrter Schrägsrich "\" stehen, und die nächste Zeile muß in Spalte 1 beginnen.

Beispiel

```
DEFINE Test
  DISPLAY 'Dieser Text erst\
reckt sich über zwei Zeilen'
END_DEFINE
```

Wirkung des #-Zeichens

Ein Buchstabe oder Symbol nach dem Zeichen # entspricht einem Steuerzeichen. Beispielsweise entspricht #G oder #g der Eingabe <CTRL>+G, die das akustische Signal auslöst. Das System interpretiert den Buchstaben bzw. das Symbol nach der Formel "Zeichen(Ordnungsnummer(Buchstabe)Modulus32)". Die Ordnungsnummer von G ist 71, Modulus 32 von 71 ist 7 (Modulus = Rest der ganzzahligen Division), und der ASCII-Wert von 7 entspricht dem akustischen Signal.

Die auf das Zeichen # folgende Zahl muß zwischen 0 und 255 liegen. Dies entspricht der Code-Darstellung eines ASCII-Zeichens. Beispiel: Die Zeichenfolge #65#66#67 entspricht der Zeichenfolge 'ABC'.

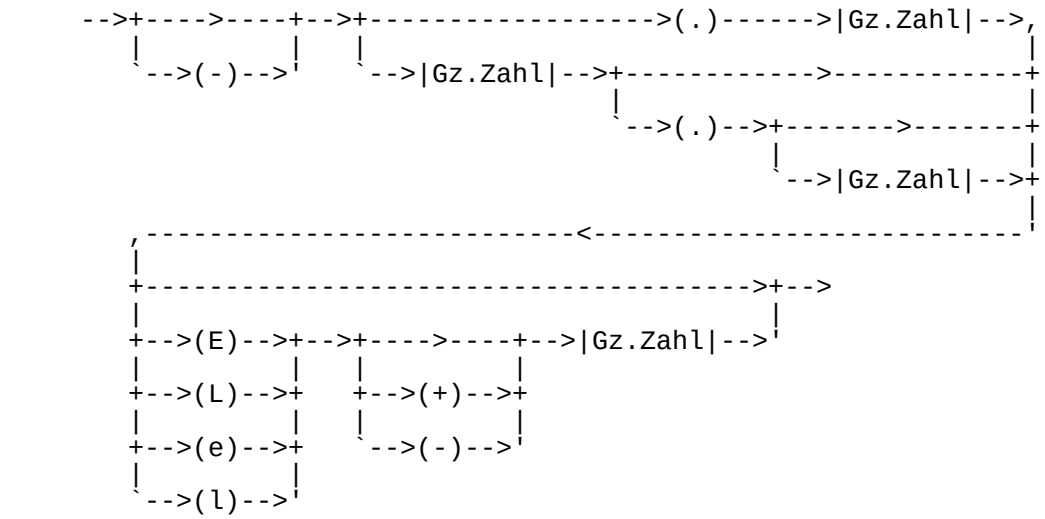
Beispiele

```
"Dies ist ein Text"         'Dies ist ein Hochkomma'''
```

7.3.1.2 Zahlenkonstanten

Zahlenkonstanten können mit und ohne Dezimalpunkt, in Dezimalschreibweise und in Exponentialschreibweise dargestellt werden.

|Zahlenkonstante|



|Gz. Zahl|

```
-->+-->|Ziffer|-->+-->
   |             |
   `------<-------'
```

Die Exponentialschreibweisen (E, L, e oder l) sind gleichbedeutend. Hierbei bedeutet z.B. "E", gefolgt von einer Zahl "x" die x-te Potenz von 10. Der maximale unterstützte Exponentenbereich liegt bei ca. -300 bis +300. Da das System jedoch häufig das Produkt von Punktkoordinaten berechnet, liegt der tatsächliche Exponentenbereich unter realistischen Gesichtspunkten eher bei -150 bis +150. Die Datengenauigkeit beträgt beim HP-UX System ungefähr 12 Dezimalstellen.

Beispiele: Die Zahlen +12 und -0,2 können wie folgt dargestellt werden:

```
Zahl "12"    :  12 ; +12 ; 12.0 ; 12E0 ; 12.0E0 ; 12.0E00 ;
                1.2E01 ; .12E02 ; ....
Zahl "-0,2"  :  -0.2 ; -0.2E0 ; -0.2E00 ; -0.02E01 ; -2E-1 ;
                -2.0E-01 ; ....
```

7.3.1.3 Punkt_2D-Konstanten (2D-Vektor-Konstanten)

Ein Punkt wird durch seine Koordinaten definiert.

|Punkt_2D-Konstante|

```
-->|Zahlenkonstante|-->(,)-->|Zahlenkonstante|-->
```

Die erste Zahlenkonstante definiert die X-Koordinate, die zweite Zahlenkonstante die Y-Koordinate.

7.3.1.4 Punkt_3D-Konstanten (3D-Vektor-Konstanten)

Ein 3D-Punkt wird durch seine Koordinaten definiert.

|Punkt_3D-Konstante|

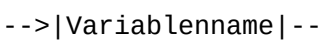
```
-->|Zahlenkonstante|-->(,)-->|Zahlenkonstante|-->(,)-->|Zahlenkonstante|-->
```

Die erste Zahlenkonstante definiert die X-Koordinate, die zweite Zahlenkonstante die Y-Koordinate und die dritte Zahlenkonstante die Z-Koordinate.

7.4 Variablen

Variablen werden durch Namen gekennzeichnet. Sie können ME10-Befehle, Optionen, integrierte Funktionen, Operator-Symbole, Texte, Zahlen, 2D-Vektoren und 3D-Vektoren zum Inhalt haben.

|Variable|

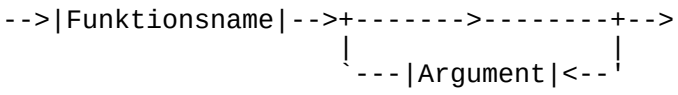


Vereinbarung und Verwendung von Variablen werden in den Kapiteln 4 und 5 beschrieben. Hier wurde auch schon erwähnt, daß ME10 genaugenommen keine Variablen in dem bei anderen Programmiersprachen üblichen Sinne kennt. Variablen sind vielmehr Makros, die einen einzigen Wert zum Inhalt haben. Bei Verwendung einer nicht definierten Variablen erscheint daher auch die Fehlermeldung "Das Makro *Makroname* ist nicht definiert".

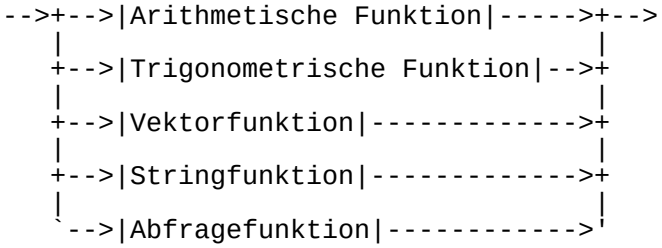
7.5 Funktionsaufrufe

Bei den in Ausdrücken aufrufbaren Funktionen handelt es sich um arithmetische Funktionen, trigonometrische Funktionen, Vektorfunktionen, Stringfunktionen und Abfragefunktionen. Sie dürfen nicht mit ME10-Interruptfunktionen verwechselt werden. Eine Funktion wird durch Angabe des Funktionsnamens und der Argumente aufgerufen. Das Ergebnis wird als Funktionswert bezeichnet.

|Funktionsaufruf|

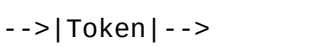


|Funktionsname|



Argumente müssen formal wie Token aufgebaut sein (-> Kap.4)

|Argument|



7.6 Reihenfolge bei der Auswertung von Ausdrücken

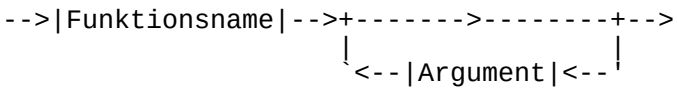
Ausdrücke werden entsprechend der Hierarchiestufe der Operanden und Operatoren ausgewertet. Innerhalb einer Hierarchiestufe erfolgt die Auswertung von links nach rechts. Mit Klammersetzung kann die Hierarchie durchbrochen werden. Nachfolgende Tabelle zeigt die Hierarchiestufen der Operanden und Operatoren. Hierbei bedeutet 1 die höchste und 6 die niedrigste Hierarchiestufe.

Stufe	Operatoren bzw. Operatoren
1	Klammerausdruck
2	Funktionsaufruf
3	^ **
4	* / DIV MOD AND
5	+ - OR EXOR
6	= <> > < >= <=

8 Integrierte Funktionen

ME10 verfügt über arithmetische Funktionen, trigonometrische Funktionen, Vektorfunktionen, Stringfunktionen und Sonstige Integrierte Funktionen. Eine Funktion wird durch Angabe des Funktionsnamens und gegebenenfalls von Argumenten aufgerufen. Funktionsname und Argumente werden durch Leerzeichen voneinander getrennt. Als Argumente sind Konstanten, Variablen und Ausdrücke zugelassen.

|Funktionsaufruf|



Das Ergebnis eines Funktionsaufrufs wird als Funktionswert bezeichnet. Der Funktionswert muß in einer Anweisung als Argument oder Parameter verwendet werden. Eine Anweisung, die nur aus einer integrierten Funktion besteht, ergibt keinen Sinn.

8.1 Arithmetische Funktionen

Arithmetische Funktionen verwenden Zahlen als Argumente und liefern Zahlen als Funktionswerte.

Funktionsname	Funktionswert
ABS	Absolutbetrag des Arguments
EXP	e hoch Argument (e = Eulersche Zahl)
FRACT	Dezimalteil des Arguments
INT	Größte ganze Zahl kleiner oder gleich dem Argument
LG	Zehnerlogarithmus des Arguments
LN	Natürlicher Logarithmus des Arguments
ROUND	Ganzzahlige Rundung
SGN	Vorzeichen des Arguments. Ergibt -1 für negative Argumente, 0 für Argumente = 0, + 1 für positive Argumente
SQR	Quadrat des Arguments
SQRT	Quadratwurzel des Arguments
TRUNC	Ganze Zahl des Arguments

Beispiel: Die Anweisung "LET A (SQRT B)" weist A die Wurzel aus B zu.

8.2 Trigonometrische Funktionen

Trigonometrische Funktionen verwenden Zahlen als Argumente und liefern Zahlen als Funktionswerte. Sie verwenden stets die aktuellen Winkeleinheiten.

Funktionsname	Funktionswert
SIN	Sinus des Arguments
COS	Cosinus des Arguments
TAN	Tangens des Arguments
ARCSIN	Arcus-Sinus des Arguments
ARCCOS	Arcus-Cosinus des Arguments
ARCTAN	Arcus-Tangens des Arguments

Beispiele

```
LET A (SIN Alpha)
LET Beta (ARCTAN (A + 2 * B))
```

8.3 Vektorfunktionen

Vektorfunktionen verwenden Vektoren (Punkt, Punkt_3D) oder Zahlen als Argumente. Ihre Funktionswerte sind Vektoren oder Zahlen. Winkel werden in den aktuellen Einheiten verwendet.

Funktionsname	Argument(e)	Funktionswert	Deutung des Funktionswertes
ABS	Pkt; Pkt_3D	Zahl	Länge des Vektors vom Ursprung zum Punkt
ANG	Pkt	Zahl	Winkel zwischen X-Achse des Eingabekoordinatensystems und dem Vektor vom Ursprung zum Punkt
INT	Pkt; Pkt_3D	Pkt; Pkt_3D	Punkt mit größten ganzzahligen Koordinaten kleiner/gleich den Argumentkoordinaten
LEN	Pkt; Pkt_3D	Zahl	Länge des Vektors vom Ursprung zum Punkt
MIRR	Pkt Zahl	Pkt	Ergebnispunkt entsteht durch Spiegelung des Argumentpunktes an einer durch den Ursprung verlaufenden, virtuellen Achse, deren Winkel zur X-Achse des Eingabekoordinatensystem durch das zweite Argument bestimmt wird
PNT_RA	Zahl Zahl	Pkt	Punkt, dessen Länge durch das erste und dessen Winkel zur X-Achse des Eingabekoordinatensystems durch das zweite Argument bestimmt wird
PNT_XY	Zahl Zahl	Pkt	Punkt, dessen X-Koordinate durch das erste und dessen Y-Koordinate durch das zweite Argument bestimmt wird
PNT_XYS	Zahl1 Zahl2 Zahl3	Pkt auf Bildschirm Nr.“Zahl3“	2D-Punkt, dessen X-Koordinate durch das erste und dessen Y-Koordinate durch das zweite Argument bestimmt wird. Das dritte Argument bestimmt den Bildschirm. Sinnvoll bei 2-Bildschirm-Installationen
PNT_XYZ	Zahl Zahl Zahl	Pkt_3D	3D-Punkt, dessen X- Y- und Z-Koordinaten durch das erste, zweite und dritte Argument bestimmt werden
ROT	Pkt Zahl	Pkt	Punkt aus Drehung des Argumentpunkts (Vektor) um einen durch das zweite Argument gegebenen Winkel
ROUND	Pkt; Pkt_3D	Pkt; Pkt_3D	Punkt mit ganzzahlig gerundeten Koordinatenwerten
TRUNC	Pkt; Pkt_3D	Pkt; Pkt_3D	Punkt mit Koordinaten aus den ganzen Zahlen der Argumentkoordinaten
X_OF	Pkt; Pkt_3D	Zahl	X-Koordinate des Argumentpunktes
Y_OF	Pkt; Pkt_3D	Zahl	Y-Koordinate des Arguments
Z_OF	Pkt_3D	Zahl	Z-Koordinate des Arguments

Beispiel

Nachfolgende Anweisungen erzeugen einen 2D-Vektor mit X = 50 und Y = 10.

```
LET Z1 50
LET Z2 10
LET P1 (PNT_XY Z1 Z2)
```

Man hätte hier auch schreiben können "LET P1 50,10". Falsch wäre hingegen "LET P1 Z1,Z2", denn die Definition einer 2D-Vektorkonstanten lautet nun einmal "Zahlenkonstante,Zahlenkonstante" und nicht "Zahlenvariable,Zahlenvariable"!

8.4 Stringfunktionen

Stringfunktionen verwenden Texte (Strings), Zahlen oder Anweisungselemente (Token) als Argumente. Ihre Funktionswerte sind Texte, Zahlen oder Anweisungselemente.

Funktionsname	Argument(e)	Funktionswert	Deutung des Funktionswertes
CHR	Zahl	Text	Das der Zahl zugeordnete Zeichen (Zuordnung erfolgt nach dem ASCII-Code)
LEN	Text	Zahl	Länge des Textes
LWC	Text	Text	Im Ergebnistext sind Großbuchstaben durch Kleinbuchstaben ersetzt
NUM	Text	Zahl	Ordnungszahl des ersten Zeichens im Text (Zuordnung erfolgt nach dem ASCII-Code)
MATCH	Text1 Text2	Zahl	Zahl = 0 wenn Text1 <> Text2 Zahl = 1 wenn Text 1 = Text 2 Text2 darf Jokerzeichen enthalten
POS	Text1 Text2	Zahl	Position von Text2 in Text1. Ergebnis = 0, wenn Text2 nicht in Text1 enthalten ist.
RPT	Text Zahl	Text	Eine durch das 2. Argument festgelegte Anzahl von Kopien des Argumenttextes
STR	Token	Text	Stringaspekt des Arguments (das Anweisungselement (Token) wird nicht ausgewertet, sondern in einen Text umgewandelt)
SUBSTR	Text Zahl1 Zahl2	Text	Teiltext aus erstem Argument. Zahl1 bestimmt die Anfangsposition, Zahl2 die Zeichenzahl des Teiltextes. Beim Auftreten eines Fehlers Signal.
TRIM	Text	Text	Führende und nachfolgende Leerstellen des Argumenttextes werden gelöscht
UPC	Text	Text	Im Ergebnistext sind Kleinbuchstaben durch Großbuchstaben ersetzt
VAL	Text	Token	Tokenaspekt des Arguments (der Argumenttext wird als Anweisungselement (Token) ausgewertet, Umkehrfunktion von STR)

Beispiele

CHR 92 liefert das Backslash-Zeichen "\"
POS '50,50' liefert die Zahl "1"
LEN 'HEWLETT PACKARD' liefert die Zahl "15"
RPT 'Ha' 5 liefert den Text "HaHaHaHaHa"
UPC 'Ja' liefert den Text "JA"

8.5 Sonstige Integrierte Funktionen

Funktionsname	Funktionswert
CHECK_BREAK	Liefert den Wert "1", wenn die BREAK-Taste mit IGNORE_BREAK deaktiviert und danach betätigt wurde. In allen anderen Fällen wird "0" geliefert.
CHECK_ERROR	Fehlerstatus (0 oder 1, siehe Kapitel 2.9 "Fehlerbehandlung")
DATE	Text mit Datum und Uhrzeit in folgendem Format: '14-DEC-96 16:49:30'
ERROR_STR	Erster, nach Aktivierung von TRAP_ERROR registrierter leichter Fehler (siehe Kapitel 2.9 "Fehlerbehandlung")
INQ	Ein durch vorherige Abfrage ermittelts Datum (siehe Kapitel 13)
LTAB_COLUMNS	Spaltenzahl einer logischen Tabelle (siehe Kapitel 15)
LTAB_ROWS	Zeilenzahl einer logischen Tabelle (siehe Kapitel 15)
LTAB_TITLES	Titelzahl einer logischen Tabelle (siehe Kapitel 15)
MAKE_TMP_NAME	Erzeugt Zugriffspfad und Namen für eine temporäre Datei
MEDIR	Pfad und Name des ME10-Installationsverzeichnisses
READ_LTAB	Feldeintrag einer logischen Tabelle (siehe Kapitel 15)
RND	Pseudozufallszahl zwischen 0 und 1
SNID	Text mit Produkt- und Seriennummer des ID-Moduls
TIME	Zahl der seit Mitternacht vergangenen Sekunden
TYPE	Liefert den Datentypen des Arguments. Nicht auf Makros und integrierte Funktionen als Argument anwendbar.
VERSION	ME10-Versionsbezeichnung

Die integrierten Funktionen CHECK_ERROR und ERROR_STR werden zur Fehlerbehandlung verwendet und in Kapitel 2.9 beschrieben. ME10 ermöglicht es, Elementdaten, den zuletzt mit einer Meßfunktion ermittelten Wert, die Nummer des zuletzt zur Digitalisierung benutzten Fensters und aktuelle Systemeinstellungen zu ermitteln. Die Abfrage dieser Daten erfolgt mit Hilfe der integrierten Funktion "INQ". In Kapitel 13 wird die Verwendung von INQ ausführlich beschrieben. Die integrierten Funktionen READ_LTAB, LTAB_ROWS, LTAB_COLUMN und LTAB_TITLES werden beim Arbeiten mit logischen Tabellen benötigt und in Kapitel 15 behandelt.

Der Funktionswert von TYPE kann COMMAND, FUNCTION, QUALIFIER, PSEUDO_COMMAND, SYMBOL, STRING, NUMBER, PNT, PNT3 und ILLEGAL (= unzulässig) sein. TYPE wird häufig dazu verwendet, den Datentyp einer Benutzereingabe abzufragen. Im nachfolgenden Beispiel wird in der Hinweiszeile bei Eingabe eines Punktes PNT, bei Eingabe einer Zahl NUMBER angezeigt.

```
READ PNT NUMBER 'Punkt oder Zahl eingeben' X
DISPLAY (TYPE X)
```

Da TYPE eine Auswertung des Arguments durchführt, können auch Ausdrücke angegeben werden. Die Auswertung ist auch der Grund, warum TYPE nicht auf die Datentypen MACRO und ARITHM_FUNCTION anwendbar ist.

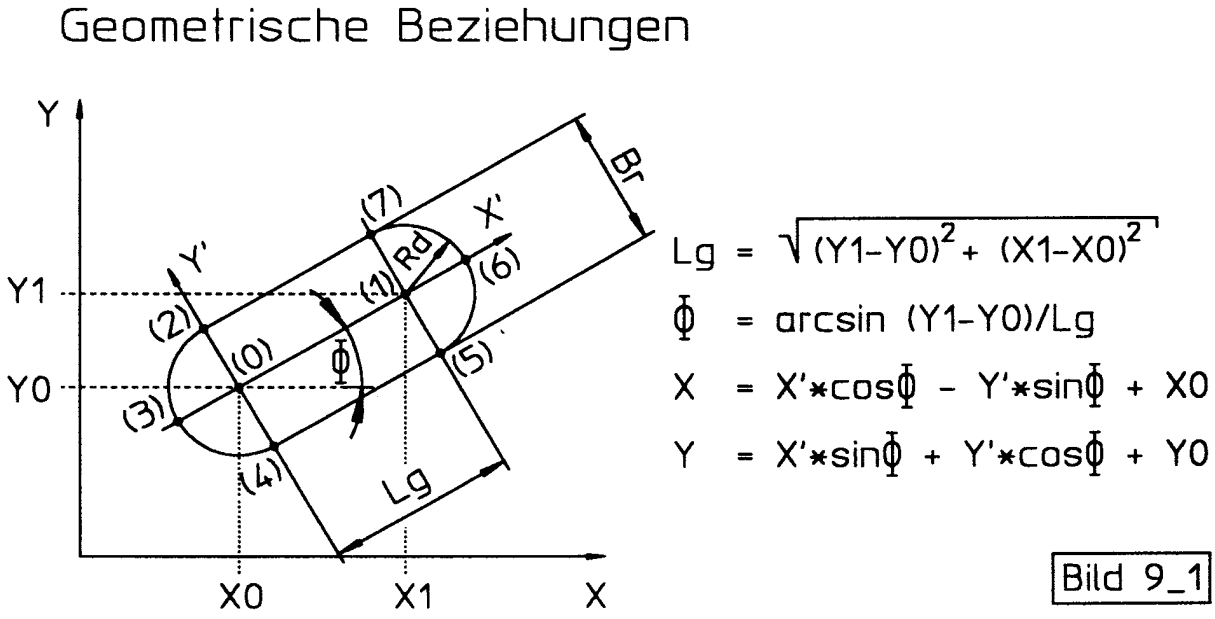
9 Verwendung von Vektoren

Punkte werden bei ME10 in Form von Ortsvektoren beschrieben. So entspricht z.B. ein Punkt P1 = 25,45 einem auf den Ursprung des Eingabekoordinatensystems bezogenen Vektor mit X = 25 und Y = 45.

Zeichenoperationen verlangen zunächst eine Bestimmung von Punkten. Punkte können durch Berechnung der Koordinatenwerte oder mit Hilfe von Vektorfunktionen bestimmt werden. Die Verwendung von Vektorfunktionen erfordert zwar Grundkenntnisse in der Vektorrechnung, führt aber zu wesentlich kürzeren und übersichtlicheren Makros.

Beispiel

Es soll ein Makro zum Zeichnen von Langlöchern mit beliebigen Abmessungen in beliebiger Winkellage geschrieben werden. Zunächst wird die Bestimmung der Konturpunkte durch Berechnung der Koordinatenwerte vorgenommen. Bild 9_1 zeigt ein um einen Winkel "ϕ" zur Horizontalen geneigtes Langloch.



Die Punkte (0) und (1) bestimmen die Lage der Rundungsmittelpunkte und damit Winkellage und Länge des Langlochs. Das Langloch hat die Breite "Br". Die Koordinatenwerte der Konturpunkte (2) bis (7) können mit den in Bild 9_1 angegebenen Formeln berechnet werden. Im nachfolgend aufgelisteten Makro Langloch_1 werden zunächst die Punkte (0) und (1) sowie die Breite des Langlochs vom Benutzer angefordert. Danach erfolgt die Berechnung der Koordinatenwerte der Konturpunkte (2) bis (7). Hierbei werden X- und Y-Werte getrennt berechnet. Die zum Zeichnen der Kontur notwendigen Punkte (2)

bis (7) werden mit Hilfe der Funktion PNT_XY aus den Koordinatenwerten gebildet.

```
DEFINE Langloch_1
{Zeichnen eines Langlochs unter Verwendung von Koordinaten}
{Fischer, 15.02.91, Stand 30.06.93}
  LOCAL P0 LOCAL P1
  LOCAL X0 LOCAL X1 LOCAL X2 LOCAL X3 LOCAL X4 LOCAL X5 LOCAL X6 LOCAL X7
  LOCAL Y0 LOCAL Y1 LOCAL Y2 LOCAL Y3 LOCAL Y4 LOCAL Y5 LOCAL Y6 LOCAL Y7
  LOCAL Br LOCAL Rd LOCAL Lg LOCAL Phi

  READ PNT 'Erster Mittelpunkt?' P0
  READ PNT 'Zweiter Mittelpunkt?' RUBBER_LINE P0 P1
  READ NUMBER 'Breite des Langlochs?' Br
  LET X0 (X_OF P0)
  LET Y0 (Y_OF P0)
  LET X1 (X_OF P1)
  LET Y1 (Y_OF P1)
  LET Rd (Br / 2)
  LET Lg (SQRT (SQR (Y1 - Y0) + SQR (X1 - X0)))
  LET Phi (ARCSIN ((Y1 - Y0) / Lg))
  LET X2 (-Rd * SIN Phi + X0)
  LET Y2 (Rd * COS Phi + Y0)
  LET X3 (-Rd * COS Phi + X0)
  LET Y3 (-Rd * SIN Phi + Y0)
  LET X4 (Rd * SIN Phi + X0)
  LET Y4 (-Rd * COS Phi + Y0)
  LET X7 (Lg * COS Phi - Rd * SIN Phi + X0)
  LET Y7 (Lg * SIN Phi + Rd * COS Phi + Y0)
  LET X5 (Lg * COS Phi + Rd * SIN Phi + X0)
  LET Y5 (Lg * SIN Phi - Rd * COS Phi + Y0)
  LET X6 ((Lg + Rd) * COS Phi + X0)
  LET Y6 ((Lg + Rd) * SIN Phi + Y0)
  LINE WHITE SOLID
    (PNT_XY X2 Y2) (PNT_XY X7 Y7) (PNT_XY X4 Y4) (PNT_XY X5 Y5)
  ARC THREE_PTS (PNT_XY X2 Y2) (PNT_XY X4 Y4) (PNT_XY X3 Y3) (PNT_XY X5 Y5)
                (PNT_XY X7 Y7) (PNT_XY X6 Y6)
  END
END_DEFINE
```

Im nachfolgenden Makro Langloch_2 werden die Punkte (2) bis (7) durch Vektoroperationen bestimmt (siehe auch Bild 9_2). Der hierfür verwendete Hilfsvektor "Hv" wird wie folgt gebildet:

- Die durch die Punkte (1) und (0) definierten Vektoren "P1" und "P0" werden voneinander abgezogen : P1 - P0.
- Der resultierende Differenzvektor wird um 90 Grad gedreht : ROT (..) 90
- Die Länge des Differenzvektors wird berechnet : LEN (P1 - P0)
- Der gedrehte Vektor wird durch seine Länge dividiert (es entsteht ein sogenannter Einheitsvektor der Länge 1)
- Der Einheitsvektor wird mit der halben Langlochbreite multipliziert.

Das Ergebnis ist ein Hilfsvektor "Hv" mit der Länge "Br/2". Er ist gegenüber der Längsrichtung des Langlochs um 90 Grad gedreht. Vektor "P2" bestimmt sich aus der Addition der Vektoren "P0" und "Hv". Vektor "P3" bestimmt sich aus der Addition von Vektor P0 und dem um 90 Grad gedrehen Vektor "Hv". Vektor "P4" bestimmt sich aus der Addition von Vektor "P0" und dem um 180 Grad gedrehten Vektor "Hv" (= -Hv). Die übrigen Vektoren berechnen sich in analoger Weise.

```
DEFINE Langloch_2
{Zeichnen eines Langlochs unter Verwendung von Vektoren}
{Fischer, 25.02.91, Stand 30.06.93}
  LOCAL P0 LOCAL P1 LOCAL P2 LOCAL P3 LOCAL P4
  LOCAL P5 LOCAL P6 LOCAL P7 LOCAL Br LOCAL Hv

  READ PNT 'Erster Mittelpunkt?' P0
  READ PNT 'Zweiter Mittelpunkt?' RUBBER_LINE P0 P1
  READ NUMBER 'Breite des Langlochs?' Br
  LET Hv (ROT (P1 - P0) 90 / LEN (P1 - P0) * Br/2)
  LET P2 (P0 + Hv)
  LET P3 (P0 + (ROT Hv 90))
  LET P4 (P0 - Hv)
  LET P5 (P1 - Hv)
  LET P6 (P1 + (ROT Hv -90))
  LET P7 (P1 + Hv)
  LINE WHITE SOLID P2 P7 P4 P5
  ARC THREE_PTS P2 P4 P3 P5 P7 P6
  END
END_DEFINE
```

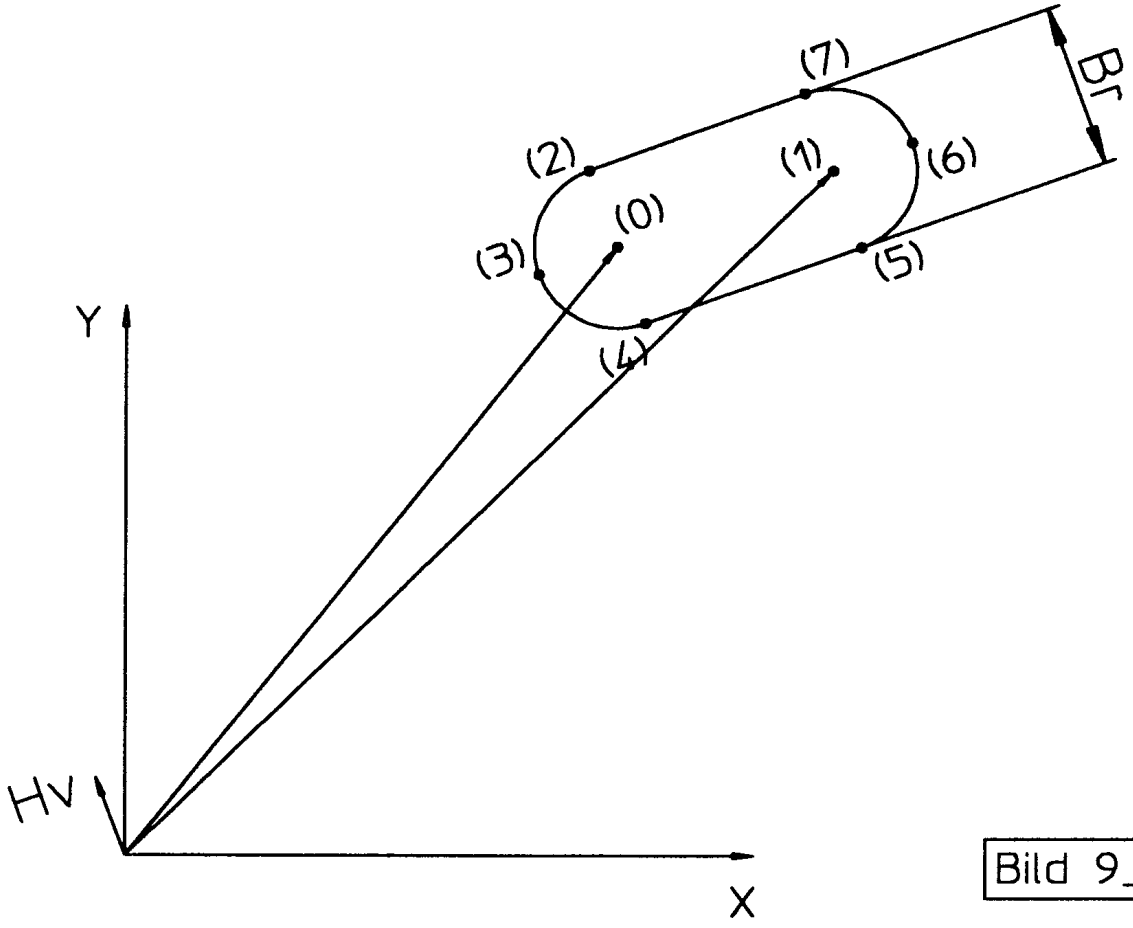
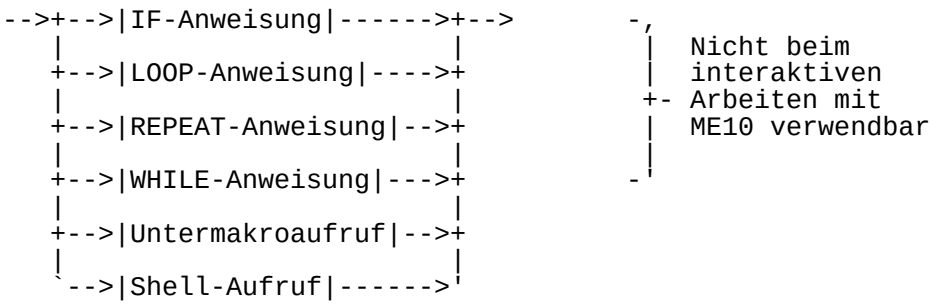


Bild 9_2

10 Steueranweisungen

Anweisungen eines Programms werden im allgemeinen nacheinander abgearbeitet. Steueranweisungen ermöglichen es, Programmteile zu wiederholen, zu überspringen oder in andere Programme zu verzweigen. Ein Teil davon darf nur in Makros und nicht beim interaktiven Arbeiten mit ME10 verwendet werden. ME10 kennt sechs Arten von Steueranweisungen.

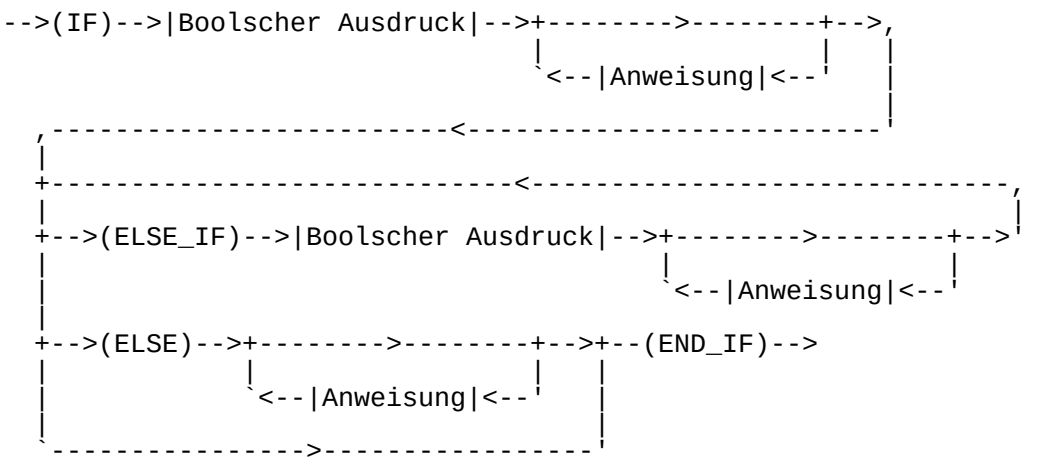
|Steueranweisung|



10.1 IF

Die IF-Anweisung ermöglicht ein gezieltes Abarbeiten bzw. ein gezieltes Überlesen von Anweisungsfolgen.

|IF|



Der auf IF folgende boolsche Ausdruck wird ausgewertet. Ist das Ergebnis "ungleich Null" oder "logisch wahr", werden die bis zum nächsten ELSE_IF, ELSE oder END_IF folgenden Anweisungen ausgeführt. Danach wird das Makro nach END_IF fortgesetzt. Ergibt die Auswertung einen Wert "gleich Null" oder "logisch unwahr", so werden die vorgenannten Anweisungen nicht ausgeführt, sondern das Makro wird mit der nächsten ELSE_IF-Anweisung, ELSE-Anweisung oder nach END_IF fortgesetzt.

Der auf ELSE_IF folgende boolsche Ausdruck wird nur geprüft, wenn eine vorangegangene, mit IF oder ELSE_IF vorgenommene, Prüfung zu einem Wert "gleich Null" oder zu "logisch unwahr" geführt hat. Prüfung und weitere Aktionen erfolgen dann analog zur IF-Anweisung. Die auf ELSE folgenden Anweisungen werden nur ausgeführt, wenn alle vorangegangenen Prüfungen zu Werten "gleich Null" oder zu "logisch unwahr" geführt haben.

Beispiel

In Makro "If_test" wird die Prüfung auf i < 8 nicht durchgeführt, da die vorangegangene Prüfung bereits zu einem Wert von "logisch wahr" geführt hat.

```
DEFINE If_test
  LOCAL I
  LET I 5
  IF (I < 10)
    DISPLAY ('Wert ist kleiner als 10')
   ELSE_IF (I < 8)
    DISPLAY ('Wert ist kleiner als 8')
   END_IF
END_DEFINE
```

Beispiel

In einen Makro soll der Benutzer aufgefordert werden, mit Ja oder Nein zu antworten und dazu entweder "J" oder "N" eingeben. Die Abfrage könnte wie folgt aussehen:

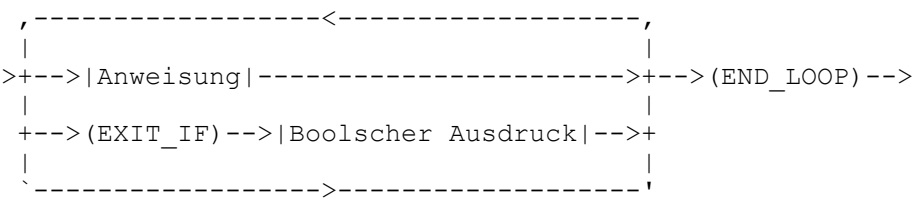
```
DEFINE Janein
  LOCAL Antwort
  READ STRING "Ihre Antwort? ('J' -> ja, 'N' -> nein)" Antwort
  IF (UPC Antwort = 'J')
    DISPLAY ("Sie haben mit 'J' oder 'j' geantwortet")
   ELSE_IF (UPC Antwort = 'N')
    DISPLAY ("Sie haben mit 'N' oder 'n' geantwortet")
   ELSE
    DISPLAY ('Ihre Antwort ist unzulaessig')
  END_IF
END_DEFINE
```

UPC wandelt Kleinbuchstaben in Großbuchstaben um. Dadurch ist sichergestellt, daß sowohl "J" als auch "j" als "Ja" interpretiert werden.

10.2 LOOP

Mit der LOOP-Anweisung können Progammschleifen erzeugt werden. Die Steuerung des Schleifendurchlaufs kann durch den Benutzer oder durch beliebig viele Abbruchkriterien erfolgen.

|LOOP|



Ohne besondere Vorkehrungen würden Schleifen solange durchlaufen, bis der Benutzer das Makro mit Hilfe einer Tastatureingabe abbricht. Ein Abbruch durch den Benutzer ist immer mit Hilfe der Break-Taste (bei DOS-Systemen mit der Tastenkombination <CTRL>+<BREAK>) möglich. Wenn das System im Rahmen des Benutzerdialogs auf Eingaben wartet, kann ein Abbruch auch mit der Escape-Taste oder durch Aufruf eines beliebigen ME10_Befehls (insbesondere END) erfolgen.

Zur Steuerung des Schleifendurchlaufs durch das Makro selbst können eine oder mehrere EXIT_IF-Anweisungen eingefügt werden. Beim Erreichen von EXIT_IF wird der nachfolgende Ausdruck ausgewertet. Ist das Ergebnis ein Wert "ungleich Null" oder "logisch wahr", wird die Schleife verlassen und der Makrolauf nach END_LOOP fortgesetzt. Die EXIT_IF-Anweisung darf sich an beliebigen Stellen der Schleife, nicht jedoch innerhalb einer IF-Anweisungsfolge (also zwischen IF und END_IF) befinden.

LOOP-Schleifen können in beliebiger Tiefe geschachtelt werden. Dabei müssen äußere Schleifen innere Schleifen stets vollständig einschließen. Das gleiche gilt auch für die Kombination von Schleifen mit IF-Anweisungsfolgen. Eine IF-Anweisungsfolge muß sich stets vollständig in einer Schleife befinden und Schleifen müssen stets vollständig in den auf IF, ELSE_IF oder ELSE folgenden Anweisungsblöcken enthalten sein.

Erlaubte Schachtelung Verbotene Schachtelung

Beispiel

In nachfolgendem Makro werden die dritten Wurzeln der Zahlen 27 bis 0 berechnet und in der Dialogzeile ausgegeben. Der Benutzer kann das Makro durch Betätigung der ESCAPE-Taste vorzeitig abbrechen. Die Anweisung "ENABLE_BREAK" stellt sicher, daß das Abbruchsignal nicht abgefangen wird (-> Kap. 2.2).

```
DEFINE Wurzel_3
{Makro zum Berechnen der dritten Wurzel von 27 bis 0}
{Fischer, 27.08.97, Stand 27.08.97}
  LOCAL N
  LET N 27
  ENABLE_BREAK
  DISPLAY 'Vorzeitiger Abbruch mit ESC-Taste möglich'
  LOOP
    DISPLAY ('Dritte Wurzel ' + STR N + ' = ' + STR (N^(1/3)))
    LET N (N - 1)
   EXIT_IF (N < 0)
  END_LOOP
END_DEFINE
```

10.3 REPEAT

Mit der REPEAT-Anweisung können Programmschleifen erzeugt werden. Die Anzahl der Schleifendurchläufe wird durch ein am Ende der Schleife stehendes Abbruchkriterium gesteuert.

|REPEAT|

```
-->(REPEAT)-->+-------->--------+-->(UNTIL)-->|Boolscher Ausdruck|-->
              |                 |
              `<--|Anweisung|<--'
```

Nach jedem Schleifendurchlauf wird der auf UNTIL folgende boolsche Ausdruck ausgewertet. Ist das Ergebnis ein Wert "gleich Null" oder "logisch unwahr", so wird die Schleife erneut durchlaufen. Bei einem Ergebnis "ungleich Null" oder "logisch wahr" erfolgt kein weiterer Schleifendurchlauf, sondern das Makro wird nach dem auf UNTIL folgenden Ausdruck fortgesetzt. Da das Abbruchkriterium am Ende der Schleife steht, erfolgt auf jeden Fall mindestens ein Schleifendurchlauf. REPEAT-Schleifen können in beliebiger Tiefe geschachtelt werden. Bezüglich der Schachtelungsmöglichkeiten und der Kombination mit IF-Anweisungsfolgen gelten die gleichen Beschränkungen wie bei LOOP-Schleifen.

Beispiel

Im Makro "Repeater" wird eine beliebige Anzahl von größer werdenden Kreisen in einer waagerechten Reihe gezeichnet. Die Steuerung des Schleifendurchlaufs erfolgt mit einer REPEAT-Anweisung.

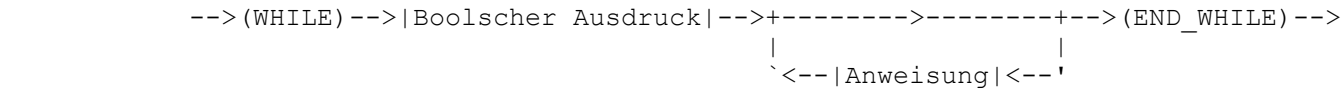
```
DEFINE Repeater
  LOCAL P0 LOCAL Pm LOCAL Anzahl LOCAL N
  READ PNT 'Startpunkt?' P0
  READ NUMBER 'Anzahl der Kreise' Anzahl
  LET N 0
  LET Pm P0
  REPEAT
    LET Pm (Pm + PNT_XY N 0)
    CIRCLE Pm (5 + N)
    LET N (N + 1)
  UNTIL (N >= Anzahl)
  END
END_DEFINE
```

Der Benutzer wird aufgefordert, die Anzahl der zu zeichnenden Kreise anzugeben. Da die Prüfung des Abbruchkriteriums erst am Ende der Schleife erfolgt, wird auch bei Eingabe von "0" ein Kreis gezeichnet. Im Abbruchkriterium erfolgt die Prüfung auf "N größer/gleich Anzahl". Eine Prüfung auf "N = Anzahl" wäre nur zulässig, wenn sichergestellt ist, daß der Benutzer die Zahl der zu zeichnenden Kreise größer als Null angibt.

10.4 WHILE

Mit der WHILE-Anweisung können Programmschleifen erzeugt werden. Die Anzahl der Schleifendurchläufe wird durch ein am Anfang der Schleife stehendes Abbruchkriterium gesteuert.

|WHILE|



Vor jedem Schleifendurchlauf wird der auf WHILE folgende boolsche Ausdruck ausgewertet. Ist das Ergebnis ein Wert "ungleich Null" oder "logisch wahr", so erfolgt ein Schleifendurchlauf. Bei einem Ergebnis "gleich Null" oder "logisch unwahr" erfolgt kein (erneuter) Schleifendurchlauf, sondern das Makro wird nach END_WHILE fortgesetzt. Da das Abbruchkriterium am Anfang der Schleife steht, erfolgt gegebenenfalls überhaupt kein Schleifendurchlauf. Man nennt solche Schleifen "abweisende Schleifen". WHILE-Schleifen können in beliebiger Tiefe geschachtelt werden. Bezüglich der Schachtelungsmöglichkeiten und der Kombination mit IF-Anweisungsfolgen gelten die gleichen Beschränkungen wie bei LOOP-Schleifen.

Beispiel

Im Makro "Whiler" wird eine beliebige Anzahl von größer werdenden Kreisen in einer waagerechten Reihe gezeichnet. Die Steuerung des Schleifendurchlaufs erfolgt mit einer WHILE-Anweisung.

```
DEFINE Whiler
  LOCAL P0 LOCAL Pm LOCAL Anzahl LOCAL N
  READ PNT 'Startpunkt?' P0
  READ NUMBER 'Anzahl der Kreise' Anzahl
  LET N 0
  LET Pm P0
  WHILE (N < Anzahl)
    LET Pm (Pm + PNT_XY N 0)
    CIRCLE Pm (5 + N)
    LET N (N + 1)
  END_WHILE
  END
END_DEFINE
```

Der Benutzer wird aufgefordert, die Anzahl der zu zeichnenden Kreise anzugeben. Da die Prüfung des Abbruchkriteriums bereits zu Beginn der Schleife erfolgt, wird bei Eingabe von "0" kein Kreis gezeichnet.

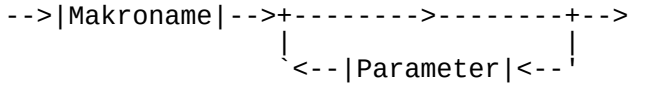
10.5 Untermakroaufruf

Wenn man als Unterprogramme Programme definiert, die nur in Verbindung mit Hauptprogrammen lauffähig sind, so kennt die Makrosprache keine Unterprogramme. Die aus Makros heraus aufrufbaren Makros unterscheiden sich nämlich formal nicht vom aufrufenden Makro, sie könnten im Prinzip auch vom Benutzer selbst aufgerufen werden. Die Bezeichnung "Untermakro" soll daher so verstanden werden, daß das Makro zwar als Untermakro verwendet wird, im Prinzip aber ein eigenständiges Makro ist.

Größere Makros sollten aus Gründen der Übersichtlichkeit generell in Untermakros unterteilt werden. Ein Hauptmakro dient zweckmäßigerweise nur dazu, den Aufruf der Untermakros zu steuern und gegebenenfalls Werte mit Hilfe von Parametern zu übergeben. Alle Eingaben, Verarbeitungen und Ausgaben erfolgen dann ausschließlich in den Untermakros.

Ein Makro wird durch Angabe seines Namens aufgerufen. Beim Aufruf können Parameter übergeben werden.

|Untermakroaufruf|



Die Bearbeitung des aufrufenden Makros wird bis zur Abarbeitung des aufgerufenen Makros (und der von diesem ggf. aufgerufenen Makros) unterbrochen. Parameter können als Konstanten oder allgemeine Ausdrücke angegeben werden. Im Falle von Ausdrücken erfolgt zunächst eine Auswertung und dann eine Übergabe der Ergebnisse. Leider ist eine Übertragung von Daten in umgekehrter Richtung, also vom aufgerufenen an das aufrufende Makro, mit Hilfe von Parametern nicht möglich. Sie kann nur auf folgende drei

Arten erfolgen.

1. Verwendung globaler Variablen

Diese Methode ist unübersichtlich und gefährlich, denn es besteht immer die Gefahr, daß bereits definierte globale Variablen unbeabsichtigt überschrieben werden. Der Benutzer kann diese Gefahr verringern, indem er bestimmte globale Variablen mit aussagefähigen Namen (z.B. Result1, Result2 usw.) ausschließlich für die Datenübertragung zwischen Makros verwendet.

2. Verwendung lokaler Variablen

Ein aufgerufenes Makro hat auf alle lokalen Variablen des aufrufenden Makros Zugriff, sofern sie im aufgerufenen Makro nicht erneut als lokal vereinbart wurden. Wenn der Inhalt der Variablen im Untermakro verändert wird, verändert sich auch ihr Inhalt im aufrufenden Makro. Auf diese Weise ist eine sicherere Datenübertragung als mit globalen Variablen möglich. Die Übersichtlichkeit wird gegenüber der ersten Methode erhöht, da die Gültigkeit der Variablen im allgemeinen auf wenige Makros beschränkt bleibt.

3. Verwendung von Zwischendateien

Zwischendateien können bei DOS- und HP-UX-Systemen auf einem Massenspeicher (z.B. der Festplatte) angelegt werden (-> Kap.11). Die Datenübertragung ist aufgrund der geringen Zugriffsgeschwindigkeit relativ langsam.

Beispiel

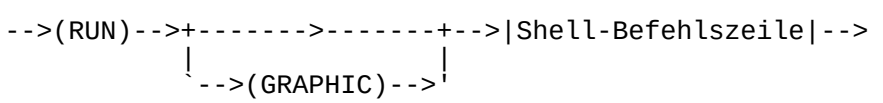
Untermakro "Iwcalc" berechnet Flächenträgheitsmoment, Biegewiderstandsmoment, polares Trägheitsmoment und Torsionswiderstandsmoment für Ringquerschnitte. Es wird von Makro "Call_test" unter Angabe der Parameter Außendurchmesser und Innendurchmesser aufgerufen. Die Übertragung der Ergebnisse an das aufrufende Makro erfolgt mit Hilfe der globalen Variablen "Result1" bis "Result4"

```
DEFINE Call_test
  LOCAL Aussen LOCAL Innen
  READ NUMBER 'Aussendurchmesser?' Aussen
  READ NUMBER 'Innendurchmesser?' Innen Iwcalc Aussen Innen
  DISPLAY ('Das Flaechentraegheitsmoment betraegt ' + STR Result1 + ' mm**4')
  DISPLAY ('Das Biegewiderstandsmoment betraegt ' + STR Result2 + ' mm**3')
  DISPLAY ('Das Polare Traegheitsmoment betraegt ' + STR Result3 + ' mm**4')
  DISPLAY ('Das Torsionswiderstandsmoment betraegt ' + STR Result4 + ' mm**4')
END_DEFINE


DEFINE Iwcalc
  PARAMETER Da PARAMETER Di
  LET Result1 (PI / 64 * (Da ^ 4 - Di ^ 4))
  LET Result2 (2 * Result1 / Da)
  LET Result3 (PI / 32 * (Da ^ 4 - Di ^ 4))
  LET Result4 (2 * Result3 / Da)
END_DEFINE
```

10.6 Shell-Aufruf

ME10 ermöglicht ein Absetzen von Befehlen an die Betriebssystemshell durch Makros oder durch den Benutzer selbst.

|Shell-Aufruf| (Interruptfunktion)



Der auf RUN bzw. GRAPHIC folgende String (Shell-Befehlszeile) wird an die Betriebssystemshell weitergegeben und von dieser als Betriebssystembefehl interpretiert. Dabei kann es sich um den Aufruf eines Betriebssystemprogramms, um den Aufruf der Shell selbst (die ja auch nur ein Programm des Betriebssystems ist) oder um den Aufruf eines anderen, in beliebigen Programmiersprachen geschriebenen Programmes handeln. Nach Abarbeitung des aufgerufenen Programms geht die Kommandogewalt an das die RUN-Anweisung absetzende Makro oder an den Benutzer zurück.

Die Option GRAPHIC ist nur von Bedeutung, wenn ME10 nicht unter der graphischen Benutzeroberfläche HP-VUE verwendet wird (näheres siehe ME10-HELP-System).

Beispiele

```
{Anlegen eines Unterverzeichnisses bei einem HP-UX-System}
RUN 'mkdir /users/hans/uebung'

{Aufruf des Programms "berechnung"}
RUN '/users/hans/berechnung'
```

11 Dateioperationen

ME10 ermöglicht die Verwendung von sequentiellen Dateien, um Daten während des Makrolaufs einzulesen, zwischenzuspeichern, abzuspeichern oder an Endgeräte oder andere Programme weiterzugeben. Sequentielle Dateien sind Textdateien, bei denen Zugriffe nur auf ganze Sätze und in der Reihenfolge der Sätze möglich sind. Jeder Satz entspricht einer Zeile der Datei und ist mit dem Zeilentrennungszeichen abgeschlossen. Die Dateien können bei DOS- und HP-UX-Systemen auf einem Massenspeicher (z.B. der Festplatte) angelegt werden.

Das bedeutet:

- Sequentielle Dateien können mit jedem ASCII-Texteditor gelesen und geschrieben werden.
- Nur ganze Sätze können gelesen oder geschrieben werden.
- Ein Zugriff auf Satz "n" ist nur möglich, wenn zuvor auf Satz "n-1" zugegriffen wurde.
- Bei der Ausgabe auf dem Bildschirm oder dem Drucker wird nach jedem Satz ein Zeilenwechsel ausgeführt.

Die HP-UX-Version von ME10 ermöglicht weiterhin eine Verwendung von benannten Pipes. Benannte Pipes sind temporäre Dateien, die im Hauptspeicher angelegt und in erster Linie zur Datenkommunikation zwischen Programmen verwendet werden. Sie werden in Kapitel 12.1.1 näher beschrieben. Die nachfolgenden Ausführungen gelten sowohl für normale Dateien als auch für benannte Pipes, wenn nicht ausdrücklich etwas anderes ausgesagt wird.

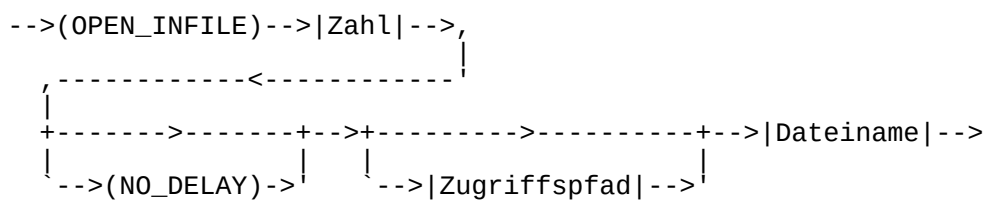
11.1 Öffnen von Dateien

Dateien können entweder zum Lesen oder zum Schreiben geöffnet werden. Ein Öffnen zum gleichzeitigen Lesen und Schreiben ist nicht möglich. Soll z.B. eine zum Lesen offene Datei beschrieben werden, muß sie zunächst geschlossen und danach zum Schreiben geöffnet werden.

11.1.1 Öffnen zum Lesen (OPEN_INFILE)

OPEN_INFILE öffnet eine Datei zum Lesen und positioniert einen Lesezeiger auf ihren ersten Satz.

OPEN_INFILE (Interruptfunktion)



Der Parameter |Zahl| ordnet der zu öffnenden Datei eine Kennummer zu, die bei allen nachfolgenden Lesezugriffen stellvertretend für die Dateibezeichnung verwendet wird. Die Kennummer muß zwischen 1 und 10 liegen. Da es aus Gründen der Eindeutigkeit nicht möglich ist, eine Kennummer mehreren offenen Dateien zuzuordnen, wird beim Öffnen einer Datei eine andere, bereits unter der gleichen Kennummer geöffnete Datei geschlossen. Damit können maximal 10 Dateien gleichzeitig geöffnet sein.

NO_DELAY steuert das Systemverhalten, wenn es sich bei der zu öffnenden Datei um eine benannte Pipe handelt. Ohne diese Option wird die Datei erst geöffnet, wenn sie zuvor von einem anderen Prozeß zum Schreiben geöffnet wurde. Das Öffnen wird also gegebenenfalls zunächst nicht ausgeführt. NO_DELAY heißt "keine Verzögerung" und bewirkt ein sofortiges Öffnen, unabhängig vom Dateistatus.

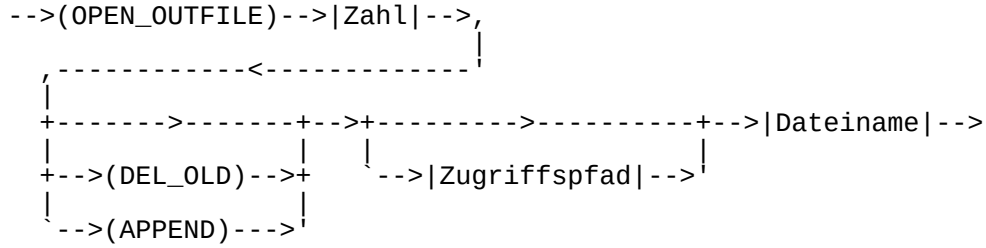
Beispiel: Die im aktuellen Verzeichnis befindliche Datei 'NORMTEIL.DAT' soll zum Lesen unter der Kennummer "5" geöffnet werden.

```
OPEN_INFILE 5 'normteil.dat'
```

11.1.2 Öffnen zum Schreiben (OPEN_OUTFILE)

OPEN_OUTFILE öffnet Dateien zum Schreiben. Bereits vorhandene Dateien können überschrieben oder ergänzt werden.

OPEN_OUTFILE (Interruptfunktion)



Der Parameter |Zahl| ordnet der zu öffnenden Datei eine Kennummer zu, die bei allen nachfolgenden Schreiboperationen stellvertretend für die Dateibezeichnung verwendet wird. Die Kennummer muß zwischen 1 und 10 liegen. Da es aus Gründen der Eindeutigkeit nicht möglich ist, eine Kennummer mehreren offenen Dateien zuzuordnen,

wird beim Öffnen einer Datei eine andere, bereits unter der gleichen Kennummer geöffnete Datei geschlossen. Damit können maximal 10 Dateien gleichzeitig geöffnet sein.

Die Optionen DEL_OLD und APPEND bestimmen, ob eine bereits existierende Datei überschrieben oder ergänzt werden soll (-> Kap.2.1.3).

Beispiel: Datei 'ergebnis.dat' soll im aktuellen Verzeichnis zum Schreiben unter der Kennummer "6" geöffnet werden. Eine eventuell existierende Datei gleichen Namens ist zu überschreiben.

```
OPEN_OUTFILE 6 DEL_OLD 'ergebnis.dat'
```

11.2 Schließen von Dateien (CLOSE_FILE)

CLOSE_FILE schließt offene Dateien.

CLOSE_FILE (Interruptfunktion)

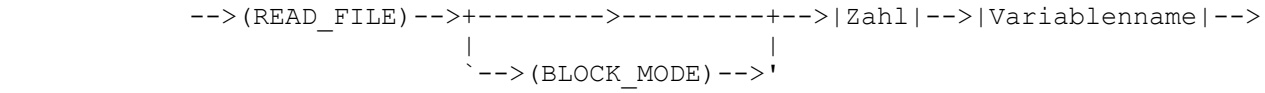
```
-->(CLOSE_FILE)-->|Zahl|-->
```

|Zahl| ist die Kennummer, unter der die Datei mit OPEN_INFILE oder OPEN_OUTFILE geöffnet wurde. Der Versuch, eine nicht geöffnete Datei zu schließen, führt zu einer Fehlermeldung. Noch geöffnete Dateien werden beim erneuten Öffnen mit OPEN_INFILE oder OPEN_OUTFILE zunächst automatisch geschlossen.

11.3 Lesen von Dateien (READ_FILE)

READ_FILE liest Sätze aus zum Lesen geöffneten Dateien.

READ_FILE (Interruptfunktion)



READ_FILE liest einen Satz (= Textzeile) aus der angegebenen Datei und ordnet ihn der mit dem Namen gekennzeichneten Variablen zu. Danach wird der Lesezeiger auf den nächsten Satz der Datei positioniert. Bei benannten Pipes werden gelesene Sätze automatisch gelöscht. Am Ende der Datei wird der Text 'END-OF-FILE' gelesen. Da immer nur ganze Sätze gelesen werden können, müssen einzelne Daten der Sätze mit Hilfe von Stringfunktionen voneinander getrennt werden.

Die Option BLOCK_MODE führt bei benannten Pipes zu einer zusätzlichen Pufferung der Eingabe. Es werden dann nur Zeichenketten zurückgegeben, die mit einem Zeilentrennungszeichen enden. Ist keine mit einem Zeilentrennungszeichen endende Zeichenkette vorhanden, wird 'END-OF-FILE' gelesen. Beim Erreichen der maximalen Pufferkapazität von 1023 Zeichen werden die gepufferten Zeichen auch ohne Zeilentrennungszeichen zurückgegeben.

Beispiel: Die Zeichen 7 bis 11 des dritten Satzes der Datei 'normteil.dat' sollen in einen Zahlenwert umgewandelt und der Variablen "Wert" zugewiesen werden.

```
OPEN_INFILE 5 'normteil.dat'
READ_FILE 5 Inzeile
READ_FILE 5 Inzeile
READ_FILE 5 Inzeile
LET Wert (VAL (SUBSTR Inzeile 7 5))
```

OPEN_INFILE öffnet die Datei zum Lesen. Danach wird der erste Satz gelesen und der Variablen "Inzeile" zugewiesen. Benötigt werden aber nicht Daten des ersten, sondern des dritten Satzes. Die Leseanweisung wird daher noch zweimal wiederholt. Dabei wird jeweils der alte Inhalt der Variable "Inzeile" vom zuletzt gelesenen Satz überschrieben. Nachdem die Variable den gewünschten Satz als Inhalt hat, werden daraus die Zeichen 7 bis 11 separiert (SUBSTR Inzeile 7 5), mittels der integrierten Funktion VAL in einen Zahlenwert umgewandelt und der Variablen "Wert" zugewiesen.

11.4 Schreiben in Dateien (WRITE_FILE)

WRITE_FILE schreibt Sätze in zum Schreiben geöffnete Dateien.

WRITE_FILE (Interruptfunktion)

```
-->(WRITE_FILE)-->|Zahl|-->|Token|-->
```

Das der Zahl folgende Anweisungselement (Token) wird ausgewertet, in einen Text umgewandelt und als nächster Satz in die durch |Zahl| gekennzeichnete Datei eingetragen. Der Begriff Token wurde in Kapitel 4 näher beschrieben. Normalerweise handelt es sich hierbei um eine Konstante oder einen Ausdruck.

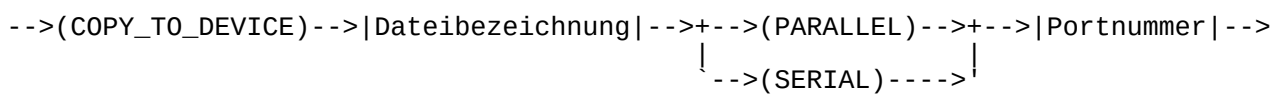
Beispiel: Die Variablen "A" und "B" sollen miteinander multipliziert und das Ergebnis mit einem entsprechenden Kommentar in die Datei "ergebnis.dat" geschrieben werden.

```
OPEN_OUTFILE 6 DEL_OLD 'ergebnis.dat'
WRITE_FILE 6 ('A mal B ergibt ' + STR (A * B))
```

11.5 Ausgabe von Dateien

COPY_TO_DEVICE gibt den Inhalt einer Plattendatei auf einem an einer seriellen oder parallelen Schnittstelle angeschlossenen Gerät aus. Diese Interruptfunktion steht nur bei DOS-Systemen zur Verfügung.

COPY_TO_DEVICE (Interruptfunktion)



Die mit |Dateibezeichnung| gekennzeichnete Datei wird auf einem an einer parallelen Schnittstelle (Option PARALLEL) oder auf einem an einer seriellen Schnittstelle (Option SERIAL) angeschlossenen Gerät ausgegeben. |Portnummer| kennzeichnet die Nummer der Schnittstelle. COPY_TO_DEVICE kann für beliebige Dateiarten verwendet werden.

Beispiel: Die Datei "ergebnis.dat" soll auf einem an der parallelen Schnittstelle Nr.1 angeschlossenen Drucker ausgegeben werden.

```
COPY_TO_DEVICE 'ergebnis.dat' PARALLEL 1
```

Bei HP-UX-Systemen steht COPY_TO_DEVICE nicht zur Verfügung. Hier können Dateien mit Hilfe des Editorbefehls $W '| lp' auf dem Drucker ausgegeben werden.

11.6 Dateiverwaltung

ME10 stellt eine Reihe von Funktionen zur Dateiverwaltung zur Verfügung. Zur Verwaltung von Plattendateien sind insbesondere folgende Interruptfunktionen nützlich:

- CATALOG : Auflisten von Verzeichnisinhalten
- COPY_FILE : Kopieren von Dateien
- PURGE_FILE : Löschen von Dateien und Verzeichnissen
- SEARCH : Einstellen des Standard-Suchpfads

Sie werden im ME10-Systemhandbuch und in der Help-Datei beschrieben.

11.7 Anwendungsbeispiel

Makro "Filedemo" liest Meßwerte aus Datei "messwert.dat", bildet die Mittelwerte aus den Werten einer Zeile und schreibt das Ergebnis in Datei "auswert.dat". Die Anzahl der Meßwerte einer Zeile der Datei "messwert.dat" ist unterschiedlich. Das Zeilenende wird durch einen Stern "*" gekennzeichnet. Zur besseren Lesbarkeit für den Anwender enthalten die ersten beiden Zeilen der Datei "messwert.dat" eine Nummerierung der Spalten. Diese Zeilen müssen im Makro überlesen werden, da sie keine Meßwerte enthalten.

Datei "messwert.dat"

```
        10        20        30        40        50
12345678901234567890123456789012345678901234567890
15.5 16.2 19.3 18.2 20.2 19.7 17.7 18.7 20.1 19.0 *
14.2 13.2 15.3 16.1 14.2 17.7 17.7 15.4 12.9 15.0 *
15.5 15.2 15.3 15.2 16.1 14.7 16.7 *
16.1 16.2 14.1 14.1 18.2 18.7 17.7 18.7 16.1 17.8 *
15.3 15.1 15.7 17.1 18.1 15.2 13.7 14.0 *
```

```
DEFINE Filedemo
{Lesen und Schreiben von sequentiellen Dateien}
{Fischer, 06.03.91, Stand 30.06.93}

  LOCAL Inzeile LOCAL I LOCAL N LOCAL Wert LOCAL Sum LOCAL Mittel

  {Öffnen der Eingabedatei und Überlesen der ersten 2 Sätze}
  OPEN_INFILE  1 'messwert.dat'
  READ_FILE 1 Inzeile
  READ_FILE 1 Inzeile

  {Öffnen der Ausgabedatei und Schreiben einer Überschrift}
  OPEN_OUTFILE 2 DEL_OLD 'auswert.dat'
  WRITE_FILE 2 'Mittelwerte'
  WRITE_FILE 2 '==========='

  {Lesen der Sätze, Trennen der Werte, Mittelwertbildung, Ausgabe}
  LET I 0

  {Lesen eines Satzes, I zählt dessen Nummer}
  LOOP

    READ_FILE 1 Inzeile
    {Abbruch bei Dateiende}
   EXIT_IF (Inzeile = 'END-OF-FILE')
    LET I (I + 1)
    LET N 0
    LET Sum 0

    {Abspalten der einzelnen Werte bis zum Satzende und Aufsummierung}
    {N zählt die Anzahl der Werte pro Satz}
    LOOP
      LET N (N + 1)
      LET Wert (SUBSTR Inzeile (N * 5 - 4) 5)
      {Abbruch, wenn Zeichen "*" erreicht ist}
     EXIT_IF (Wert = '*')
      LET Sum (Sum + VAL Wert)
    END_LOOP

    {Mittelwertbildung und Schreiben eines Ergebnissatzes}
    LET Mittel (Sum / (N - 1))
    WRITE_FILE 2 ('Messreihe ' + STR I + ': ' + STR Mittel)

  END_LOOP

  {Schließen der Dateien}
  CLOSE_FILE 1
  CLOSE_FILE 2

END_DEFINE
```

Datei "auswert.dat"

```
Mittelwerte
===========
Messreihe 1: 18.46
Messreihe 2: 15.17
Messreihe 3: 15.5285714285714
Messreihe 4: 16.77
Messreihe 5: 15.525
```

Hinweis: In den Zeilen der Eingabedatei "messwert.dat" sind mehrere, durch Leerzeichen getrennte, Werte enthalten. Mehrere Werte in einer Zeile sind zwar sehr übersichtlich, müssen aber im Makro durch aufwendige Stringoperationen wieder voneinander getrennt werden. Außerdem erhöhen die Leerzeichen die Größe der Datei unnötig. Einfacher und speicherplatzsparender ist es, wenn pro Zeile nur ein Zahlenwert eingetragen wird.

12 Einbinden von Anwendungsprogrammen

12.1 Grundlagen

In Kapitel 10 wurde gezeigt, daß beliebige Anwendungsprogramme aus einem Makro heraus aufgerufen werden können. Dadurch ist es möglich, vorhandene Programme für Problemlösungen zu verwenden oder Operationen auszuführen, die im Makro nicht oder nur sehr umständlich ausgeführt werden können. Die Anwendungsprogramme können als Vordergrundprozesse und - allerdings nur bei HP-UX-Systemen - als Hintergrundprozesse laufen. Der Datenaustausch zwischen einem Makro und einem Anwendungsprogramm kann über Plattendateien oder benannte Pipes erfolgen. Benannte Pipes sind allerdings nur bei HP-UX-Systemen möglich. Daten des Makros werden in eine erste Plattendatei bzw. Pipe geschrieben. Das Anwendungsprogramm liest diese Daten, verarbeitet sie und schreibt seine Ausgabedaten in eine zweite Plattendatei bzw. Pipe. Das Makro wiederum liest seine Eingabedaten aus der zweiten Plattendatei bzw. Pipe. Der Datenfluß sieht dann wie folgt aus:

```
Makro --> Plattendatei 1 --> Anwendungsprogramm --> Plattendatei 2 --> Makro
Makro --> Benannte Pipe 1 --> Anwendungsprogramm --> Benannte Pipe 2 --> Makro
```

Normalerweise ist es notwendig, Plattendateien bzw. benannte Pipes im Makro und im Anwendungsprogramm mit gleichen Namen zu benennen. Eine Umadressierung der Ein- und Ausgabe des Anwendungsprogramms hebt diese Einschränkung auf.

In diesem Kapitel werden benannte Pipes, das Starten und Abbrechen von Hintergrundprozessen und die Technik der Umadressierung erläutert.

12.1.1 Benannte Pipes

Eine benannte Pipe ist eine Pufferdatei, die im Hauptspeicher des Rechners angelegt wird und die Aufgabe hat, Daten zwischen Programmen auszutauschen. Benannte Pipes unterscheiden sich von Plattendateien vor allem durch wesentlich geringere Zugriffszeiten. Da die Hauptspeicheradressen benannter Pipes auf der Platte protokolliert werden, ist ein Arbeiten ganz ohne Plattenzugriffe leider nicht möglich.

Mit "Pipe" wird im Englischen eine Rohrleitung bezeichnet. Eine benannte Pipe ist, bildlich gesprochen, eine Daten-Rohrleitung, über die Programme miteinander kommunizieren. Sie kann die Daten nicht nur weiterleiten, sondern auch puffern. Dadurch ist es möglich, Programme miteinander kommunizieren zu lassen, die sich in ihrer Arbeitsgeschwindigkeit unterscheiden. Der Datenfluß ist nur in eine Richtung möglich. Wenn das Anwendungsprogramm vom Makro Daten erhält und auch Daten an das Makro zurückgibt, was meist der Fall ist, sind stets zwei Pipes notwendig. Beim Datenaustausch über Plattendateien gilt diese Einschränkung nicht. Er kann mit einer einzigen Datei erfolgen, wenn sichergestellt ist, daß es zu keinem gegenseitigen Überschreiben der Daten kommt.

Eine benannte Pipe arbeitet nach dem First-In-First-Out-Prinzip und wird daher auch als FIFO-Datei bezeichnet. Das First-In-First-Out-Prinzip besagt, daß zuerst geschriebene Daten auch wieder zuerst gelesen werden. Der Zugriff erfolgt satzweise und sequentiell. Gelesene Sätze werden automatisch gelöscht. Der Dateizeiger für das Lesen steht daher

immer vor dem ersten Satz und der Dateizeiger für das Schreiben immer hinter dem letzten Satz einer benannten Pipe. ME10 verwendet benannte Pipes ähnlich wie Plattendateien. Einige Besonderheiten sind jedoch dabei zu beachten. Zunächst zu den Gemeinsamkeiten.

- Eine benannte Pipe muß wie eine Plattendatei vor dem Beschreiben durch ME10 mit OPEN_OUTFILE und vor dem Lesen durch ME10 mit OPEN_INFILE geöffnet werden.
- Sie wird wie eine Plattendatei mit CLOSE_FILE für Zugriffe durch ME10 geschlossen.
- Sie wird wie eine Plattendatei mit READ_FILE gelesen und mit WRITE_FILE beschrieben.

Unterschiede bestehen in folgenden Punkten:

- Eine benannte Pipe muß vor ihrer Benutzung mit dem HP-UX-Befehl "/etc/mknod *pipename* p" eingerichtet werden und wird mit dem HP-UX-Befehl "rm *pipename*" gelöscht.
- Bevor eine benannte Pipe von ME10 zum Schreiben geöffnet werden darf, muß ein Anwendungsprogramm gestartet worden sein, das die geschriebenen Daten aus der Pipe liest.
- Bevor eine benannte Pipe von ME10 zum Lesen geöffnet werden darf, muß ein Anwendungsprogramm gestartet worden sein, das Daten in die Pipe schreibt.
- Da gelesene Daten automatisch gelöscht werden, ist ein wiederholtes Lesen gleicher Daten nicht möglich.
- Eine benannte Pipe kann Daten nur in eine Richtung weiterleiten.

Die HP-UX-Befehle zum Einrichten und zum Löschen der benannten Pipes werden zweckmäßigerweise mit Hilfe von Shell-Aufrufen abgesetzt. Das Anwendungsprogramm muß unbedingt als Hintergrundprozeß gestartet werden, wenn es Daten von ME10 erhält, die dort erst nach dem Start des Anwendungsprogramms eingelesen werden. Wenn dies nicht der Fall ist, könnte es prinzipiell auch als Vordergrundprozeß laufen. Um Schwierigkeiten aus dem Wege zu gehen, sollten jedoch Anwendungprogamme generell als Hintergrundprozesse gestartet werden. Diese Empfehlung kann erst begründet werden, wenn der Unterschied zwischen Hintergrund- und Vordergrundprozessen im folgenden Kapitel erläutert wurde.

12.1.2 Vorder- und Hintergrundprozesse

HP-UX ist ein Multitasking-Betriebssystem und bietet die Möglichkeit, einen Prozeß im Vordergrund und weitere Prozesse im Hintergrund laufen zu lassen. Um einen Prozeß im Hintergrund ablaufen zu lassen, muß dem Aufrufkommando lediglich das Zeichen "&" nachgestellt werden.

Der Vordergrundprozeß wird vom Benutzerterminal kontrolliert, die Hintergrundprozesse laufen von selbst ab und können nicht durch Eingaben am Benutzerterminal gesteuert werden. Aus diesem Grund ist es wichtig, daß Hintergrundprozesse ohne

Benutzereingaben auskommen. Ein Hintergrundprozeß, der eine Benutzereingabe benötigt, hat keine Möglichkeit, sie vom Benutzer anzufordern oder Eingaben des Benutzers zu lesen. Er kann dann nicht weiter bearbeitet werden und bleibt so lange in Warteposition, bis er mit einem besonderen HP-UX-Befehl abgebrochen oder bis das System heruntergefahren wird.

Wird ein Anwendungsprogramm aus ME10 heraus als Vordergrundprozeß gestartet, obwohl es Daten benötigt, die ME10 erst nach dem Start des Anwendungsprogramms liefert, so führt dies zu einer Fehlfuktion, im Extremfall sogar zu einer Blockade von ME10. Es wird daher dringend empfohlen, Anwendungsprogramme aus ME10 heraus nur als Hintergrundprozesse zu starten.

Vordergrundprozesse lassen sich mit Hilfe der Tastenkombination <Strg> + <c> abbrechen. Bei Hintergrundprozessen ist dies nicht möglich, da sie der direkten Kontrolle des Benutzers entzogen sind. Sie werden mit Hilfe des UNIX-Befehls "kill *prozeßnummer*" unter Angabe der Prozeßnummer abgebrochen. Die Prozeßnummern aller aktiven Prozesse können mit Hilfe des HP-UX-Befehls "ps -e" ermittelt werden.

12.1.3 Umadressierung

Die Betriebssysteme MS-DOS und HP-UX stellen eine Standardeingabe und eine Standardausgabe zur Verfügung, die im Normalfall mit der Tastatur und dem Bildschirm des Benutzerterminals verbunden sind. Dieser Dienst steht jedem Programm zur Verfügung, muß aber nicht genutzt werden. Die einem Programm zur Verfügung stehende Standardeingabe und Standardausgabe kann beim Aufruf des Programms so verändert werden, daß sie nicht mehr mit der Tastatur bzw. dem Bildschirm, sondern mit einer Datei verbunden ist. Man nennt diesen Vorgang "Umadressierung der Standardeingabe bzw. der Standardausgabe". Wenn das Programm dann die Standardeingabe oder Standardausgabe benutzt, kommuniziert es nicht mehr mit der Tastatur oder dem Bildschirm, sondern mit Dateien, ohne etwas davon zu bemerken. Bei den Dateien kann es sich um normale Dateien, um Gerätedateien oder - bei HP-UX-Systemen - um benannte Pipes handeln.

Ein Anwendungsprogramm, das die Standardeingabe und die Standardausgabe benutzt, läßt sich besonders einfach in ein Makro einbinden. Seine Ein- und Ausgaben können auf beliebige Dateien umadressiert werden, ohne es verändern zu müssen. Es kommuniziert nach wie vor mit der Standardeingabe und der Standardausgabe und merkt nichts von der Umadressierung. Die Benutzung der Standardeingabe mit nachfolgender Umadressierung hat nicht nur für das Schreiben von Makros, sondern auch für das Schreiben von Anwendungsprogrammen Vorteile. In vielen Programmiersprachen sind Lese- und Schreibanweisungen für die Standardeingabe und die Standardausgabe einfacher zu programmieren als entsprechende Anweisungen für Dateien. Der Programmierer des Anwendungsprogramms muß sich auch nicht um das Einrichten, Öffnen und Schließen der Dateien kümmern, da diese Aufgaben vom Betriebssystem übernommen werden. Die Übertragbarkeit der Anwendungsprogramme auf andere Rechnersysteme wird in der Regel erhöht, da Ein- und Ausgabeanweisungen für Dateien häufig sytemspezifisch sind.

Die Umadressierung findet bei MS-DOS-Systemen und bei HP-UX-Systemen mit Hilfe der

Zeichen "<", ">" und ">>" statt. "<" symbolisiert die Eingabe, ">" die Ausgabe mit Überschreiben vorhandener Dateien und ">>" die Ausgabe mit Ergänzen vorhandener Dateien. HP-UX-Systeme bieten darüber hinaus noch die Möglichkeit, Fehlermeldungen in Dateien umzulenken.

Beispiele für Shell-Aufrufe aus Makros (für MS-DOS-Systeme und HP-UX-Systeme)

```
RUN 'program1 < datei1'
```

ruft Programm "program1" auf und bestimmt, daß es seine Eingabedaten aus Datei "datei1" liest. Der Grafikbildschirm wird während des Abarbeitens des aufgerufenen Programms ausgeblendet.

```
RUN 'program2 > datei2'
```

ruft Programm "program2" auf und bestimmt, daß es seine Ausgabedaten in Datei "datei2" schreibt.

```
RUN 'program2 >> datei2'
```

ruft Programm "program2" auf und bestimmt, daß es seine Daten an das Ende der bereits vorhandenen Datei "datei2" anhängt.

```
RUN 'program3 < datei3 > datei4'
```

ruft Programm "program3 auf und bestimmt, daß es seine Eingabedaten aus Datei "datei3" liest und seine Ausgabedaten in Datei "datei4" schreibt.

Umlenkung der Fehlerausgabe (nur bei HP-UX-Systemen möglich)

HP-UX bietet die Möglichkeit, Fehlerausgaben des Anwendungsprogramms in eine Datei umzuleiten. Dies geschieht mit Hilfe der Umleitungssymbole "2>" und "2>>".

```
RUN 'program3 < datei3 > datei4 2> fehler'
```

ruft Programm "program3" auf und bestimmt, daß es seine Eingabedaten aus Datei "datei3" liest, seine Ausgabedaten in Datei "datei4" schreibt und Fehlermeldungen, die beim Lauf des Anwendungsprogramms auftreten, in Datei "fehler" einträgt. Der Grafikbildschirm bleibt erhalten. Die Datei "fehler" wird, sofern schon vorhanden, überschrieben. Bei Verwendung des Umleitungssymbols "2>>" würde eine bereits vorhandene Fehlerdatei ergänzt.

Hinweis

Namen von Dateien und benannten Pipes, welche für einen Datenaustausch mit Anwendungsprogrammen verwendet werden, dürfen die vom Betriebssystem vorgegebene Länge nicht überschreiten, obwohl diese Grenzen für ME10 normalerweise nicht existieren.

12.2 Anwendungsbeispiel

Die Makrosprache verfügt im Gegensatz zu vielen anderen Programmiersprachen über keine integrierten Funktionen zur Berechnung der hyperbolischen Funktionen "Sinus hyperbolicus", "Cosinus hyperbolicus" und "Tangens hyperbolicus". Im folgenden Beispiel wird das Anwendungsprogramm "hypfunc" in ein Makro eingebunden. "hypfunc" ist in der Programmiersprache "C" geschrieben und berechnet die hyperbolischen Funktionen durch Funktionsaufrufe. Es ist am Schluß dieses Kapitels aufgelistet. Mathematisch interessierte Leser haben natürlich schon bemerkt, daß sich die hyperbolischen Funktionen auch mit Hilfe der in der Makrosprache vorhandenen e-Funktionen berechnen lassen. Diese Möglichkeit soll hier jedoch nicht genutzt werden.

Zum Verständnis des Makros ist es nicht notwendig, sich in das C-Programm einzuarbeiten. Wenn die erforderlichen Ein- und Ausgabe eines Programms genau spezifiziert sind, ist es nicht notwendig, den Quellcode zu kennen, um damit zu arbeiten. Man sollte sich allerdings sicher sein, daß das Programm fehlerfrei arbeitet und nur das tut, was man von ihm erwartet.

"hypfunc" erwartet folgende Eingabedaten in der Standardeingabe:

- Kennummer für die Art der hyperbolischen Funktion (1 für Sinh, 2 für Cosh, 3 für Tanh)
- Das Winkelargument in Grad

Die Eingaben sind in getrennte Zeilen zu schreiben. "hypfunc" berechnet nun den gewünschten Funktionswert und schreibt das Ergebnis in die Standardausgabe.

Das Makro wurde in zwei Versionen geschrieben. Beide Versionen sind im Anschluß an diese Erläuterungen aufgelistet. Die zuerst angegebene Version "Hyp_file" ist auf MS-DOS-Systemen und auf HP-UX-Systemen lauffähig und verwendet zwei Plattendateien zum Datenaustausch. Es wäre durchaus möglich, nur eine einzige Datei zu verwenden, da die vom Makro in die erste Zwischendatei geschrieben Daten nach dem Lesen durch das Anwendungsprogramm "hypfunc" nicht mehr benötigt werden und daher von diesem überschrieben werden können. Aus Gründen der Übersichtlichkeit wurde jedoch darauf verzichtet.

In "Hyp_file" wird, nachdem die Kennung für die Art der hyperbolischen Funktion und der Winkel eingelesen und überprüft wurden, die Plattendatei "out.dat" geöffnet und die Werte dort eingetragen. Der Aufruf des C-Programms lautet dann:

```
RUN 'hypfunc < out.dat > in.dat'
```

Das Anwendungsprogramm "hypfunc" wird aufgerufen und dabei angewiesen, seine Eingabedaten aus "out.dat" zu lesen und seine Ausgabedaten in "in.dat" zu schreiben. Danach wird im Makro die Datei "in.dat" geöffnet und gelesen. "in.dat" kann nicht schon früher geöffnet werden, da sie erst von "hypfunc" erzeugt wird. Die Zwischendateien werden anschließend gelöscht und das Ergebnis in der Hinweiszeile ausgegeben.

Das Makro ist so geschrieben, daß nach einem Durchlauf eine erneute Benutzereingabe angefordert wird. Es wäre möglich, die Zwischendateien nur einmal anzulegen und erst beim endgültigen Beenden des Makros zu löschen. Dieses Verfahren hat jedoch den Nachteil, daß das Löschen unterbleibt, wenn der Benutzer den Makrolauf abbricht.

Das Makro "Hyp_pipe" ist nur auf HP-UX-Systemen lauffähig, da es mit benannten Pipes arbeitet. Es unterscheidet sich von "Hyp_file" in folgenden Punkten:

- Es werden zwei benannt Pipes als Zwischenablage eingerichtet:

```
RUN '/etc/mknod out_pipe p'
RUN '/etc/mknod in_pipe p'
```

- Das Anwendungsprogramm wird unmittelbar nach dem Einrichten der benannten Pipes als Hintergrundprozeß gestartet.

```
RUN 'hypfunc < out.pipe > in.pipe &'
```

- Die benannten Pipes werden mit Hilfe des HP-UX-Befehls "rm" gelöscht:

```
RUN 'rm out_pipe'
RUN 'rm in_pipe'
```

Makro "Hyp_file" (auf MS-WINDOWS-Systemen und HP-UX-Systemen lauffähig)

```
DEFINE Hyp_file
{Makro zum Berechnen der hyperbolischen Funktionen}
{Sinh (Phi), Cosh (Phi) und Tanh (Phi)}
{MS-WINDOWS- und HP-UX-Version}
{Zugelassen werden Winkel Phi = 0 - 360 Grad}
{Die Berechnung erfolgt im C-Programm "hypfunc"}
{Datenaustausch über Plattendateien}
{Fischer, 23.09.91, Stand 15.11.96}

LOCAL F_nummer  {Nummer der hyperbolischen Funktion}
LOCAL Phi       {Winkel in Grad}
LOCAL F_wert    {Funktionswert}

LOOP

  LOOP
    READ NUMBER 'F_nummer? (Sinh = 1, Cosh = 2, Tanh = 3, Ende = 0) : ' F_nummer
    {Prüfen auf Abbruch}
    IF (F_nummer = 0) CANCEL END_IF
    {Prüfen auf zulässige Eingabe}
   EXIT_IF ((F_nummer = 1) OR (F_nummer = 2) OR (F_nummer = 3))
    BEEP DISPLAY 'Nur 1, 2, 3 oder 0 zulässig'
  END_LOOP

  LOOP
    READ NUMBER 'Winkel in Grad (Maximalwert ± 360 Grad) : ' Phi
    {Prüfen auf zulässige Eingabe}
   EXIT_IF (ABS Phi <= 360)
    BEEP DISPLAY 'Nur Winkel bis maximal ± 360 Grad zulässig'
  END_LOOP

  {Öffnen der Ausgabedatei "out.dat"}
  OPEN_OUTFILE 6 DEL_OLD 'out.dat'

  {Schreiben in Datei "outdat" und Schließen der Datei}
  WRITE_FILE 6 F_nummer
  WRITE_FILE 6 Phi
  CLOSE_FILE 6
  {Aufruf des C-Programms "hypfunc", Umadressierung der Eingabe auf}
  {out.dat und der Ausgabe auf in.dat}
  RUN 'hypfunc < out.dat > in.dat'

  {Öffnen der Eingabedatei "in.dat", Lesen des vom C-Programm}
  {eingetragenen Funktionswertes und Schließen der Datei}
  OPEN_INFILE 5 'in.dat'
  READ_FILE 5 F_wert
  CLOSE_FILE 5

  {Umwandlung STRING --> NUMBER}
  LET F_wert (VAL F_wert)

  {Löschen der Dateien}
  PURGE_FILE 'out.dat' CONFIRM
```

```
  PURGE_FILE 'in.dat' CONFIRM

  {Anzeigen des Ergebnisses}
  IF (F_nummer = 1)
    DISPLAY ('Sinh (' + (STR Phi) + ') = ' + STR F_wert)
   ELSE_IF (F_nummer = 2)
    DISPLAY ('Cosh (' + (STR Phi) + ') = ' + STR F_wert)
   ELSE
    DISPLAY ('Tanh (' + (STR Phi) + ') = ' + STR F_wert)
  END_IF

END_LOOP
END_DEFINE
```

Makro "Hyp_pipe" (nur auf HP-UX-Systemen lauffähig)

```
DEFINE Hyp_pipe
{Makro zum Berechnen der hyperbolischen Funktionen}
{Sinh (Phi), Cosh (Phi) und Tanh (Phi)}
{HP-UX-Version}
{Zugelassen werden Winkel Phi = 0 - 360 Grad}
{Die Berechnung erfolgt im C-Programm "hypfunc"}
{Datenaustausch über benannte Pipes}
{Fischer, 24.09.91, Stand 04.09.96}

LOCAL F_nummer  {Nummer der hyperbolischen Funktion}
LOCAL Phi       {Winkel in Grad}
LOCAL F_wert    {Funktionswert}

LOOP

  LOOP
    READ NUMBER 'F_nummer? (Sinh = 1, Cosh = 2, Tanh = 3, Ende = 0) : ' F_nummer
    {Prüfen auf Abbruch}
    IF (F_nummer = 0) CANCEL END_IF
    {Prüfen auf zulässige Eingabe}
   EXIT_IF ((F_nummer = 1) OR (F_nummer = 2) OR (F_nummer = 3))
    BEEP DISPLAY 'Nur 1, 2, 3 oder 0 zulässig'
  END_LOOP

  LOOP
    READ NUMBER 'Winkel in Grad (Maximalwert ± 360 Grad) : ' Phi
    {Prüfen auf zulässige Eingabe}
   EXIT_IF (ABS Phi <= 360)
    BEEP DISPLAY 'Nur Winkel bis maximal ± 360 Grad zulässig'
  END_LOOP

  {Einrichten zweier benannter Pipes und Starten des C-Programms als}
  {Hintergrundprozeß mit Umadressierung der Ein- und Ausgabe}
  RUN '/etc/mknod out_pipe p'
  RUN '/etc/mknod in_pipe p'
  RUN 'hypfunc < out_pipe > in_pipe &'

  {Öffnen von "out_pipe" zum Schreiben}
  OPEN_OUTFILE 6 DEL_OLD 'out_pipe'
  {Schreiben in "out_pipe" und Schließen}
  WRITE_FILE 6 F_nummer
  WRITE_FILE 6 Phi
  CLOSE_FILE 6

  {Öffnen von "in_pipe" zum Lesen}
  OPEN_INFILE 5 'in_pipe'

  {Lesen des vom C-Programm in "in_pipe" eingetragenen}
  {Funktionswertes und Schließen}
  READ_FILE 5 F_wert
  CLOSE_FILE 5

  {Umwandlung STRING --> NUMBER und Löschen der benannten Pipes}
  LET F_wert (VAL F_wert)
  RUN 'rm out_pipe' RUN 'rm in_pipe'

  {Anzeigen des Ergebnisses}
  IF (F_nummer = 1)
    DISPLAY ('Sinh (' + (STR Phi) + ') = ' + STR F_wert)
   ELSE_IF (F_nummer = 2)
    DISPLAY ('Cosh (' + (STR Phi) + ') = ' + STR F_wert)
```

```
   ELSE
    DISPLAY ('Tanh (' + (STR Phi) + ') = ' + STR F_wert)
  END_IF

END_LOOP

END_DEFINE
```

C-Programm "hypfunc" als Quellcode

```
/* C-Programm "Hypfunc" zur Berechnung der hyperbolischen         */
/* Funktionen sinh (phi), cosh (phi) und tanh (phi)               */
/* Eingabedaten : Funktionsnummer ( 1 = Sinh, 2 = Cosh, 3 = Tanh  */
/*                Winkel in Grad                                  */
/* Ausgabedaten : Funktionswert                                   */
/* Fischer, 23.09.91, Stand 30.06.93                              */

#include <stdio.h>
#include <math.h>
#define pi 3.14159265

main()
{
 int nummer;
 float phi, wert;
 scanf ("%d", &nummer);
 scanf ("%f", &phi);
 phi = phi * pi / 180;
 if (nummer == 1)
  wert = sinh (phi);
 else if (nummer == 2)
  wert = cosh (phi);
 else if (nummer == 3)
  wert = tanh (phi);
 printf ("%2.6f\n", wert);
}
```

13 Abfrageanweisungen

Mit Hilfe von Abfrageanweisungen lassen sich folgende Daten ermitteln:

- Elementart, Geometriedaten, Farbe und Linienart von Zeichnungselementen (z.B. Mittelpunktskoordinaten, Radius, Farbe und Linienart eines Kreises).
- Daten von ME10-Teilen
- Systemeinstellungen (z.B. der aktuelle Fangmodus).
- Information zu Bildschirmmenüs und Anzeigetabellen
- Information zu vorausgegangenen Systemaktionen

Diese Daten werden beim Aufruf bestimmter Interruptfunktionen in eine systeminterne Abfrageliste eingetragen und können daraus mit Hilfe der Interruptfunktion INQ unter Angabe eines Parameters (inq_index) beliebig oft gelesen werden. Ein erneuter Aufruf einer eintragenden Funktion überschreibt die Liste mit den zugehörigen aktuellen Daten. Beim Versuch, Daten aus einem leeren Listenfeld zu lesen, wird ein Fehler gemeldet.

Die Abfrage von Daten geschieht also in zwei Schritten:

1. Eintrag der Daten in die Abfrageliste durch Aufruf einer eintragenden Funktion
2. Auslesen der interessierenden Daten mit Hilfe der Abfragefunktion INQ

Der Funktionswert der Abfragefunktion INQ hängt vom verwendeten Parameter (inq_index) ab.

inq_index	Funktionswert
1 - 9	Zahl
101 - 109	2D-Vektor
201 - 209	3D-Vektor
301 - 309	Text
401 - 409	Befehl

inq_index	Funktionswert
501 - 509	Interruptfunktion
601 - 609	Option
701 - 709	Pseudokommando
801 - 809	Integrierte Funktion

13.1 Abfrage von Elementdaten

Elementdaten können für einzelne Elemente oder für mehrere Elemente ermittelt werden.

13.1.1 Elementdatenermittlung für ein einzelnes Element

Schritt 1: Eintrag der Daten in die Abfrageliste mit Hilfe der Interruptfunktion INQ_ELEM

INQ_ELEM (Interruptfunktion)

-->(INQ_ELEM)-->|Punkt|-->

INQ_ELEM trägt Information über das durch den Punkt identifizierte Element in die Abfrageliste ein. Bezüglich des Objektfangs gelten die in Kapitel 2.12 gemachten Ausführungen.

Schritt 2: Auslesen der interessierenden Daten mit Hilfe der Abfragefunktion INQ

INQ (Integrierte Funktion)

-->(INQ)-->|inq_index|-->

Je nach Elementart und Abfrageindex (inq_index) liefert INQ folgendes Ergebnis:

Elementart	inq_index	Funktionswert
Jede Elementart	1	107 (impliziter inq_code)
	9	Anzahl der Infostrings des Elements
	10	Stiftbreite des Elements
	13	Polyelement-Flag 0 : Element gehört nicht zu einem Poyelement 1 : Element gehört zu einem Polyelement
	14	Anzahl der erfaßten Elemente, wenn mehrere Elemente unter Verwendung der Funktion INQ_SELECTED_ELEM ausgewählt wurden
	15	Linienbreite des Elements
	201	Farbe des Elements
	301	Linienart des Elements
	403	Art des Elements
	900	Erster Infostring des Elements
	901	Nächster Infostring des Elements
ARC (Bogen)	3	Radius
	4	Anfangswinkel
	5	Endwinkel
	101	Mittelpunkt
	102	Anfangspunkt auf Umfang
	103	Endpunkt auf Umfang

(Tabelle wird fortgesetzt)

Elementart	inq_index	Funktionswert
BSPLINE (B-Spline)	101	Anfangspunkt
	102	Endpunkt
		Weitere Daten werden in die systemdefinierte Logische "Tabelle BSpline_basic_table" geschrieben (-> Kap.15.2 ff). Die Tabelle enthält folgende Daten: Kopffeld 1 : Ordnung des B-Splines Kopffeld 2 : Länge des B-Splines Kopffeld 3 : Anzahl der Interpolationspunkte Kopffeld 4 : Länge des Knotenvektors Kopffeld 5 : Geschlossen-Flag 0 : offen, -1 : geschlossen_periodisch 1 : geschlossen_nichtperiodisch Datenspalte 1 : X-Koordinaten der Stützpunkte Datenspalte 2 : Y-Koordinaten der Stützpunkte Datenspalte 3 : X-Koordinaten der Interpolationspunkte Datenspalte 4 : Y-Koordinaten der Interpolationspunkte Datenspalte 5 : Anzahl im Knotenvektor
CIRCLE (Kreis)	3	Radius
	101	Mittelpunkt
	102	Umfangspunkt (= Scheitelpunkt)
CENTERLINE (Mittellinie)	101	Mittelpunkt
	102	Anfangspunkt für Mittellinie 1
	103	Endpunkt für Mittellinie 1
	104	Anfangspunkt für Mittellinie 2
	105	Endpunkt für Mittellinie 2
C_CIRCLE (Hilfskreis)	101	Mittelpunkt
	3	Radius
C_LINE (Hilfslinie)	101	Mittelpunkt (= Definitionspunkt)
	3	Winkel zur X-Achse
Bemaßung : DIM_LINE DIM_RADIUS DIM_DIAMETER DIM_ARC DIM_ANGLE DIM_CHAIN DIM_DATUM_SHORT DIM_DATUM_LONG DIM_COORD DIM_CHAMFER	3 4 5 6 7 8 11 - 307	Maßzahl [mm bzw. Grad] Oberer Toleranzwert [mm bzw. Grad] Unterer Toleranzwert [mm bzw. Grad] Editierstatus der Primär- oder Sekundärmaßzahl (1 : editiert, 0 : nicht editiert) Editierstatus oberer Toleranzwert (1 : editiert, 0 : nicht editiert) Editierstatus unterer Toleranzwert (1 : editiert, 0 : nicht editiert) Weitere Abfrageindices und ihre Bedeutung können der Help-Datei entnommen werden
FILLET (Rundung)	3	Radius
	4	Anfangswinkel
	5	Endwinkel
	101	Mittelpunkt
	102	Anfangspunkt auf Umfang
	103	Endpunkt auf Umfang
LEADERLINE (Hinweislinie)	3	Pfeilgröße der Hinweislinie
	601	Art der Linienbegrenzung (z.B. ARROW_TYPE)

(Tabelle wird fortgesetzt)

Elementart	inq_index	Funktionswert
LINE (Linie)	101	Anfangspunkt
	102	Endpunkt
POINT (Punkt)	101	Punkt
SYMLINE (Symmetrielinie)	3	Radius (bei kreisförmiger Symmetrielinie)
	4	Anfangswinkel (bei kreisförmiger Symmetrielinie)
	5	Endwinkel (bei kreisförmiger Symmetrielinie)
	6	Art der Symmetrielinie (0 : gerade, 1 : kreisf.)
	101	Mittelpunkt bei kreisförmiger oder Anfangspunkt bei gerader Symmetrielinie
	102	Anfangspunkt bei kreisförmiger oder Endpunkt bei gerader Symmetrielinie
	103	Endpunkt bei kreisförmiger Symmetrielinie
TEXT	3	Referenzpunktlage (1 - 9)
	4	Zeilenabstandsfaktor
	5	Breite/Höhe-Verhältnis
	6	Texthöhe
	7	Textneigung
	8	Zeilen-Neigungswinkel
	101	Textursprungspunkt
	302	Textfont
	601	Textrahmen (OFF, BOX, BALLON)
	602	Textfüllung (OFF, ON)
	902	Erster Textstring
	903	Nächster Textstring
Keine der vorgenannten Arten	403	END

Beispiel: Makro COLANI ermöglicht eine Abfrage der Daten von Kreisbögen und Rundungen.

```
DEFINE Colani
{Abfragemakro für Kreisbögen und Rundungen}
{Fischer, 08.03.91, Stand 30.06.93}

  LOCAL P LOCAL Art

  LOOP
    READ PNT 'Bitte Element identifizieren' P
    INQ_ELEM P
    LET Art (INQ 403)
    IF (Art = END)
      DISPLAY 'Kein abfragbares Element gefunden'
     ELSE_IF (Art = ARC)
      DISPLAY 'Sie haben einen Kreisbogen identifiziert'
      DISPLAY ('Farbe : ' + STR (INQ 201))
      DISPLAY ('Linienart : ' + INQ 301)
      DISPLAY ('Radius : ' + STR (INQ 3))
      DISPLAY ('Anfangswinkel : ' + STR (INQ 4))
      DISPLAY ('Endwinkel : ' + STR (INQ 5))
      DISPLAY ('Mittelpunkt : ' + STR (INQ 101))
      DISPLAY ('Anfangspunkt auf Umfang : ' + STR (INQ 102))
      DISPLAY ('Endpunkt auf Umfang : ' + STR (INQ 103))
     ELSE_IF (Art = FILLET)
      DISPLAY 'Sie haben eine Rundung identifiziert'
      DISPLAY ('Farbe : ' + STR (INQ 201))
      DISPLAY ('Linienart : ' + INQ 301)
      DISPLAY ('Radius : ' + STR (INQ 3))
      DISPLAY ('Anfangswinkel : ' + STR (INQ 4))
      DISPLAY ('Endwinkel : ' + STR (INQ 5))
      DISPLAY ('Mittelpunkt : ' + STR (INQ 101))
      DISPLAY ('Anfangspunkt auf Umfang : ' + STR (INQ 102))
      DISPLAY ('Endpunkt auf Umfang : ' + STR (INQ 103))
    ELSE
      DISPLAY 'Das Element ist kein Kreisbogen und keine Rundung'
```

```
    END_IF
  END_LOOP
END_DEFINE
```

13.1.2 Elementdatenermittlung für mehrere Elemente

Schritt 1: Auswählen der Elemente und Eintrag der Daten des ersten Elements in die Abfrageliste mit Hilfe der Interruptfunktion INQ_SELECTED_ELEM

INQ_ELEM (Interruptfunktion)

```
-->(INQ_SELECTED_ELEM)-->|Auswahl|-->
```

Es sind alle vom interaktiven Arbeiten bekannten Auswahlmethoden (Auswahlrahmen, Auswahlpolygon, Auswahlprozeß) zulässig.

Schritt 2: Auslesen der Elementdaten

Die Anzahl der erfaßten Elemente läßt sich mit Hilfe von "INQ 14" (siehe vorangegangenes Kapitel, Schritt 2) bestimmen. Da die Daten des ersten erfaßten Elements bereits in die Abfrageliste eingetragen sind, lassen sie sich, wie im vorangegangen Kapitel beschrieben, sofort mit Hilfe der Integrierten Funktion INQ auslesen. Die Daten des nächsten Elements können dann mit Hilfe der Interruptfunktion INQ_NEXT_ELEM in die Liste eingetragen und wiederum mit INQ ausgelesen werden. Bei den restlichen Elemente wird in gleicher Weise verfahren.

INQ_NEXT_ELEM (Integrierte Funktion)

```
-->(INQ_NEXT_ELEM)-->
```

13.2 Abfrage von Teiledaten

Bei den Teile kann es sich um eigenständige Teile, um mehrfach abgebildete Teile oder um die Darstellung einer Einzelheit handeln. Die Teiledaten werden mit Hilfe der Interruptfunktion INQ_PART in die Abfrageliste eingetragen.

INQ_PART (Integrierte Funktion)

```
-->(INQ_PART)-->+------>------+-->+-->|Punkt|------+-->
                |             |   |                |
                `-->(DETAIL)--'   `-->|Teilename|--'
```

Bei interaktiver Anwendung (also außerhalb eines Makros) schreibt INQ_PART den Vergrößerungsfaktor des Teils in die Eingabezeile. In einem Makro unterbleibt diese Rückgabe. Mit der Option DETAIL läßt sich feststellen, ob das ausgewählte Teil eine Einzelheit ist. Falls es sich um eine Einzelheit handelt, wird bei interaktiver Anwendung der Vergrößerungsfaktor zurückgegeben. Handelt es sich um keine Einzelheit, erfolgt eine Fehlermeldung und die Rückgabe eines Vergrößerungsfaktors von "0".

Nachdem die Teiledaten in die Abfrageliste eingetragen wurden, können die Daten mit Hilfe der Integrierten Funktion INQ ausgelesen werden (--> Kapitel 13.1.1). Dabei finden folgende Abfrageindices Verwendung:

inq_index	Funktionswert
1	140 (impliziter inq_code)
2	Info über die verwendete Option 1 : DETAIL 0 : keine Option
3	Vergrößerungsfaktor des Teils
4	Zeichnungsmaßstab des Teils
5	Zählwert des Kunden-Teilenamens
101	Bezugspunkt des Teils
103	Translationsvektor vom Ursprung des Teils zum Ursprung des globalen Koordinatensystems
301	Kunden-Teilename
302	Eindeutiger Teilename
900	Erste Zeichenfolge des eindeutigen Teilenamens
901	Nächste Zeichenfolge des eindeutigen Teilenamens (wiederholte Verwendung liefert eine Liste aller eindeutigen Teilenamen, die zum Kundenteilenamen gehören)

13.3 Abfrage von Systemeinstellungen

Die Abfrage von Systemeinstellungen geschieht in zwei Schritten:

Schritt 1: Eintrag der Daten in die Abfrageliste durch Aufruf der Interruptfunktionen INQ_ENV

INQ_ENV (Interruptfunktion)

-->(INQ_ENV)-->|inq_code|-->

Der Abfragecode (inq_code = Inquiry Code) bestimmt, welche Daten ermittelt und in die Abfrageliste eingetragen werden.

Schritt 2: Auslesen der interessierenden Daten mit Hilfe der Abfragefunktion INQ

INQ (Integrierte Funktion)

-->(INQ)-->|inq_index|-->

Je nach Abfragecode (inq_code) und Abfrageindex (inq_index) liefert INQ folgendes Ergebnis:

inq_code	inq_index	Funktionswert
0	1	0 (inq_code)
	2	ME10-Versionsnummer
	301	ME10-Versionsbezeichnung
	302	Aktuelles Verzeichnis
1	1	1 (inq_code)
	2	Aktuelles Fenster, Bereich 0 bis 16
	5	Bildschirmnummer des aktuellen Fensters
	101	Untere linke Koordinaten des akt. Fensters in Pixel
	102	Obere rechte Koordinaten des akt. Fensters in Pixel
	301	Zeichenfolge, die dem erstellten Fenster entspricht
2	1	2 (inq_code)
	101	Unterer linker Eckpunkt des aktuellen Fensters in aktuelle Einheiten
	102	Oberer rechter Eckpunkt des aktuellen Fensters in aktuellen Einheiten
3	1	3 (inq_code)
	2	Aktueller Fangbereich beim Digitalisieren in Pixel
	3	Aktueller Fangbereich bei Koordinatenangaben in akt.Einh.
	10	Aktuelle Stiftbreite der Geometrie
	11	Aktuelle Stiftbreite der Maßlinie
	12	Aktuelle Linienbreite der Geometrie
	201	Aktuelle Farbe der Geometrie
	202	Aktuelle Farbe der Hilfsgeometrie
	203	Aktuelle Textfarbe
	204	Aktuelle Schraffurfarbe
	205	Aktuelle Bemaßungsfarbe
	206	Aktuelle Farbe der Maßangaben
	301	Aktuelle Linienart der Geometrie
	302	Aktuelle Linienart der Hilfsgeometrie
	304	Aktuelle Schraffurlinienart
	603	Aktueller Fangmodus
4	1	4 (inq_code)
	301	Plottername
5	1	5 (inq_code)
	2	1 wenn Plotterabfrage positiv, andernfalls 0
	3	Breite des Plotterfensters
	4	Höhe des Plotterfensters
6	1	6 (inq_code)
	2	Größe der aktuellen Längeneinheit der X-Achse in mm
	3	Größe der aktuellen Winkeleinheit in RAD
	4	Zeichnungsmaßstab
	5	Trennmodus
	6	Spline-Abschnittmodus
	8	Aktueller Wert von PART_DRW_SCALE_REF 0 : REF_PNT 1 : CENTER 2 : PART_ORIGIN
	9	Aktueller Wert von CS_SYMBOL 0 : OFF 1 : ON
	10	Aktueller Wert von PARTS_LIST_FORMAT 0 : STANDARD 1 : ENHANCED
	301	Letzter generierter eindeutiger Teilename
7	1	7 (inq_code)
	4	Zeichnungsmaßstab des aktuellen Teils
	101	Unterer linker Punkt des aktuellen Teils
	102	Oberer rechter Punkt des aktuellen Teils
	103	Unterer linker Punkt des aktuellen Teils
	104	Oberer rechter Punkt des aktuellen Teils
	105	Translationsvektor vom Ursprung des aktuellen Teils zum Ursprung des globalen Koordinatensystems zum
	301	Teilename des aktuellen Teils
	302	Eindeutiger Teilename des aktuellen Teils

(Tabelle wird fortgesetzt)

inq_code	inq_index	Funktionswert
8	1	8 (inq_code)
	2	Anzahl der geladenen Zeichensätze
	3	Anzahl der global festgelegten Infostrings
	4	Anzahl der aktuell festgelegten Infostrings
	5	Höchste verwendete eindeutige Teilenummer
	301	Name des aktuellen Zeichensatzes
9	1	9 (inq_code)
	2	Unterlänge des aktuellen Zeichensatzes in Gitterpunkten
	3	Schriftzeichenhöhe des aktuellen Zeichensatzes
	4	Schriftzeichenbreite des aktuellen Zeichensatzes
	301	Name eines geladenen Zeichensatzes *)
	302	Text eines in der Zeichnung verwendeten Infos *)
	303	Text eines aktuellen Infos *)
10	1	10 (inq_code)
	2	Bildschirmtyp (siehe Help-Datei, Stichwort INQ_ENV)
	3	Tablett-Typ (siehe Help-Datei, Stichwort INQ_ENV)
	4	Betriebssystem 1 : Pascal Workstation 2 : HP-UX 3 : DOS 4 : SunOS 5 : NEC EWS-UX 6 : MS-WINDOWS 3.x 7 : MS-WINDOWS NT 8 : MS-WINDOWS 4.x (WINDOWS 95)
	6	ME PE-Look aktiviert (0 : nein, 1 : ja)
	7	Mehrere Fenster aktiviert (0 : nein, 1 : ja)
	8	Anzahl der angeschlossenen Bildschirme
	9	Nummer des aktuellen Bildschirms
	10	Typ eines angeschlossenen Drehknopfmoduls (-->Help-Datei)
	101	untere linke Koordinate des Grafikbereichs
	102	obere rechte Koordinate des Grafikbereichs
	103	Breite und Höhe des Pixelfonts (Schriftart)
12	1	12 (inq_code)
	3	Aktuelle Text-Referenzpunktlage (1 - 9)
	4	Aktueller Text-Zeilenabstandsfaktor
	5	Aktuelles Breite/Höhe-Verhältnis für Text
	6	Aktuelle Texthöhe
	7	Aktuelle Textneigung
	8	Aktueller Textzeilen-Neigungswinkel
	302	Aktueller Textfont
	601	Aktueller Textrahmen (OFF, BOX, BALLON)
	602	Aktuelle Textfüllung (OFF, ON)
13	1	13 (inq_code)
	2	Aktuelle Wert von DIM_UNDERLINE_EDITED (0 : OFF 1 : ON)
	3	Aktuelle Wert von DIM_CATCH_LINES (0 : OFF 1 : ON)
	4	Aktuelle Wert von DA_DIM_AUTO_STRATEGY (--> Help-Datei)
	5	Aktuelle Wert von DA_DIM_AUTO_POSITION 0 : BELOW 1 : ABOVE

*) : Muß in Verbindung mit INQ_ENV 8 verwendet werden.
Der auf das Objekt verwendete Zeiger wird durch INQ_ENV 8 vor den ersten Wert und durch INQ_ENV 9 auf den jeweils nächsten Wert positioniert.
Nach dem letzten Wert wird 'END-OF-LIST' zurückgegeben.

Beispiel für INQ_303 (für INQ_301 und INQ 302 gilt das gleiche):
INQ_ENV 8 INQ_ENV 9 DISPLAY (INQ 303) liefert den ersten aktuellen Infotext
INQ_ENV 9 DISPLAY (INQ 303) liefert den nächsten aktuellen Infotext
INQ_ENV 9 DISPLAY (INQ 303) liefert den übernächsten aktuellen Infotext
usw.

Beispiel: Makro "Status" informiert den Benutzer über die aktuelle ME10-Version und den Typ des definierten Plotters.

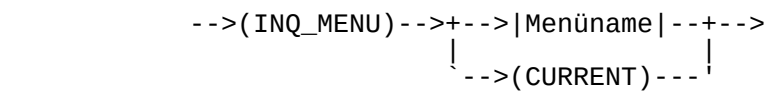
```
DEFINE Status
  INQ_ENV 0
  DISPLAY ('Sie arbeiten mit HP-ME10 ' + INQ 301)
  INQ_ENV 4
  DISPLAY ('Definiert ist folgender Plotter : ' + INQ 301)
END_DEFINE
```

13.4 Abfrage von Information zu Bildschirmmenüs und Anzeigetabellen

Bildschirmmenüs werden in Kapitel 14, Anzeigetabellen in Kapitel 15 behandelt.

Daten von Bildschirmmenüs werden mit Hilfe der Interruptfunktion INQ_MENU in die Abfrageliste eingetragen und können mit der integrierten Funktion INQ ausgelesen werden.

INQ_MENU (Interruptfunktion)



ME10 bietet die Möglichkeit, mehrere Bildschirmmenüs zu definieren und nebeneinander zu verwenden. Ein Bildschirmmenü ist dabei aktuell, die übrigen sind nicht aktuell. Wenn mehrere Menüdefinitionen nebeneinander existieren, müssen sie sich durch ihre Namen voneinader unterscheiden.

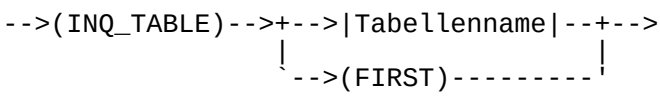
Mit dem Parameter "Menüname" lassen sich die Daten eines bestimmten, durch den Namen gekennzeichneten Menüs in die Abfrageliste eintragen. Der Menüname ist dabei in Hochkomma einzuschließen. Die Angabe des Parameters CURRENT bewirkt ein Eintragen von Daten des aktuellen Menüs (--> Kap. 14.3.2).

Nachdem die Menüdaten in die Abfrageliste eingetragen wurden, können die Daten mit Hilfe der Integrierten Funktion INQ ausgelesen werden (--> Kapitel 13.1.1). Dabei finden folgende Abfrageindices Verwendung:

inq_index	Funktionswert
1	131 (impliziter inq_code)
2	1 : Ursprung oben (UPPER)
	0 : Ursprung unten (LOWER)
3	1 : Ursprung links (LEFT)
	0 : Ursprung rechts (RIGHT)
4	1 : Menü ist fixiert
	0 : Menü ist beweglich
5	1 : Menü ist abgebildet
	0 : Menü ist nicht abgebildet
6	1 : Menü ist abfragebereit
	0 : Menü ist nicht abfragebereit
8	Nummer des Stammbildschirms
9	Nummer des aktuellen Bildschirms
10	Anzahl der vorhandenen Menüs
101	Stammrahmen unten links
102	Stammrahmen oben rechts
103	Aktueller Rahmen unten links
104	Aktueller Rahmen oben rechts
301	Menüname
900	Name des ersten Menüs in der Liste der vorhandenen Menüs
901	Name des nächsten Menüs in der Liste der vorhandenen Menüs

Daten von Anzeigetabellen werden mit Hilfe der Interruptfunktion INQ_TABLE in die Abfrageliste eingetragen und können mit der integrierten Funktion INQ ausgelesen werden.

INQ_TABLE (Interruptfunktion)



ME10 bietet die Möglichkeit, mehrere Bildschirmmenüs zu definieren und nebeneinander zu verwenden. Ein Bildschirmmenü ist dabei aktuell, die übrigen sind nicht aktuell. Wenn mehrere Menüdefinitionen nebeneinander existieren, müssen sie sich durch ihre Namen voneinader unterscheiden.

Mit dem Parameter "Tabellenname" lassen sich die Daten einer bestimmten, durch den Namen gekennzeichneten Anzeigetabelle in die Abfrageliste eintragen. Der Tabellenname ist dabei in Hochkomma einzuschließen. Die Angabe des Parameters "FIRST" bewirkt ein Eintragen von Daten der in einer internen Liste an erster Position aufgeführten Tabelle.

Nachdem die Tabellendaten in die Abfrageliste eingetragen wurden, können die Daten mit Hilfe der Integrierten Funktion INQ ausgelesen werden (--> Kapitel 13.1.1). Dabei finden folgende Abfrageindices Verwendung:

inq_index	Funktionswert
1	131 (impliziter inq_code)
2	1 : Ursprung oben (UPPER)
	0 : Ursprung unten (LOWER)
3	1 : Ursprung links (LEFT)
	0 : Ursprung rechts (RIGHT)
4	1 : Tabelle ist fixiert
	0 : Tabelle ist beweglich
5	1 : Tabelle ist abgebildet
	0 : Tabelle ist nicht abgebildet
6	1 : Tabelle ist abfragebereit
	0 : Tabelle ist nicht abfragebereit
8	Nummer des Stammbildschirms
9	Nummer des aktuellen Bildschirms
10	Anzahl der vorhandenen Anzeigetabellen
101	Stammrahmen unten links
102	Stammrahmen oben rechts
103	Aktueller Rahmen unten links
104	Aktueller Rahmen oben rechts
301	Tabellenname
900	Name der ersten Tabelle in der Liste der vorhandenen Anzeigetabellen
901	Name der nächsten Tabelle in der Liste der vorhandenen Anzeigetabellen

13.5 Abfrage von Information zu vorausgegangenen Systemaktionen

Bei der Ausführung von zahlreichen Befehlen und Interruptfunktionen werden automatisch Einträge in die systeminterne Abfrageliste vorgenommen. Leider sind nicht alle Abfragemöglichkeiten dokumentiert. Folgende Einträge werden beschrieben:

- Letzter Meßwert
- Vergrößerungsfaktor eines Teils
- Zuletzt benutztes Digitalisierfenster
- Dateiformat der zuletzt geladenen Zeichnung
- Einträge des Hidden-Line-Moduls

Letzter Meßwert

Beim Aufruf einer der Meßfunktionen MEASURE_ANGLE, MEASURE_AREA, MEASURE_COORDINATE, MEASURE_DISTANCE, MEASURE_LENGTH und MEASURE_RADIUS wird das Meßergebnis automatisch in die Abfrageliste eingetragen und kann danach mit der Abfragefunktion INQ (siehe Kapitel 13.1) abgefragt werden. Je nach verwendeter Meßfunktion und Abfrageindex (inq_index) liefert INQ folgendes Ergebnis:

Meßfunktion	inq_index	Funktionswert
MEASURE_ANGLE	1	104 (impliziter inq_code)
	2	Letzter gemessener Winkel
MEASURE_AREA	1	105 (impliziter inq_code)
	2	Letzte gemessene Fläche
MEASURE_COORDINATE	1	101 (impliziter inq_code)
	101	Punkt
MEASURE_DISTANCE	1	102 (impliziter inq_code)
	2	0 bei Abstandsmessung direkt 1 bei Abstandsmessung horizontal 2 bei Abstandsmessung vertikal
	3	Letzter gemessener Abstand
MEASURE_LENGTH	1	103 (impliziter inq_code)
	2	Letzte gemessene Strecke
MEASURE_RADIUS	1	106 (impliziter inq_code)
	2	Letzter gemessener Radius

Letztes Digitalisierfenster

Die Digitalisierung eines Punktes unter Verwendung der Dialogfunktion READ bewirkt automatisch einen Eintrag der Nummer des dabei verwendeten Bildschirmfensters in die Abfrageliste. Die Daten können mit der Abfragefunktion INQ (siehe Kapitel 13.1) abgefragt werden. Je nach Abfrageindex (inq_index) liefert INQ folgendes Ergebnis:

inq_index	Funktionswert
1	110 (impliziter inq_code)
2	Digitalisierfenster (0..16)
301	Name des Fensters

Dateiformat der zuletzt geladenen Zeichnung

Beim Laden einer Zeichnung wird Information über das Dateiformat in die systeminterne Abfrageliste eingetragen. Sie kann über den Abfrageindex 301 abgefragt werden.

Einträge des Hidden-Line-Moduls

Der Hidden-Line-Modul ermöglicht das Erzeugen von verdeckten Kanten. Er ist derzeit nicht für ME10d verfügbar. Die von diesem Modul automatisch vorgenommenen Einträge in die systeminterne Abfrageliste werden im Abschnitt HIDDEN2D der Helpdatei beschrieben.

14 Systemanpassung

Vorbemerkung: In diesem Kapitel werden Zugriffspfade und Namen von Dateien genannt, die beim Einrichten des Systems eine Rolle spielen. Dabei finden die für HP-UX-Systeme gültigen Bezeichnungen Verwendung. Diese Bezeichnungen sind für andere Betriebssysteme meist sehr ähnlich, aber nicht identisch und müssen dann sinngemäß verwendet werden.

ME10 wird beim Systemstart in einer Weise eingerichtet, die durchschnittlichen Benutzerwünschen Rechnung trägt. Hierzu werden zahlreiche Dateien mit Anweisungen und Makros zum Einrichten des Systems geladen. Das automatische Laden dieser Dateien wird durch entsprechende Einträge in der Startdatei "startup" gesteuert. "startup" befindet sich im gleichen Verzeichnis wie die ME10-Programmdateien und wird beim Systemstart automatisch ausgeführt. ME10 läßt sich jedoch auf sehr einfache Weise speziellen Benutzerwünschen anpassen, indem die das System einrichtenden Dateien verändert oder neue Einrichtdateien erstellt werden. Ein automatisches Anpassen beim Systemstart läßt sich wie folgt erreichen:

Die Startdatei "startup" enthält als letzten Eintrag eine Anweisung zum Laden einer Datei, in der spezielle Anweisungen des Benutzers enthalten sein können.

```
{INPUT 'customize'}
```

Standardmäßig ist die Anweisung durch geschweifte Klammern wirkungslos gemacht. Der Benutzer kann die geschweiften Klammern entfernen und eine Datei anlegen, in der die von ihm gewünschten Einrichtanweisungen enthalten sind.

In diesem Kapitel werden die Anpassung der Systemumgebung, die Funktionstastenbelegung und die Anpassung der Tablett- und Bildschirmmenüs beschrieben. Anzeigetabellen, die ebenfalls als Menüs verwendet werden können, sind Thema von Kapitel 15.

Über Kapitel 14 hinausgehende Informationen lassen sich bei einem gründlichen Studium der durch die startup-Datei geladenen Dateien gewinnen. Besonders zu erwähnen ist hierbei das Makro Configure_system. Jeder Anwender sollte sich jedoch darüber in Klaren sein, daß Änderungen der ME10-Konfiguration sehr gute Systemkenntnisse erfordern. Geändert werden sollten grundsätzlich nie Originaldateien, sondern nur Kopien davon. Für Testzwecke ist es auch oft ausreichend, lediglich die im Hauptspeicher befindlichen Makros mit EDIT_MACRO zu ändern. Ein Neustart von ME10 stellt dann notfalls wieder der ursprünglichen Systemzustand her.

14.1 Anpassen der Systemumgebung

Als Systemumgebung werden die aktuellen Einstellungen der Systemparameter (wie z.B. Farbe und Linienart von Geometrie, Hilfsgeometrie und Bemaßung) bezeichnet. Die Systemumgebung wird beim Systemstart durch Laden einer Datei voreingestellt, in der ca. 180 Einstellanweisungen enthalten sind. Die Datei kann danach vom Benutzer den jeweiligen Erfordernissen entsprechend verändert werden. Einzelne Größen der Systemumgebung lassen sich mit Hilfe der im Tablettmenü und in den Bildschirmmenüs aufgeführten Einstellfunktionen verändern. So ist z.B. die aktuelle Texthöhe über den Menüpunkt "SETZEN Größe" des Bildschirmmenüs TEXT 1 einstellbar. Die Interruptfunktion EDIT_ENVIRONMENT (Menüsystem: STANDARDS -> KONFIG EDI) erlaubt ein Ändern aller aktuellen Einstellungen mit Hilfe des ME10-Texteditors.

EDIT_ENVIRONMENT (Interruptfunktion)

-->(EDIT_ENVIRONMENT)-->

Auf dem Textbildschirm erscheint eine Datei, in der die aktuellen Systemeinstellungen definiert sind. Nachfolgend ist ein Auszug aus einer Umgebungsdatei dargestellt.

```
SEARCH '/usr/PE/me10'
       '/usr/PE/privat'
END
MAX_FEEDBACK 100
CONFIGURE_EDITOR '$' 1 79
UNITS 1 MM
UNITS 1 DEG
CS_REF_PT 0,0
.
.
.
```

Änderungen können mit Hilfe der Editorbefehle (-> Kap. 3) vorgenommen und durch gleichzeitiges Betätigen der Tasten <CTRL> und <D> für die aktuelle Sitzung gespeichert werden. Sie gelten bis zu ihrer Neudefinition. Nach Beendigung von ME10 sind die Änderungen verloren. Eine dauerhafte Speicherung ist mit Hilfe des Editorbefehls "W" oder mit Hilfe der Interruptfunktion SAVE_ENVIRONMENT möglich (Menüsystem: STANDARDS ->KONFIG SP).

SAVE_ENVIRONMENT (Interruptfunktion)

-->(SAVE_ENVIRONMENT)-->|Ausgabeziel|-->

SAVE_ENVIRONMENT überträgt die aktuellen Einstellungen der Systemumgebung im ASCII-Format an das angegebene Ausgabeziel (-> Kap.2.1.3). Beim Laden einer aus diese Weise erzeugten Datei mit INPUT werden die darin enthaltenen Einstellanweisungen ausgeführt.

14.2 Belegung der Funktionstasten

Die Funktionstasten Nr. 1 bis 8 können mit Hilfe der Interruptfunktion DEFINE_KEY belegt werden.

DEFINE_KEY (Interruptfunktion)

-->(DEFINE_KEY)-->|Tastennummer|-->|Belegungstext|-->

|Tastennummer| gibt die Nummer der zu belegenden Funktionstaste an. Es sind Zahlen zwischen 1 und 8 zulässig. Die Tasten werden mit einem Text (|Belegungstext|) belegt. Beim Betätigen der Taste erfolgt einen Auswertung des Belegungstextes (im Prinzip eine Anwendung der Stringfunktion VAL auf den Belegungstext) und eine Übergabe an das System. Funktionstasten können grundsätzlich mit allen von ME10 zugelassenen Daten (-> Kap.2.7) belegt werden. Die Bezeichnung |Belegungstext| bedeutet, daß die Daten zunächst in Textform, also als Stringkonstanten, Stringvariablen oder Stringausdrücke, anzugeben sind, um eine sofortige Auswertung zu unterdrücken. Sie bedeutet nicht, daß nur eine Belegung mit Text möglich ist. (Die Unterdrückung der sofortigen Auswertung einer Anweisung durch Angabe als Text wird als Maskierung bezeichnet und findet zum Beispiel auch bei UNIX-Befehlen Anwendung.) In Kapitel 14.3.2 wird die Bildung von Belegungstext für Bildschirmmenüs eingehend behandelt. Die Ausführungen gelten sinngemäß auch für die Belegung von Funktionstasten.

Häufig werden Funktionstasten mit Sonderzeichen belegt, die nicht auf der Tastatur enthalten sind. Beispiele:

```
DEFINE_KEY 1 #242 {Belegung von F1 mit Grad-Zeichen}
DEFINE_KEY 2 #243 {Belegung von F2 mit Durchmesser-Zeichen}
DEFINE_KEY 3 #254 {Belegung von F3 mit Plus/Minus-Zeichen}
```

Hochkommas sind hier nicht notwendig, da das Symbol # einen nachfolgenden ASCII-Wert kennzeichnet.

Die Funktionstasten können, wie bereits erwähnt, auch mit Aufrufkommandos für ME10-Befehle, Interruptfunktionen oder Makros belegt werden. Beim Betätigen einer Taste wird der Belegungstext zunächst nur in der Eingabezeile angezeigt. Das Drücken der Enter-Taste bewirkt eine Übergabe an das System. Soll der Belegungstext ohne zusätzliche Betätigung der Enter-Taste an das System übergeben werden, so ist er mit dem <CR>-Zeichen "CHR 13" zu verketten.

Beispiel

Bei Betätigung der Funktionstaste F5 soll ein Bildneuaufbau erfolgen.

```
Anweisung                          |Wirkung beim Betätigen der Taste F5
-----------------------------------+-----------------------------------------
DEFINE_KEY 5 'REDRAW'              |In der Eingabezeile erscheint REDRAW.
                                   |Durch Betätigung der Enter-Taste werden
                                   |die Eingabe an das System übergeben und
                                   |ein Bildneuaufbau durchgeführt.
-----------------------------------+-----------------------------------------
DEFINE_KEY 5 ('REDRAW' + CHR 13)   |REDRAW wird sofort an das System über-
                                   |geben. Der Bildneuaufbau erfolgt sofort.
-----------------------------------+-----------------------------------------
```

Beispiel

Folgende Tastenbelegungen können bei der Entwicklung von Makros nützlich sein:

```
DEFINE_KEY 1 ('EDIT_FILE Testfilename' + CHR 13)
DEFINE_KEY 2 ('INPUT Testfilename' + CHR 13)
DEFINE_KEY 3 (('ENTER (VAL Testmacname)' + CHR 13) + CHR 13)
DEFINE_KEY 4 ('Def_test' + CHR 13)
DEFINE_KEY 7 ('TRACE DEL_OLD "/usr/tmp/me10_trace"' + CHR 13)
DEFINE_KEY 8 ('TRACE OFF EDIT_FILE "/usr/tmp/me10_trace"' + CHR 13)
```

Für Dateien und Zugriffspfade wurden die für HP-UX-Systeme geeigneten Namen gewählt. Bei anderen Betriebssystemen müßten die Namen gegebenenfalls angepaßt werden.

Die Funktionstastenbelegungen der Tasten F1, F2 und F3 enthalten Textvariablen, denen zunächst noch ein Wert zugewiesen werden muß. "Testfilename" soll hierbei den Namen der zu editierenden Textdatei enthalten (also der Datei, welche die zu entwickelnden Makros enthält), "Testmacname" den Namen des Makros, das beim Betätigen der Funktionstaste F3 gestartet werden soll. Das nachfolgend aufgeführte Makro "Def_test" ermöglicht dem Benutzer eine Angabe dieser Werte.

```
DEFINE Def_test
 READ STRING 'Name des Testfiles (in Hochkomma eingeben)' Testfilename
 READ STRING 'Name des Testmakros (in Hochkomma eingeben)?' Testmacname
END_DEFINE
```

Bei Betätigung der Funktionstaste F1 wird eine durch die Variable "Testfilename" gekennzeichnete Datei in den Editor, bei Betätigung der Funktionstaste F2 in den Hauptspeicher geladen. Funktionstaste F3 ist mit dem Aufrufkommando des durch die Variable "Testmacname" gekennzeichneten Makros belegt. Funktionstaste F4 ruft das Makro "Def_test" auf, in dem eine Wertzuweisung für die Variablen "Testfilename" und "Testmacname" stattfindet. Die Funktionstasten F5 und F6 sind nicht belegt. Die Funktionstasten F7 und F8 dienen der Fehlersuche mittels einer TRACE-Datei. Bei Betätigung der Funktionstaste F7 wird eine TRACE-Datei angelegt. Funktionstaste F8 schließt diese Datei und zeigt sie am Bildschirm an.

Die Belegungsanweisungen und das Makro "Def_test" werden zweckmäßigerweise in eine Datei geschrieben und bei Bedarf mit INPUT geladen. Eine Tastaturschablone schafft zusätzlichen Komfort. Sie könnte wie folgt aussehen:

```
+-------------------------------------------------------------------------------+
|   F1    |   F2    |   F3    |   F4    |   F5    |   F6    |   F7    |   F9    |
|         |         |         |         |         |         |         |         |
|EDIT_FILE|  INPUT  |  Start  |Datei und|         |         |TRACE ON |TRACE OFF|
|         |         |         |Makro    |         |         |         |         |
|         |         |         |angeben  |         |         |         |         |
+-------------------------------------------------------------------------------+
```

14.3 Anpassen des Menüsystems

ME10 kann wahlweise mit Hilfe eines Grafiktabletts oder einer Maus bedient werden. Im ersten Fall besteht das Menüsystem aus einer Kombination aus Tablett- und Bildschirmmenü, im zweiten Fall aus einem reinen Bildschirmmenü, das in ein Bildschirm-Hauptmenü und ein Bildschirm-Nebenmenü unterteilt ist. Die Hauptmenüs der mausgesteuerten Benutzeroberfläche entsprechen den Bildschirmmenüs der tablettgesteuerten Benutzeroberfläche. Das Nebenmenü übernimmt bei der mausgesteuerten Benutzeroberfläche die Aufgaben des Tablettmenüs.

In beiden Fällen werden die Menüs durch Dateien definiert, die beim Laden ausgeführt werden oder Makrodefinitionen des Menüsystems enthalten. Durch Ändern oder Neuerstellung der Dateien kann der Benutzer das Menüsystem vollständig umgestalten. Nachfolgend wird eine Übersicht über die das Menüsystem definierenden Dateien gegeben. Die letzte Spalte der Tabelle zeigt, bei welcher Benutzeroberfläche die Dateien vewendet werden (Tablettversion, Mausversion oder beide Benutzeroberflächen).

Dateiname unter HP-UX	Art	Inhalt	Benutz.-Oberfl.
hp_tmenu	ASCII	Layout und Definition des Tablettmenüs	Tablett
tmenu_mac	Binär	Makros des Tablettmenüs und der Bildschirm-Nebenmenüs	beide
hp_menu_t	Binär	Layout von Bildschirm-Hauptmenüs	beide
hp_lay.mac	Binär	Layout von Bildschirm-Hauptmenüs und Bildschirmnebenmenüs	beide
hp_macro_t	Binär	Definitionen von Bildschirm-Hauptmenüs	beide
hp_menu_s	Binär	Layout, Definition und Makros von Bildschirm-Nebenmenüs	Maus
hp_init.mac	Binär	Makros zum Festlegen des gültigen Bildschirmmenü-Layouts	beide
hp_icon1.mac	Binär	Pictogramme von Bildschirmmenüs	beide
hp_icon2.mac	Binär	Pictogramme von Bildschirmmenüs	Maus
hp_icon3.mac	Binär	Pictogramme von Bildschirmmenüs	Maus
hp_icons.mac	ASCII	Pictogramme von Bildschirmmenüs	beide
hp_iconm.mac	ASCII	Pictogramme von Bildschirmmenüs	beide

Beim Systemstart werden die für die jeweilige Systemkonfiguration zutreffenden Dateien geladen. Die Startdatei "startup" enthält hierfür entsprechende Anweisungen. Das zuvor durch "startup" aufgerufene Makro Configure_system (bei HP-UX-Systemen in der Datei "hp_macro" enthalten) bestimmt, abhängig von der aktuellen Systemkonfiguration, welche Dateien geladen werden.

Wenn der Benutzer eine Anpassung des Menüsystems durch Ändern der obengenannten Dateien durchführen will, sollte er die Änderungen nicht an den Originaldateien, sondern an Kopien vornehmen. Handelt es sich bei der zu ändernden Datei um eine ASCII-Datei, kann sie durch EDIT_FILE geladen und mit Hilfe des Editorbefehls "W" unter einem anderen Namen gespeichert werden. Das Laden einer Binärdatei mit EDIT_FILE ist zwar

möglich, aber nicht sinnvoll, da der Inhalt vom Benutzer nicht gelesen und damit auch nicht editiert werden kann. Hier hilft ein kleiner Trick.

Zunächst werden alle im Hauptspeicher befindlichen Makros mit Hilfe der Anweisung DELETE_MACRO ALL aus dem Hauptspeicher gelöscht. Danach wird die gewünschte Binärdatei mit der Anweisung LOAD_MACRO *'Dateibezeichnung_1'* in den Hauptspeicher geladen. Mit der Anweisung SAVE_MACRO ALL *'Dateibezeichnung_2'* werden alle im Hauptspeicher befindlichen Makros im ASCII-Format in die durch "Dateibezeichnung_2" gekennzeichnete Datei geschrieben. Nachdem zuvor alle Makros gelöscht wurden, handelt es sich hierbei nur um die Makros der geladenen Datei.

Da im ersten Schritt alle Makros aus dem Hauptspeicher entfernt wurden, muß nachfolgend das System neu eingerichtet werden. Dies geschieht am einfachsten durch Verlassen und Neustart von ME10.

Beispiel: Die Makros der Binärdatei "tmenu_mac" sollen im ASCII-Format in Datei TMENU_ASC gespeichert werden.

```
1: DELETE_MACRO ALL
2: LOAD_MACRO 'tmenu_mac'
3: SAVE_MACRO ALL 'tmenu_asc'
4: System verlassen und neu starten
```

Selbstverständlich ist es auch möglich, die das Menüsystem definierenden Dateien völlig neu zu schreiben. Die Originaldateien können dabei als Anregung dienen.

14.3.1 Anpassung des Tablettmenüs

Standardmäßig wird das Tablettmenüsystem durch zwei Dateien definiert. Die Datei "hp_tmenu" enthält Anweisungen, mit denen eine Einteilung der Tablettfläche in Felder und eine Belegung der Felder mit Text vorgenommen wird. Bei dem Text handelt es sich um Makroaufrufe. HP_TMENU wird beim Systemstart mit INPUT geladen und ausgeführt. Die über das Tablettmenü aufrufbaren Makros enthalten Aufrufe von ME10-Befehlen oder ME10-Interruptfunktionen. Sie sind in Datei "tmenu_mac" enthalten und werden ebenfalls beim Systemstart geladen, aber erst beim Antippen des zugeordneten Tablettfeldes ausgeführt. Die Belegung der Tablettfelder mit Makroaufrufen ist zweckmäßig, aber nicht zwingend vorgeschrieben. Der Benutzer kann die Felder auch direkt mit Aufrufen für Befehle oder Interruptfunktionen belegen.

ME10 bietet die Möglichkeit, mehrere Tablettmenüs zu definieren und nebeneinander zu verwenden. Ein Tablettmenü ist dabei aktuell, die übrigen sind nicht aktuell. Die nachfolgend beschriebenen Anweisungen zur Menüdefinition beziehen sich stets auf das aktuelle Menü. Am Ende dieses Kapitels wird beschrieben, wie mehrere Tablettmenüs nebeneinander eingerichtet und verwendet werden können.

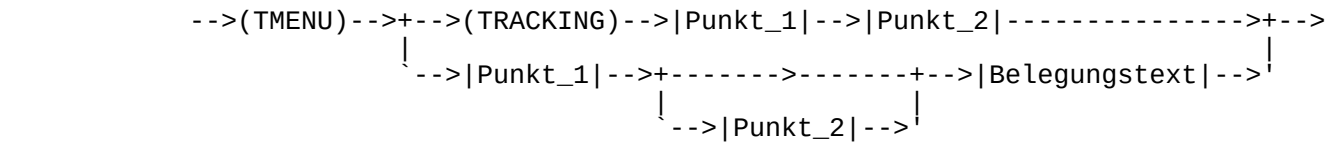
Das aktuelle Tablettmenü kann durch Ändern oder durch Neudefinition den Wünschen des Benutzers angepaßt werden. Bei einer Neudefinition ist es zweckmäßig, die aktuelle Definition zunächst mit Hilfe der Interruptfunktion DELETE_TMENU zu löschen.

DELETE_TMENU (Interruptfunktion)

```
-->(DELETE_TMENU)-->
```

Anweisungen zum Ändern der aktuellen Menüdefinition oder zur Neudefinition verwenden die Interruptfunktion TMENU. Sie können interaktiv eingegeben werden oder auch Bestandteile einer INPUT-Datei oder eines Makros sein.

TMENU (Interruptfunktion)



TMENU TRACKING definiert den Fadenkreuz-Führungsbereich des aktuellen Tablettmenüs. Hierzu müssen die linke untere Ecke (Punkt_1) und die rechte obere Ecke (Punkt_2) des Bereichs angegeben werden. Die Angabe kann in Millimetern (Bezugspunkt ist die linke untere Ecke der aktiven Tablettfläche) oder durch Auslösen des Tablettstifts an den entsprechenden Stellen der Tablettfläche erfolgen. Mit "TMENU *Punkt_1 Punkt_2 Belegungstext*" wird ein neues Tablettfeld durch Angabe der linken unteren Ecke (Punkt_1) und der rechten oberen Ecke (Punkt_2) definiert und mit einem Text belegt. Beim Auslösen des Tablettstifts auf dem Feld wird der Belegungstext ausgewertet (es erfolgt im Prinzip eine Anwendung der Stringfunktion VAL auf den Belegungstext) und an das System geschickt. Wurde ein Feld bereits definiert, genügt die Angabe eines Punktes im Feld. Auf diese Weise ist es z.B. beim interaktiven Arbeiten möglich, Felder des Tabletts vorübergehend neu zu belegen.

Menüfelder können grundsätzlich mit allen von ME10 zugelassenen Daten (-> Kap.2.7) belegt werden. Die Bezeichnung |Belegungstext| bedeutet, daß die Daten in Textform, also als Stringkonstanten, Stringvariablen oder Stringausdrücke, anzugeben sind, um eine sofortige Auswertung zu unterdrücken. Sie bedeutet nicht, daß nur eine Belegung mit Text möglich ist. Soll zum Beispiel ein Menüfeld mit der Zahl 125 belegt werden, so ist als Belegungstext '125' anzugeben. Wenn die Belegung hingegen mit dem Text 125 erfolgen soll, muß als Belegungstext '"125"' angegeben werden. Der Belegungstext 'LINE' bewirkt eine Belegung mit dem ME10-Befehl LINE, der Belegungstext '"LINE"' hingegen eine Belegung mit dem Text LINE. In Kapitel 14.3.2 wird die Bildung von Belegungstext für Bildschirmmenüs eingehend behandelt. Die Ausführungen gelten sinngemäß auch für die Belegung von Tablettfeldern.

Beispiel: Die nachfolgende Anweisungen definieren einen Fadenkreuz-Führungsbereich mit den Eckpunktskoordinaten x/y = 30/80 und 130/160 und belegen ein Feld mit den Eckpunktskoordinaten x/y = 115,160 und 130,170 mit dem Kommando zum Umschalten in die Betriebssystem-Shell.

```
TMENU TRACKING 30,80 130,160
TMENU 115,160 130,170 'RUN ""'
```

Anweisungen zur Definition des Tablettmenüs gelten bis zur Neudefinition oder bis zum Herunterfahren des Systems. Mit SAVE_TMENU kann die aktuelle Definition in einer Datei gespeichert werden. Sie ist dann mit INPUT jederzeit wieder aufrufbar.

SAVE_TMENU (Interruptfunktion)

```
-->(SAVE_TMENU)-->|Ausgabeziel|-->
```

SAVE_TMENU überträgt die aktuelle Tablettmenüdefinition im ASCII-Format an das angegebene Ausgabeziel (-> Kap.2.1.3). Nachfolgend wird ein Auszug aus Datei "hp_tmenu" wiedergegeben.

```
Tm_fc  {Verwenden Sie den richtigen Koeffizienten für Ihr aktuelles Tablett}
TMENU TRACKING Tm_cc 30,89   187,198
TMENU Tm_cc  30,252       50,263       'Tm_linetype_solid'
TMENU Tm_cc  30,241       50,252       'Tm_linetype_dashed'
TMENU Tm_cc  30,230       50,241       'Tm_linetype_dot_center'
TMENU Tm_cc  30,219       50,230       'Tm_linetype_phantom'
.
.
TMENU Tm_cc  50,198       70,209       'Empty_tablet_slot'
.
.
```

ME10 kann mit unterschiedlichen Tabletts betrieben werden. Um nicht für jedes Tablett eine eigene Talettmenüdefinition schreiben zu müssen, erfolgt eine Umrechnung der Feldkoordinaten mit Hilfe der Makros Tm_fc und Tm_cc. Beide Makros sind in Datei "tmenu_mac" enthalten. Tm_fc stellt den Typ des angeschlossenen Tabletts fest und berechnet die Umrechnungsfaktoren. Tm_cc verwendet die dem Aufruf folgenden Punktkoordinaten als Parameter und berechnet mit Hilfe der von Tm_fc gelieferten Umrechnungsfaktoren die für das jeweilige Tablett zutreffenden Feldkoordinaten.

Die Namen der den Feldern zugeordneten Makros orientieren sich an der englischen Bezeichnung der damit aufzurufenden Befehle oder Interruptfunktionen. Nachfolgend wird ein Auszug aus Datei TMENU_MAC wiedergegeben, in der die Makros definiert sind.

```
DEFINE Tm_linetype_phantom
  PHANTOM
END_DEFINE

DEFINE Empty_tablet_slot
  BEEP
END_DEFINE
```

"hp_tmenu" und "tmenu_mac" können als Vorlagen für eigene Menüdefinitionen dienen. Es empfiehlt sich jedoch, Änderungen nicht an den Originaldateien, sondern an Kopien vorzunehmen. Nach der Anpassung des Tablettmenüs sollte auch die Tablettauflage verändert werden. Zeichnungen der Standardauflagen werden mit ME10 ausgeliefert. Ihre Namen bestehen aus den Zeichen "OVL" (Overlay) und einer nachfolgenden Nummer, die den Typ des zugehörigen Tabletts kennzeichnet. Sie befinden sich normalerweise im ME10-Installationsverzeichnis.

Verwendung mehrerer Tablettmenüs

Wie bereits oben erwähnt, bietet ME10 die Möglichkeit, mehrere Tablettmenüs zu definieren und nebeneinander zu verwenden. Ein Tablettmenü ist dabei aktuell, die übrigen sind nicht aktuell. Wenn mehrere Menüdefinitionen nebeneinander existieren sollen, müssen sie sich durch ihre Namen unterscheiden. Mit diesen Namen kann dann gezielt eine Menüdefinition zur aktuellen Menüdefinition gemacht werden. Die Interruptfunktion CURRENT_TMENU dient dazu, Namen für Menüdefinitionen zu vergeben oder eine bereits existierende Menüdefinition zur aktuellen Definition zu machen.

CURRENT_TMENU (Interruptfunktion)

```
-->(CURRENT_TMENU)-->|Tablettmenüname|-->
```

Existiert noch keine Menüdefinition mit dem angegebenen Namen, erzeugen alle nachfolgend eingegebenen TMENU-Anweisungen eine Menüdefinition, die danach unter dem angegebenen Namen angesprochen werden kann. Existiert bereits eine Menüdefinition mit dem angegebenen Namen, wird sie zur aktuellen Definition und kann dann verwendet oder geändert werden. Alle bis zur nächsten CURRENT_TMENU-Anweisung eingegebenen TMENU-Anweisungen beziehen sich dann auf diese Menüdefinition.

ME10 wird standardmäßig nur mit einer einzigen Tablettmenü-Definition ausgeliefert. Sie ist in Datei HP_TMENU enthalten. Beim Systemstart werden unter anderem folgende Anweisungen ausgeführt:

```
CURRENT_TMENU ''
INPUT 'hp_tmenu'
```

Mit der ersten Anweisung wird ein leerer String als Name für die aktuelle Menüdefinition festgelegt. Die in Datei "hp_tmenu" enthaltenen Anweisungen werden beim Laden ausgeführt und definieren somit das aktuelle Tablettmenü. Will der Anwender eine weitere Menüdefinition neben der Standarddefinition verwenden, muß er zunächst einen Menünamen vergeben und danach die Anweisungen zur Menüdefinition eingeben.

Beispiel

```
CURRENT_TMENU 'Zweitmenu'
{Tablettmenu Nr. 2 mit grossem Fadenkreuz-Fuehrungsbereich}
Tm_fc
TMENU TRACKING Tm_cc 30,89   240,240
TMENU Tm_cc  30,260       50,270       'TM_LINETYPE_SOLID'
TMENU Tm_cc  30,250       50,260       'TM_LINETYPE_DASHED'
.
.
.
```

Zweckmäßigerweise werden auch diese Anweisungen in eine Datei geschrieben. Durch einen entsprechenden Eintrag in die Datei "customize" läßt sich eine automatische Ausführung beim Systemstart erreichen. Der Anwender kann mit den Anweisungen CURRENT_TMENU " und CURRENT_TMENU 'Zweitmenu' zwischen beiden Menüdefinitionen wählen.

14.3.2 Anpassung des Bildschirmmenüs

ME10 wird mit zwei unterschiedlichen Arten von Bildschirmmenüs ausgeliefert. Wenn ME10 mit einem Tablett bedient wird, werden Bildschirmmenüs über entsprechend belegte Felder des Tablettmenüs aufgerufen. Die linke Hälfte des mit ERSTELLEN beschrifteten Tablettfeldes ist z.B. mit Tm_create_1 belegt. Tm_create_1 ist ein Makro und ruft seinerseits das Makro Sm_create_1 auf. Sm_create_1 schließlich enthält die Definition des Bildschirmmenüs ERSTELLEN 1.

Wenn ME10 mit der Maus bedient wird, bestehen die Bildschirmmenüs aus einem Hauptmenübereich und einem Nebenmenübereich. Die Hauptmenüs entsprechen den Bildschirmmenüs der Tablettversion, die Nebenmenüs dem Tablettmenü. Haupt- und Nebenmenüs werden mit den gleichen ME10-Funktionen eingerichtet.

Beim Tablettmenü wird die Einteilung der Tablettfläche in Felder und die Belegung der Felder mit einer einzigen Anweisung vorgenommen (-> Kap.14.3.1). Bei den Bildschirmmenüs sind hierzu zwei Anweisungen vorgesehen. MENU_LAYOUT erzeugt lediglich eine Feldeinteilung (ein sogenanntes Layout). Entsprechende Anweisungen sind in den Dateien "hp_menu_t", "hp_lay.mac" und "hp_menu_s" abgelegt. Die Belegung der Felder erfolgt mit Hilfe der Interruptfunktion MENU. Zugehörige Anweisungen befinden sich in den Dateien "hp_macro_t" und "hp_menu_s".

ME10 bietet die Möglichkeit, mehrere Bildschirmmenüs zu definieren und nebeneinander zu verwenden. Ein Bildschirmmenü ist dabei aktuell, die übrigen sind nicht aktuell. Die nachfolgend beschriebenen Anweisungen zur Menüdefinition beziehen sich stets auf das aktuelle Menü. Am Ende dieses Kapitels wird beschrieben, wie mehrere Bildschirmmenüs nebeneinander eingerichtet und verwendet werden können.

Das aktuelle Bildschirmmenü kann durch Ändern oder durch Neudefinition den Wünschen des Benutzers angepaßt werden. Wie beim Tablettmenü kann das aktuelle Bildschirmmenü vollständig gelöscht werden. Hierzu dient die Interruptfunktion DELETE_MENU.

DELETE_MENU (Interruptfunktion)

```
-->(DELETE_MENU)-->
```

Das aktuelle Bildschirmmenü verschwindet dadurch vollständig. Wenn mehrere Bildschirmmenüs existieren und ein nicht aktuelles Menü von dem gelöschten Menü verdeckt wurde, erscheint das zuvor verdeckte Menü. Häufig soll jedoch das aktuelle Bildschirmmenü nicht vollständig gelöscht werden. Will man die Menüfeldeinteilung des Standardmenüs beibehalten und nur die Belegung der Menüfelder löschen, können die Felder mit leeren Strings belegt werden. Dies geschieht zweckmäßigerweise durch ein Makro.

Das aktuelle Menü läßt sich mit Hilfe der Interruptfunktion SAVE_MENU in einer mit INPUT lesbaren Form in eine Datei oder ein anderes Ausgabeziel schreiben (-> Kap.2.1.3).

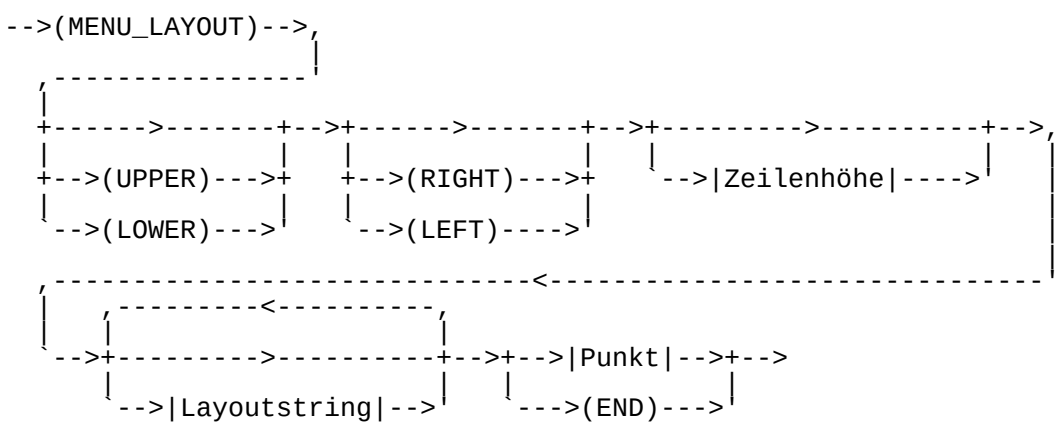
SAVE_MENU (Interruptfunktion)

```
-->(SAVE_MENU)-->|Ausgabeziel|-->
```

Festlegung des Menü-Layouts

Unter Menü-Layout werden Position, Größe und Aufteilung des Bildschirmmenüs verstanden. Das Layout des aktuellen Bildschirmmenüs wird mit Hilfe der Interruptfunktion MENU_LAYOUT bestimmt.

MENU_LAYOUT (Interruptfunktion)



Die Optionen UPPER, LOWER, RIGHT und LEFT bestimmen, an welcher Stelle des Grafikbildschirms (oben, unten, rechts, links) das Menü positioniert wird. Die Position kann auch in Bildpunkten angegeben oder durch Digitalisierung bestimmt werden (Parameter |Punkt|).

|Zeilenhöhe| bestimmt die Höhe der nächsten Menüzeile in Bildpunkten (0 bis 255). Der Standardwert hierfür hängt von der Größe des verwendeten Standardschriftfonts ab und ist normalerweise in der globalen Variablen "Text_slot_height" gespeichert. (Der Standardschriftfont findet für Menüs, Anzeigetabellen, den Dialogbereich am unteren Bildschirmrand und für Schrift auf dem Textbildschirm Verwendung. Bei der HP-UX-Version von ME10 wird er der Umgebungsvariablen MEFONT zugewiesen (z.B. MEFONT=hp8.10x20b). Die Zuweisung findet in dem Shellskript statt, mit dem auch ME10 gestartet wird.)

Menübreite und Anordnung der Menüfelder werden über Zeichenfolgen (|Layoutstring|) festgelegt. Pro Menüzeile muß je eine Zeichenfolge eingegeben werden. Innerhalb einer Zeichenfolge werden die einzelnen Menüfelder durch einen senkrechten Strich (|) getrennt. Die Anzahl der Zeichen zwischen den Strichen bestimmt die Breite des entsprechenden Feldes. Es können dabei beliebige Zeichen verwendet werden. Meist werden Leerzeichen oder Zahlen eingegeben. Wird nur die Menüposition, aber kein String eingegeben, erfolgt lediglich eine Verschiebung des aktuellen Menüs. Ohne Angabe einer Position wird das Menü in der rechten unteren Ecke des Grafikbildschirms (LOWER RIGHT) angeordnet.

MENU_LAYOUT versieht die Menüfelder mit folgenden Standardfarben:

	MEPELOOK = 0	MEPELOOK = 1
Anzeigefarbe	Weiß	Grau
Hintergrundfarbe	Schwarz	Weiß

Beispiel

```
MENU_LAYOUT UPPER LEFT
   '123456|89|12|45|78|01'
   '      |  |  |  |  |  '
40 '      |               '
   '      |               '
END
```

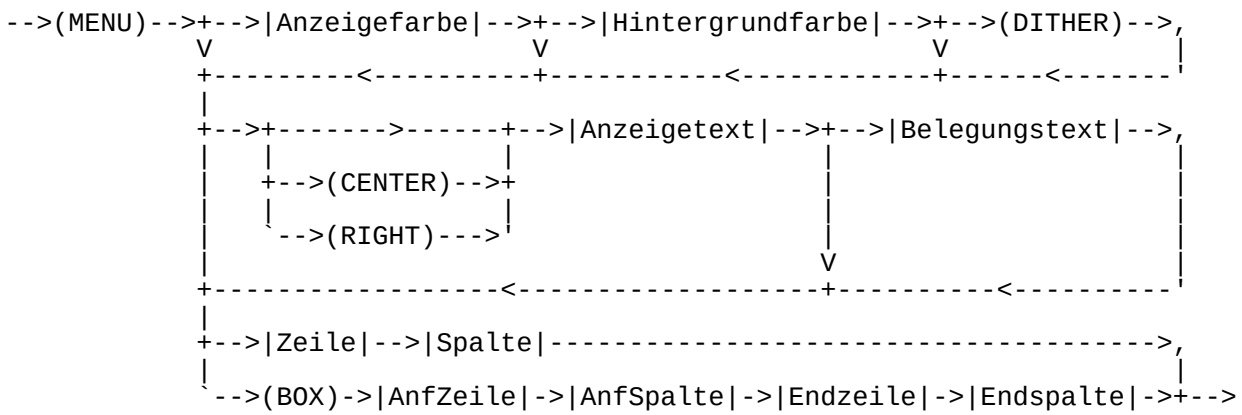
Mit dieser Anweisung wird ein Menü in der oberen linken Ecke des Bildschirms eingerichtet. Das Menü besteht aus vier Zeilen zu je 21 Zeichen (die Trennstriche sind mitzuzählen). Die zwischen die Trennungsstriche geschriebenen Zahlen erleichtern ein Abzählen der Feldbreite, sie erscheinen nachher nicht im Menü. Die ersten beiden Zeilen sind in sechs, die nächsten beiden Zeilen in zwei Felder unterteilt. Das erste Feld in jeder Zeile kann maximal 6 Zeichen aufnehmen. Die Höhen der ersten beiden und der vierten Zeile sind gleich dem Standardwert, die dritten Menüzeile ist 40 Pixel hoch.

Die Gesamthöhe eines Menüs darf einen Mindestwert nicht unterschreiten. Die Mindesthöhe ist gleich der Standard-Menüzeilenhöhe plus 2 Pixel (also "Text_slot_height + 2"). Menüs mit einer geringeren Höhe werden nicht angezeigt.

Menüfeldbelegung

Die mit MENU_LAYOUT eingerichteten Felder des aktuellen Bildschirmmenüs können mit Hilfe der Interruptfunktion MENU beschriftet und mit einem beliebigen Text belegt werden. Bei der Beschriftung kann es sich sowohl um Text als auch um Grafik (sogenannte Piktogramme) handeln. Beim Belegungstext handelt es sich meist um Aufrufkommandos für Interrupt-Funktionen, Befehle oder Makros. Im Prinzip ist jedoch eine Belegung mit beliebigen Daten möglich. Beim Auslösen des Fadenkreuzes in einem Menüfeld wird der Belegungstext ausgewertet (es erfolgt im Prinzip eine Anwendung der Stringfunktion VAL auf den Belegungstext) und an das System übergeben.

MENU (Interruptfunktion)



Die Optionen und Parameter haben folgende Bedeutung:

Optionen und *Parameter*	Bedeutung
Anzeigefarbe	Farbe des im Menüfeld angezeigten Textes oder Piktogramms
Hintergrundfarbe	Hintergrundfarbe des Menüfeldes
DITHER	Erzeugung von Mischfarben durch "DITHER-Technik" *) (nur bei Grafikkarten mit geringer Farbtiefe wirksam)
CENTER/RIGHT	Zentrische oder rechtsbündige Ausrichtung des Anzeigetexts
Anzeigetext	Der im Menüfeld angezeigte Text bzw. die Piktogramm-Zeichenfolge
Belegungstext	Zeichenfolge, die beim Antippen des Menüfeldes ausgewertet und danach an das System übergeben wird
Zeile	Zeilennummer des Menüfeldes
Spalte	Spaltennummer des Menüfeldes
BOX	Ermöglicht die Belegung mehrerer Menüfelder
AnfZeile	Zeilennummer des ersten angewählten Menüfeldes
AnfSpalte	Spaltennummer des ersten angewählten Menüfeldes
Endzeile	Zeilennummer des letzten angewählten Menüfeldes
Endspalte	Spaltennummer des letzten angewählten Menüfeldes

*) Die Anzahl der gleichzeitig darstellbaren Farben wird durch die Größe des Videospeichers und die Anzahl der dargestellten Bildpunkte bestimmt. Bei der DITHER-Technik werden Gruppen von Bildpunkten in den verfügbaren Farben so angesteuert, daß für das Auge der Eindruck einer Mischfarbe entsteht. Auf diese Weise läßt sich die Anzahl der darstellbaren Farben erhöhen.

Anzeige- und Hintergrundfarbe können mit den Farbbezeichnungen BLACK, BLUE usw., als RGB-Farben oder als HSL-Farben angegeben werden (siehe auch Help-Datei, Schlüsselwort "RGB_COLOR"). Beim den zum Lieferumfang von ME10 gehörigen Bildschirmmenüs werden Vordergrundfarben durch die Makros "Colo0" bis "Colo7" und Hintergrundfarben durch die Makros "Bcol0 bis Bcol7" festgelegt. Werden keine Anzeige- und Hintergrundfarben angegeben, so erschienen die Menüs in den Standardfarben der Funktion MENU_LAYOUT.

Mit Hilfe der Option BOX können mehrere Menüfelder gleichzeitig belegt werden. Anfangszeile, Anfangsspalte, Endzeile und Endspalte bestimmen hier den Bereich. Wenn die Menüfeldbelegung interaktiv erfolgt, können statt der Angabe von Zeilen- und Spaltennummern auch die entsprechenden Felder angetippt werden.

Beispiel: Das in Zeile 4 und Spalte 5 des aktuellen Bildschirmmenüs vorhandene Feld soll mit dem Anzeigetext 'Neuaufbau' und mit dem Aufrufkommando 'REDRAW' (Neuaufbau des aktuellen Fensters) belegt werden. Dabei sollen der Text schwarz und die Hintergrundfarbe des Feldes rot sein. Die Anweisung müßte lauten:

```
MENU BLACK RED 'Neuaufbau' 'REDRAW' 4 5
```

Eine Belegung von Menüfeldern mit konstantem Belegungstext ist im allgemeinen sehr einfach. Etwas schwieriger ist dagegen eine Belegung mit dem Inhalt von Variablen. Das nächste Beispiel soll dies demonstrieren.

Im nachfolgenden Makro "Belegen" wird ein einspaltiges Bildschirmmenü eingerichtet. Die Felder werden mit dem Text "Zeile" und der Zeilennummer beschriftet und so belegt, daß beim Antippen eines Feldes eine Ausgabe der Zeilennummer als Zahl in der Hinweiszeile erfolgt.

```
DEFINE Belegen
  CURRENT_MENU 'Test_menu'
  MENU_LAYOUT UPPER LEFT
   '12345678'
   '        '
   '        '
   '        '
   '        '
  END
  LET Nr 0
  WHILE (Nr < 5)
    LET Nr (Nr + 1)
    MENU ('Zeile ' + STR Nr) ('DISPLAY ' + STR Nr) Nr 1
  END_WHILE
END_DEFINE
```

Die Zeilennummer hat den Datentyp NUMBER (Zahl). Sie wird in der WHILE-Schleife errechnet und der Variablen "Nr" zugewiesen. Der Anzeigetext setzt sich aus der Stringkonstanten 'Zeile ' und der in einen Text umgewandelten Zeilennummer zusammen. Der Belegungstext setzt sich aus der Stringkonstanten 'DISPLAY ' und der in einen Text umgewandelten Zeilennummer zusammen. Die zweite Zeile ist zum Beispiel nach der Ausführung des Makros mit dem Text 'DISPLAY 2' belegt. Beim Antippen des Feldes wird der so erzeugte Belegungstext wie folgt ausgewertet:

```
'DISPLAY 2' -----> DISPLAY 2
```

Aus dem Text ist also das Aufrufkommando einer ME10-Funktion geworden, dem ein Leerzeichen und eine Zahl folgen.

Angezeigt wird also eine Zahl. Soll die Zeilennummer nicht als Zahl, sondern als Text angezeigt werden, müßte die Anweisung zum Belegung der Felder in obenstehendem Makro wie folgt lauten:

```
MENU ('Zeile ' + STR Nr) ('DISPLAY ' + '"' + STR Nr + '"') Nr 1
```

Die zweite Zeile wäre dann zum Beispiel nach der Ausführung des Makros mit dem Text 'DISPLAY "2"' belegt. Beim Antippen des Feldes wird der so erzeugte Belegungstext wie folgt ausgewertet:

```
'DISPLAY "2"' -----> DISPLAY "2"
```

Ein völlig anderes Ergebnis liegt vor, wenn die Anweisung zur Belegung im Makro wie folgt lautet:

```
MENU ('Zeile ' + STR Nr) 'DISPLAY Nr' Nr 1
```

Hier ist jedes Feld nach der Ausführung des Makros mit dem Text 'DISPLAY Nr' belegt. Beim Antippen des Feldes erfolgt dann folgende Auswertung:

```
'DISPLAY Nr' -----> DISPLAY Nr
```

Angezeigt wird also nicht die Nummer der jeweiligen Zeile, sondern der aktuelle Inhalt der Variablen "Nr", der nach der Abarbeitung des Makros "5" ist! Im Beispiel wurde "Nr" bewußt als globale Variable verwendet. Wird Nr als lokale Variable vereinbart, so führt das Antippen eines Feldes zur Fehlermeldung "Das Makro Nr ist nicht definiert", da "Nr" nur im Makro "Belegen" bekannt ist.

Die Ausführungen haben noch einmal deutlich gezeigt, daß Belegungstext eines Feldes und Systemeingabe beim Antippen des Feldes nicht identisch sind. Die Systemeingabe entsteht vielmehr durch Auswertung des Belegungstextes. Das letzte Beispiel zeigt außerdem, daß es ein großer Unterschied ist, ob ein Feld mit dem Namen einer Variablen oder mit ihrem Inhalt belegt wird.

Verwendung mehrere Bildschirmmenüs nebeneinander

Wie bereits erwähnt, bietet ME10 die Möglichkeit, mehrere Bildschirmmenüs zu definieren und nebeneinander zu verwenden. Ein Bildschirmmenü ist dabei aktuell, die übrigen sind nicht aktuell. Wenn mehrere Menüdefinitionen nebeneinander existieren sollen, müssen sie sich durch einen Namen unterscheiden. Mit Hilfe dieses Namens kann gezielt eine Menüdefinition zur aktuellen Menüdefinition gemacht werden. Die Interruptfunktion CURRENT_MENU dient dazu, Menüdefinitionen mit einem Namen zu kennzeichnen und eine bereits gekennzeichnete Menüdefinition zur aktuellen Definition zu machen.

CURRENT_MENU (Interruptfunktion)

```
-->(CURRENT_MENU)-->|Name|-->
```

Existiert noch keine Menüdefinition mit dem angegebenen Namen, erzeugen alle nachfolgend eingegebenen MENU_LAYOUT- und MENU-Anweisungen eine Menüdefinition, die danach unter dem angegebenen Namen angesprochen werden kann. Existiert bereits eine Menüdefinition mit dem angegebenen Namen, so wird sie zur aktuellen Definition und kann dann verwendet oder geändert werden. Alle bis zur nächsten CURRENT_MENU-Anweisung eingegebenen MENU- und MENU_LAYOUT-Anweisungen beziehen sich dann auf diese Menüdefinition.

Beispiel

Mit den nachfolgenden Anweisungen wird ein zeilenförmiges Bildschirmmenü am unteren linken Rand des Grafikbildschirms eingerichtet und mit Fangfunktionen belegt. Bei der Definition des Menü-Layouts wird die Stringfunktion RPT verwendet (-> Kap.8.4).

```
DEFINE Fangen
  CURRENT_MENU 'Fangmenu'
  {Bildschirmmenue mit Fangfunktionen}
  MENU_LAYOUT
  (RPT '        |' 8 + '        ')
  LOWER LEFT
  END
  MENU WHITE BLACK 'FANGEN' '' 1 1
  MENU BLACK CYAN 'Alles' 'CATCH ALL' 1 2
  MENU BLACK CYAN 'Scheitel' 'CATCH VERTEX' 1 3
  MENU BLACK CYAN 'Element' 'CATCH ELEM' 1 4
  MENU BLACK CYAN 'Schnittp' 'CATCH INTERSECTION' 1 5
  MENU BLACK CYAN 'Mitte' 'CATCH CENTER' 1 6
  MENU BLACK CYAN 'Gitter' 'CATCH GRID' 1 7
  MENU BLACK CYAN 'Aus' 'CATCH OFF' 1 8
  MENU BLACK CYAN 'Bereich' 'CATCH RANGE' 1 9
END_DEFINE
```

Zweckmäßigerweise werden auch diese Anweisungen in eine Datei geschrieben. Sie können beim Systemstart durch einen entsprechenden Eintrag in Datei 'customize' automatisch ausgeführt werden.

Einfügen von Piktogrammen in Bildschirmmenüs

Menüfelder können auch mit Grafiken (sogenannten Piktogrammen) beschriftet werden. Die Piktogramme müssen aus Geradenstücken aufgebaut werden. Bogenförmige Teile sind durch Geraden anzunähern. Eine Folge von Geradenstücken entsteht, wenn in der MENU-Anweisung folgender Anzeigetext verwendet wird:

```
(CHR 255 + CHR x_start + CHR y_start + CHR x_folg1
 + CHR y_folg1 + CHR x_folg2 + CHR y_folg2 + .....)
```

CHR 255 bewirkt, daß das nächste Wertepaar (CHR x_start + CHR y_start) als Anfangspunkt eines Geradenstücks interpretiert wird. x_start und y_start stellen Pixelkoordinaten dar und beziehen sich auf die linke untere Ecke des mit dem Piktogramm zu belegenden Menüfelds. Die nachfolgenden Wertepaare (x_folg1 + y_folg1 usw.) werden als Folgepunkte des Geradenzugs interpretiert und haben den gleichen Bezugspunkt. Wenn sich das Piktogramm nicht aus einem geschlossenen Geradenzug aufbauen läßt, können im Anzeigetext weitere Anfangspunkte und Folgepunkte in der oben beschriebenen Weise definiert werden.

Beispiel

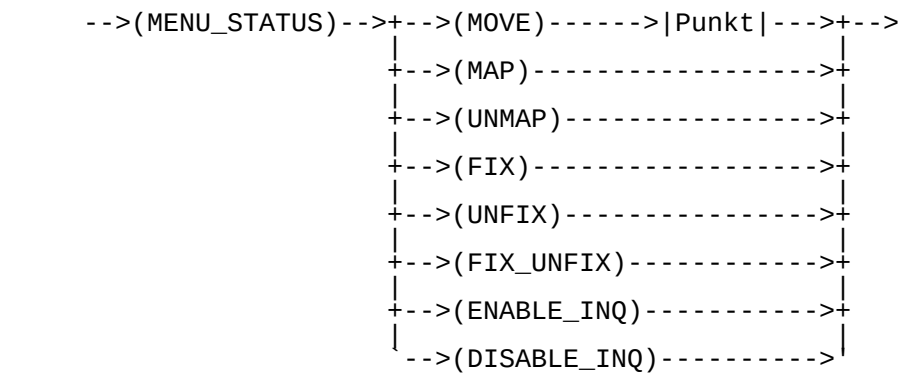
Ein in Zeile 1 und Spalte 1 des aktuellen Bildschirmmenüs befindliches Menüfeld soll mit einem Rechtecksymbol beschriftet und mit dem Befehl LINE RECTANGLE belegt werden. Das Menüfeld soll zuvor in einer Breite von 60 Pixel und eine Höhe von 40 Pixel eingerichtet worden sein. Zwischen Symbol und Menüfeldbegrenzung ist ein Abstand von 5 Pixel einzuhalten.

```
MENU BLACK WHITE          {Hintergrundfarbe schwarz, Beschriftung Weiss}
(CHR 255 + CHR 5 + CHR 5  {Anfangspunkt = Ecke links unten}
+ CHR 55 + CHR 5          {Ecke rechts unten}
+ CHR 55 + CHR 35         {Ecke rechts oben}
+ CHR 5 + CHR 35          {Ecke links oben}
+ CHR 5 + CHR 5)          {Ecke links unten, = Endpunkt}
'LINE RECTANGLE' 1 1      {Belegung mit ME10-Befehl}
```

Festlegen und Ändern des Menüstatus

Das aktuelle Bildschirmmenü kann mit Hilfe der Interruptfunktion MENU_STATUS bewegt, angezeigt, ausgeblendet, fixiert/nichtfixiert und für Abfragen zugelassen oder gesperrt werden.

MENU_STATUS (Interruptfunktion)



Die Optionen und Parameter haben folgende Bedeutung:

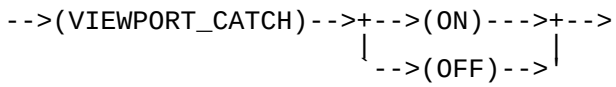
Optionen und *Parameter*	Bedeutung
MOVE *Punkt*	Das Menü wird an den angegebenen Punkt verschoben
MAP	Das Menü wird vollständig abgebildet
UNMAP	Das Menü wird ausgeblendet
FIX	Das Menü wird an der momentanen Position fixiert
UNFIX	Das Menü wird an seiner Stammposition fixiert
FIX_UNFIX	Schaltet zwischen FIX und UNFIX um
ENABLE_INQ	Menüabfrage wird ermöglicht (-> Kap.13)
DISABLE_INQ	Menüabfrage wird gesperrt (Standardeinstellung)

Mit Hilfe von Feldern der ersten Menüzeile können die Standard-Bildschirmmenüs fixiert/nichtfixiert, bewegt und ausgeblendet werden. Dabei findet ein Teil der vorgenannten Funktionen Verwendung. Die Belegung der ersten Menüzeile erfolgt durch das Systemmakro "Menu_control_icons". Um eigene Bildschirmmenüs mit dieser Standard-Kontrollzeile zu versehen, genügt es, das Layout der ersten Zeile der Systemmenüs zu übernehmen, Feld Nr. 3 mit dem Menütitel als Anzeigetext und die übrigen Felder mit Hilfe des Systemmakros "Menu_control_icons" zu belegen (siehe dazu auch das Beispielmakro in Kapitel 14.3.3).

Auf eine Gefahr muß dabei allerdings hingewiesen werden. Bei "Menu_control_icons" handelt es sich um ein nicht dokumentiertes Makro. Der Systemhersteller behält sich damit das Recht vor, diese Funktion in späteren Softwareversionen anders zu definieren oder auch nicht mehr zu verwenden. Wer ganz sicher gehen will, daß seine Makros auch unter späteren ME10-Versionen noch laufen, sollte grundsätzlich auf nicht dokumentierten Funktionen verzichten. Im vorliegenden Fall ist dies kein großes Problem. Das Systemmakro "Menu_conrol_icons" und alle von ihm verwendeten Untermakros und Piktogramme können mit Hilfe von SAVE_MACRO in einer ASCII-Datei gespeichert werden. Sie stehen damit auch für spätere ME10-Versionen zur Verfügung.

Wenn Bildschirmmenüs bewegt werden, ist es manchmal wünschenswert, daß sie exakt in die Position anderer Menüs, Fenster oder Anzeigetabelle plaziert werden können. Dies läßt sich mit Hilfe der Funktion VIEWPORT_CATCH erreichen.

VIEWPORT_CATCH (Interruptfunktion)



Bei der Standardeinstellung ON rasten neu erstellte, vergrößerte, verkleinerte oder verschobene Fenster, Bildschirmmenüs oder Anzeigetabellel in die Position anderer Fenster, Bildschirmmenüs oder Anzeigetabellen ein. Der Fangbereich beträgt 8 Bildpunkte. Mit VIEWPORT_CATCH OFF wird das Einrasten ausgeschaltet.

14.3.3 Benutzerdialog über das Menüsystem

Benutzerdialog findet bei ME10 über die Hinweiszeile statt. Die Anweisung "READ NUMBER 'Bitte Wert eingeben' Sigma" bewirkt zum Beispiel, daß der Benutzer zur Eingabe eines Wertes aufgefordert wird. Eine danach vom Benutzer über die Tastatur eingegebene Zahl erscheint in der Eingabezeile und wird der Variablen "Sigma" zugewiesen. Die Eingabe über die Tastatur läßt sich auf sehr einfache Weise durch eine Eingabe über das Menüsystem ersetzen.

Felder des Tablett- und des Bildschirmmenüs können nämlich nicht nur mit Aufrufkommandos, sondern mit beliebigen Zeichenfolgen belegt werden. Beim Antippen eines Feldes wird die Zeichenfolge an das System übergeben. Wenn der Benutzer zum Beispiel nach der obengenannten Eingabeanforderung ein mit einem Zahlenwert belegtes Menüfeld antippt, wird der Zahlenwert übernommen und der Variablen "Sigma" zugewiesen. Insbesondere mit Bildschirmmenüs lassen sich komfortable Benutzeroberflächen für Makros schaffen. Das folgende Beispiel soll die Vorgehensweise erläutern. Es handelt sich dabei um ein Makro zum Zeichnen von Kegelstiften nach DIN 1. Die untenstehende Tabelle gibt eines Auszug aus DIN 1 wieder.

Auszug aus DIN 1 Kegelstifte (September 1981)

Kegelstifte nach DIN 1 (Bohrungen mit Kegelreibahlen DIN 9 gerieben)

d	l von	l bis	r
2	12	36	2,5
3	14	50	4
4	16	60	4
5	20	70	6
6	24	100	6
8	28	120	10
10	32	140	10
12	36	165	12
14	36	165	16
16	40	230	16
20	50	230	20

Stufung der Länge				
12	14	16	18	20
22	24	26	28	30
32	36	40	45	50
55	60	70	80	90
100	110	120	130	140
150	165	180	200	230

Beim Aufruf des Makros soll ein Bildschirmmenü erscheinen, das den Benutzer über die zur Verfügung stehenden Durchmesser und Längen informiert und eine Auswahl dieser Größen durch Antippen der Menüfelder ermöglicht. Dabei ist eine Auswahl nicht genormter Abmessungen zu verhindern. Gelöst wird die Aufgabe mit den nachfolgend dargestellten Makros "Stiftmenu_layout", "Stiftmenu" und "Kegelstift". "Stiftmenu_layout" und "Stiftmenu" definieren lediglich das Auswahlmenü. Die Steuerung des Programmablaufs und das Zeichnen des Stiftes erfolgt mit Makro "Kegelstift". Die Makros sind ausführlich mit Kommentar versehen, so daß sich eine weitere Beschreibung erübrigt.

```
DEFINE Stiftmenu_layout
 {Bildschirmmenülayout für Kegelstifte DIN 1}
 {Fischer, 12.09.96, Stand: 26.10.96}
 CURRENT_MENU 'Kegelstift_ly'
 MENU_LAYOUT
  Menu_position RIGHT
  Headline_height    ' | |               | '
  Text_slot_height   '           |        '
  Text_slot_height   '           |        '
  Text_slot_height   '423456|89012345678901'
  Text_slot_height   '05    |            '
  Text_slot_height   '06    |            '
  Text_slot_height   '07    |            '
  Text_slot_height   '08    |            '
  Text_slot_height   '09    |            '
  Text_slot_height   '10    |            '
  Text_slot_height   '11    |            '
  Text_slot_height   '12    |            '
  Text_slot_height   '13    |            '
  Text_slot_height   '14    |            '
  Text_slot_height   '15    |            '
  Text_slot_height   '16                 '
  Text_slot_height   '17  |   |   |   |  '
  Text_slot_height   '18  |   |   |   |  '
  Text_slot_height   '19  |   |   |   |  '
  Text_slot_height   '20  |   |   |   |  '
  Text_slot_height   '21  |   |   |   |  '
  Text_slot_height   '22  |   |   |   |  '
  Text_slot_height   '23      |   |   |  '
  Text_slot_height   '24                 '
  Text_slot_height   '25          |      '
  Text_slot_height   '26  |       |   |  '
  Text_slot_height   '27  |       |   |  '
  Menu_home_point_top
 END
 CURRENT_MENU Lastmen
END_DEFINE



DEFINE Stiftmenu
 {Bildschirmmenüdefinition für Kegelstifte DIN 1}
 {Fischer, 12.09.96, Stand: 20.10.96}

 {Prüfen, ob Menü darstellbar ist}
 IF (I_port) Check_i_port END_IF
 IF (NOT I_port)
  MENU_BUFFER ON
  CURRENT_MENU 'Kegelstift_ly'
  Menu_control_icons {Standardbelegung für erste Zeile}
  MENU Colo0 Bcol5 CENTER 'Stifte' '' 1 3
  MENU Colo0 Bcol1 'Kegelstift' 'Kegelstift' 3 1
  MENU Colo0 Bcol4 'Nenn-Ø' '' 4 1
  MENU Colo0 Bcol4 'Längenbereich' '' 4 2

  {Menüfelder, beschriftet mit Nenndurchmesser, belegt mit}
  {Nenndurchmesser, Kuppenradius, minimaler und maximaler Länge}
  MENU '  2' '2 2.5 12 36' 5 1
  MENU '  3' '3 4 14 50' 6 1
  MENU '  4' '4 4 16 60' 7 1
  MENU '  5' '5 6 20 70' 8 1
  MENU '  6' '6 6 24 100' 9 1
  MENU '  8' '8 10 28 120' 10 1
  MENU ' 10' '10 10 32 140' 11 1
  MENU ' 12' '12 12 36 165' 12 1
  MENU ' 14' '14 16 36 165' 13 1
```

```
  MENU ' 16' '16 16 40 230' 14 1
  MENU ' 20' '20 20 50 230' 15 1

  {Menüfelder ohne Belegung, beschriftet mit minimaler und maximaler Länge}
  MENU ' 12  -   36' 'BEEP'  5 2
  MENU ' 14  -   50' 'BEEP'  6 2
  MENU ' 16  -   60' 'BEEP'  7 2
  MENU ' 20  -   70' 'BEEP'  8 2
  MENU ' 24  -  100' 'BEEP'  9 2
  MENU ' 28  -  120' 'BEEP'  10 2
  MENU ' 32  -  140' 'BEEP' 11 2
  MENU ' 36  -  165' 'BEEP' 12 2
  MENU ' 36  -  165' 'BEEP' 13 2
  MENU ' 40  -  230' 'BEEP' 14 2
  MENU ' 50  -  230' 'BEEP' 15 2

  {Menüfelder mit genormten Längen beschriftet und belegt}
  MENU Colo0 Bcol4 CENTER 'Längenauswahl' '' 16 1

  MENU '  12' '12' 17 1
  MENU ' 14' '14' 17 2
  MENU ' 16' '16' 17 3
  MENU ' 18' '18' 17 4
  MENU ' 20' '20' 17 5

  MENU '  22' '22' 18 1
  MENU ' 24' '24' 18 2
  MENU ' 26' '26' 18 3
  MENU ' 28' '28' 18 4
  MENU ' 39' '30' 18 5

  MENU '  32' '32' 19 1
  MENU ' 36' '36' 19 2
  MENU ' 40' '40' 19 3
  MENU ' 45' '45' 19 4
  MENU ' 50' '50' 19 5

  MENU '  55' '55' 20 1
  MENU ' 60' '60' 20 2
  MENU ' 70' '70' 20 3
  MENU ' 80' '80' 20 4
  MENU ' 90' '90' 20 5

  MENU ' 100' '100' 21 1
  MENU '110' '110' 21 2
  MENU '120' '120' 21 3
  MENU '130' '130' 21 4
  MENU '140' '14' 21 5

  MENU ' 150' '150' 22 1
  MENU '165' '165' 22 2
  MENU '180' '180' 22 3
  MENU '200' '200' 22 4
  MENU '230' '230' 22 5

  {Menüfelder zur Auswahl der Ansicht}
  MENU Colo0 Bcol4 'Ansicht' '' 23 1
  MENU Colo0 Bcol0 CENTER 'S' '"S"' 23 2
  MENU Colo0 Bcol0 CENTER 'V' '"V"' 23 3
  MENU Colo0 Bcol0 CENTER 'R' '"R"' 23 4
  MENU CYAN Bcol0 CENTER 'Gewählt wurde:' '' 24 1
  MENU CYAN Bcol0 '' '' BOX 25 1 25 2
  Eight_menu_slots_add
 END_IF

END_DEFINE


{Einrichten des Layouts und Einblenden des Menüs beim Laden der Datei}
Stiftmenu_layout
Stiftmenu


DEFINE Kegelstift
{Zeichnen eines Kegelstifts nach DIN1}
{Fischer, 12.09.96, Stand 13.09.96}

 LOCAL Dn LOCAL Dm LOCAL Ls LOCAL Rk LOCAL Nochmal LOCAL H0 LOCAL H1
 LOCAL P0 LOCAL P1 LOCAL P2 LOCAL P3 LOCAL P4 LOCAL P5 LOCAL P6 LOCAL P7
 LOCAL L_farbe LOCAL L_art
```

```
LOOP {Endlosschleife}

 {Einlesen des Nenndurchmessers}
 READ NUMBER 'Nenndurchmesser?' Dn
 IF (NOT I_Port)
  CURRENT_MENU 'Kegelstift_ly'
  MENU Colo0 Bcol0 CENTER 'S' '"S"' 23 2
  MENU Colo0 Bcol0 CENTER 'V' '"V"' 23 3
  MENU Colo0 Bcol0 CENTER 'R' '"R"' 23 4
  MENU CYAN Bcol0 BOX '' '' 25 1 25 2
  MENU CYAN Bcol0 ('Ý = ' + STR Dn) '' 25 1
 END_IF
 {Bei Eingabe von Dn über Menü werden Rk, Lmin und Lmax automatisch zugewiesen}
 READ NUMBER 'Kuppenradius?' Rk
 READ NUMBER 'Min-Länge?' Lmin
 READ NUMBER 'Max-Länge?' Lmax

 {Einlesen der Länge}
 LOOP {Schleife zum Überprüfen der Eingaben}
  LET Nochmal FALSE {Bei unzulässigen Eingaben wird Nochmal auf TRUE gesetzt}
  READ NUMBER 'Länge?' Ls
  {Überprüfung der eingegebenen Länge}
  IF ((Ls < Lmin) OR (Ls > Lmax))
   LET Nochmal TRUE
   IF (NOT I_port)
    CURRENT_MENU 'Kegelstift_ly'
    MENU_RED Bcol0 ('L = ' + STR Ls) 25 2
   UPDATE_SCREEN
   END_IF
   BEEP DISPLAY 'Länge unzulässig, bitte Neueingabe'
  END_IF
 EXIT_IF (NOT Nochmal)
 END_LOOP
 IF (NOT I_Port)
  CURRENT_MENU 'Kegelstift_ly'
  MENU CYAN Bcol0 ('L = ' + STR Ls) 25 2
 END_IF

 {Auswahl der Ansicht und Eintrag in Eingabekontroll-Menüfeld}
 READ STRING 'Ansicht? ( S = Seitenansicht, V = Kegel zum Betrachter,\
 R = Kegel vom Betrachter weg)' Ans
 IF (NOT I_Port)
  CURRENT_MENU 'Kegelstift_ly'
  MENU IF (Ans = 'S') CYAN Bcol0 ELSE Colo0 Bcol0 END_IF CENTER 'S' '"S"' 23 2
  MENU IF (Ans = 'V') CYAN Bcol0 ELSE Colo0 Bcol0 END_IF CENTER 'V' '"V"' 23 3
  MENU IF (Ans = 'R') CYAN Bcol0 ELSE Colo0 Bcol0 END_IF CENTER 'R' '"R"' 23 4
 END_IF

 {Positionierung des Kegelstifts}
 READ PNT 'Kegelposition?' P0
 {Richtung durch zweiten Punkt oder durch Winkelangabe definiert}
 READ PNT NUMBER 'Richtung? (Punkt oder Winkel)' RUBBER_LINE P0 P1
 IF (TYPE P1 = NUMBER) LET P1 (P0 + (PNT_RA 1 P1)) END_IF

 {Berechnung von Einheitsvektoren längs und quer}
 LET Evl ((P1 - P0) / LEN (P1 - P0))
 LET Evq (ROT Evl 90)

 {Berechnung des maximalen Kegeldurchmessers für Kegel 1 : 50}
 LET Dm (Dn + Ls / 50)

 {Anlegen eines Teils mit Namen = DIN-Bezeichnung}
 INIT_PART ('Kegelstift DIN 1 - ' + STR Dn + ' x ' + STR Ls)

 {Retten von aktueller Farbe und Linienart}
 INQ_ENV 3
 LET L_farbe (INQ 201)
 LET L_art (INQ 301)

 {Zeichnen der Seitenansicht}
 IF (Ans = 'S')
  LET H0 (Rk - 0.5 * SQRT (4 * Rk * Rk - Dm * Dm))
  LET H1 (Rk - 0.5 * SQRT (4 * Rk * Rk - Dm * Dm))
  LET P2 (P0 + Evq * Dm / 2)
  LET P3 (p0 - Evl * H0)
  LET P4 (P0 - Evq * Dm / 2)
  LET P5 (P0 + Evl * Ls - Evq * Dn / 2)
  LET P6 (P0 + Evl * (Ls + H1))
  LET P7 (P0 + Evl * Ls + Evq * Dn / 2)
  LINE POLYGON WHITE SOLID P2 P4 P5 P7 P2
```

```
   ARC THREE_PTS P2 P4 P3
                 P5 P7 P6
   LINE YELLOW DOT_CENTER (P3 - Evl * 2) (P6 + Evl * 2)
    RGB_COLOR L_farbe LINEPATTERN L_art END
  END_IF

  {Zeichnen der Vorderansicht}
  IF (Ans = 'V')
   CIRCLE WHITE SOLID P0 (Dn / 2) END
   LINE YELLOW DOT_CENTER (P0 - Evl * (Dn / 2 + 2)) (P0 + Evl * (Dn / 2 + 2))
    (P0 - Evq * (Dn / 2 + 2)) (P0 + Evq * (Dn / 2 + 2))
    RGB_COLOR L_farbe LINEPATTERN L_art END
  END_IF

  {Zeichnen der Rückansicht}
  IF (Ans = 'R')
   CIRCLE WHITE SOLID P0 (Dm / 2)
   LINE YELLOW DOT_CENTER (P0 - Evl * (Dm / 2 + 2))  (P0 + Evl * (Dm / 2 + 2))
    (P0 - Evq * (Dm / 2 + 2))  (P0 + Evq * (Dm / 2 + 2))
    RGB_COLOR L_farbe LINEPATTERN L_art END
  END_IF
  END_PART

 END_LOOP

END_DEFINE
```

15 Tabellen

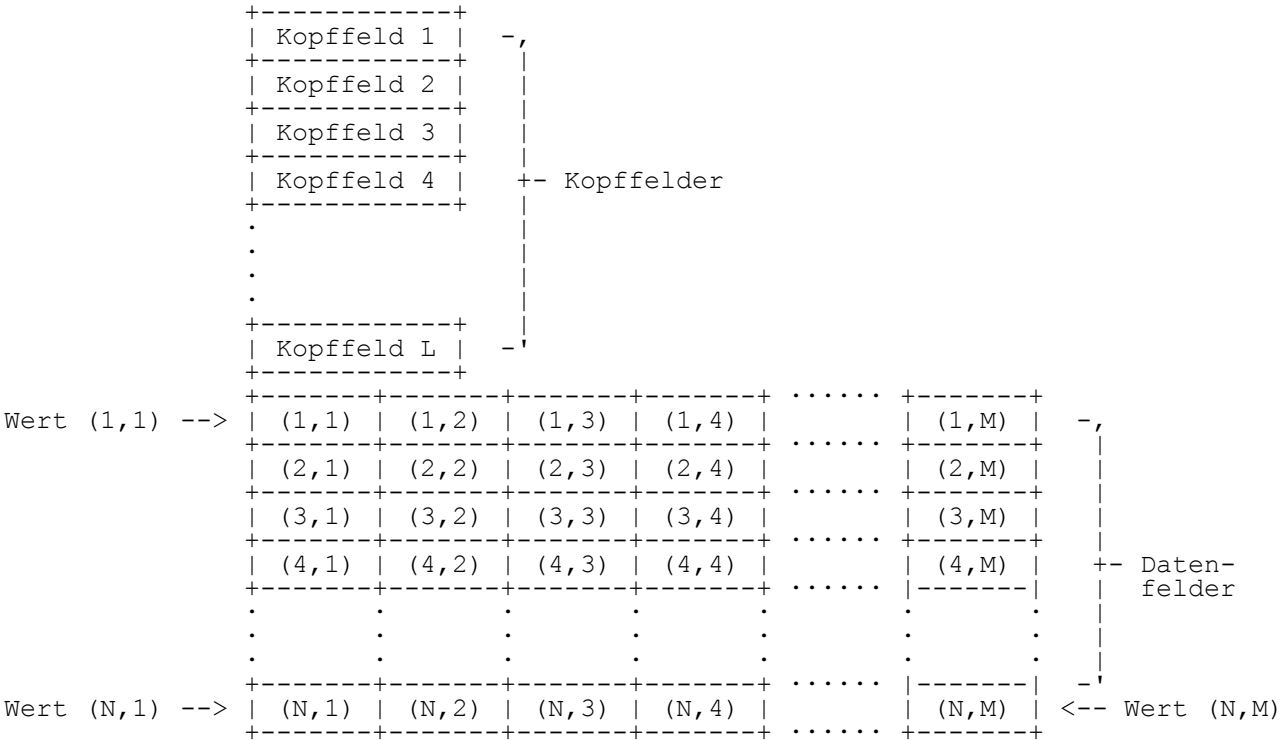
15.1 Eigenschaften

ME10 kann Daten in Tabellenform speichern, anzeigen und benutzen. Die Hilfsmittel dazu werden als Logische Tabellen und Anzeigetabellen bezeichnet. Eine Verwendung dieser Tabellen ist immer dann vorteilhaft, wenn größere, strukturierbare Datenmengen vorliegen und übersichtlich verwaltet werden sollen. Anzeigetabellen bieten überdies noch die Möglichkeit, sehr flexible Bildschirmmenüs einzurichten.

Logische Tabellen

Logische Tabellen sind interne Datenstrukturen, die Daten der Datentypen "STRING" (Text) und "NUMBER" (Zahl) aufnehmen können und einen schnellen Zugriff auf die Daten ermöglichen. Sie sind für den Benutzer nicht sichtbar, ihre Einträge können aber in Anzeigtabellen sichtbar gemacht werden. ME10 unterscheidet systemdefinierte und benutzerdefinierte Logische Tabellen. Systemdefinierte Logische Tabellen enthalten Systemdaten. Sie sind gegen Verändern gesichert, können also nur gelesen werden. Obwohl prinzipiell die Möglichkeit besteht, den Schreibschutz aufzuheben und die Daten zu verändern, sollte davon kein Gebrauch gemacht werden, da die Folgen nur für sehr erfahrene Anwender vorhersehbar sind.

Man kann sich Logische Tabellen wie folgt vorstellen:



Logische Tabellen sind in Kopffelder und Datenfelder unterteilt. Die Anzahl der Felder ist nicht begrenzt. In Kopffeldern werden meist Texte gespeichert, die dann in zugeordneten Anzeigetabellen als Überschriften Verwendung finden. Sie können aber auch Zahlenwerte aufnehmen. Kopffelder sind durch Kopffeldnummern gekennzeichnet, die mit 1 beginnen.

In Datenfelder können ebenfalls Texte oder Zahlenwerte eingetragen werden. Sie sind durch Zeilen- und Spaltennummern gekennzeichnet, die jeweils mit 1 beginnen. Das Kopffeld einer logischen Tabelle kann also ein Vektor, das Datenfeld als eine zweidimensionale Matrix angesehen werden. Logische Tabellen können auch nur aus Kopffeldern oder nur aus Datenfeldern bestehen.

Anzeigetabellen

Anzeigetabellen sind mit Bildschirmmenüs vergleichbar, besitzen aber noch zusätzliche Leistungsmerkmale. Wie bei Bildschirmmenüs lassen sich damit Daten in Tabellenform auf dem Bildschirm anzeigen. Es kann dabei eine beliebige Unterteilung in farbig angelegte Felder vorgenommen werden. Man unterscheidet hierbei einen Kopfbereich und einen Datenbereich. Die Felder dieser Bereiche lassen sich ebenfalls mit beliebigen Daten (z.B. ME10-Befehlen, Texten, Zahlenwerten) belegen, die beim Antippen an das System übergeben werden. Wesentliche Unterschiede zu Bildschirmmenüs bestehen in folgenden Punkten:

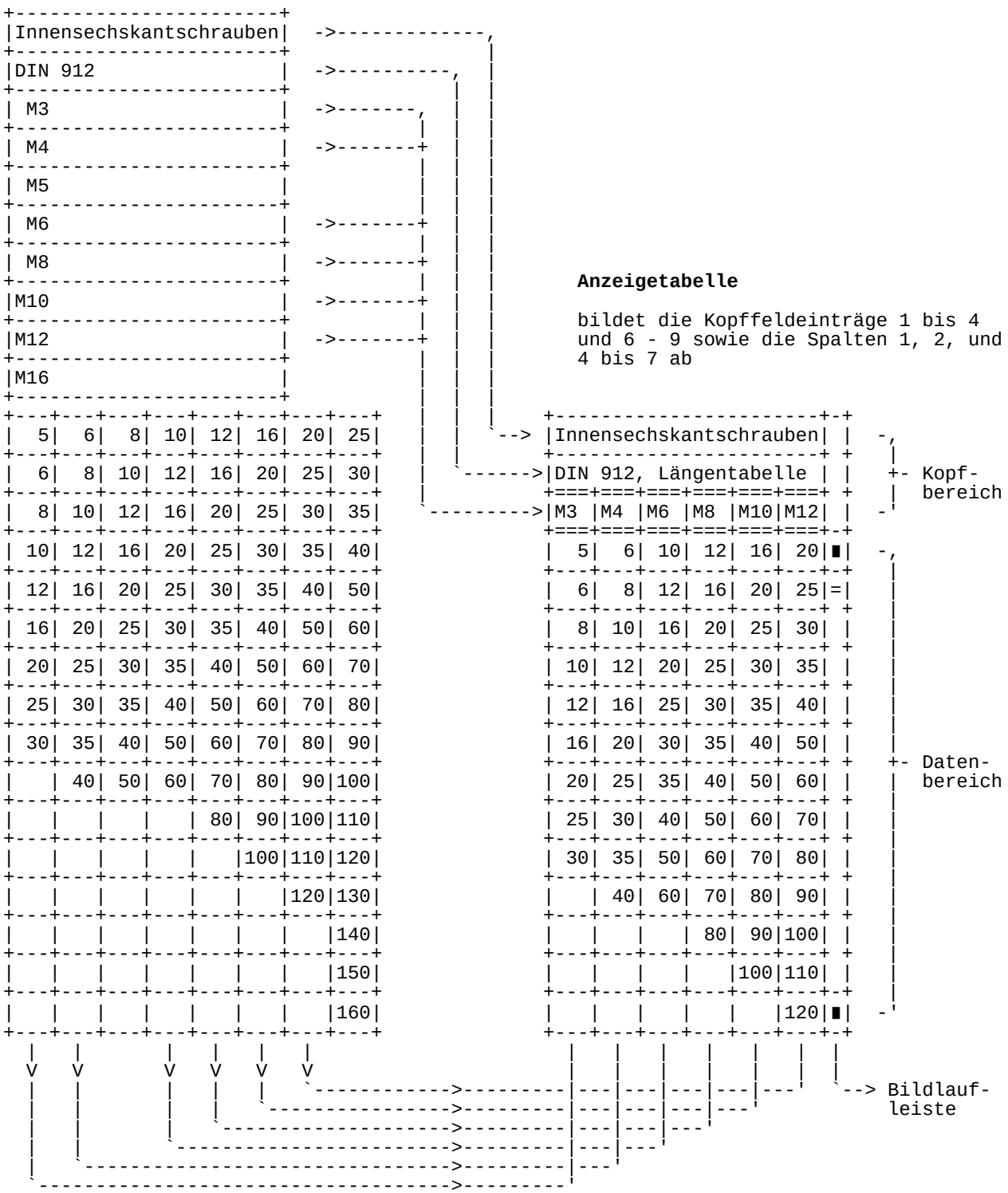
- Der Datenbereich von Anzeigetabellen kann als vertikale Rollanzeige eingerichtet werden. Dadurch lassen sich sehr große Tabellen handhaben.
- Anzeigetabellen lassen sich mit logischen Tabellen verknüpfen. Anzeige- und Belegungsdaten können dann den logischen Tabellen entnommen werden. Dies ermöglicht eine sehr flexible Verwendung der Anzeigetabellen. So werden z.B. Anzeige- und Belegungsdaten einer Anzeigetabelle bei Veränderung der logischen Tabelle automatisch aktualisiert.

Eine Anzeigetabelle kann gleichzeitig nur mit einer einzigen logischen Tabelle verbunden sein. Die Daten einer logischen Tabelle lassen sich jedoch gleichzeitig in mehreren Anzeigetabellen darstellen. Dadurch ist es möglich, verschiedene Ausschnitte einer umfangreichen logischen Tabelle separat anzuzeigen. Anzeigetabellen sind also mit Fenstern vergleichbar, die einen Blick auf Logische Tabellen ermöglichen. Meist werden Kopffeldeinträge der logischen Tabellen für Überschriften und Spalteneintrage der logischen Tabellen für Datenspalten der Anzeigetabellen verwendet.

In der nächsten Abbildung sind eine Logische Tabelle und eine Anzeigetabelle dargestellt. Die Logische Tabelle enthält die in DIN 912 genormten Längen von Innensechskantschrauben mit den Nenndurchmessern M3 bis M16. Die für M3 genormten Längen sind in Spalte 1 enthalten, die Längen für M4 in Spalte 2 usw. In den Kopffeldern sind die Schraubenbezeichnung, die DIN-Nummer sowie die Nenndurchmesser als Text gespeichert. Die rechts von der logischen Tabelle dargestellte Anzeigetabelle bildet die Spalten 1, 2, und 4 bis 7 der logischen Tabelle ab und verwendet die Texte der Kopffelder 1 bis 4 und 6 bis 9 als Überschriften. Der Text ", Längentabelle" entstammt nicht der logischen Tabelle, sondern wird als Textkonstante direkt in die Anzeigetabelle eingesetzt.

Wenn Daten nicht angezeigt, sondern nur gespeichert und bei Bedarf schnell gelesen werden sollen, können Logische Tabellen auch ohne Anzeigetabellen verwendet werden. Eine Anzeigetabelle ist hingegen in der Regel nur zusammen mit einer logischen Tabelle zu verwenden, da die im Datenbereich angezeigten Daten stets der logischen Tabelle entnommen werden müssen. Ausnahmen bilden Anzeigetabellen ohne Datenbereich, also Anzeigetabellen, die nur einen Kopfbereich besitzen. Daten des Kopfbereichs können direkt in dessen Felder eingetragen werden..

Logische Tabelle



15.2 Logische Tabellen

Alle die Logischen Tabellen betreffenden Funktionen benötigen den Tabellennamen als Parameter. Bei etlichen Funktionen müssen darüber hinaus auch noch ein Kopffeld, ein Datenfeld, eine Datenzeile oder eine Datenspalte angegeben werden. Normalerweise erfolgen solche Angaben über die Tastatur. Wenn eine Logische Tabelle mit einer sichtbaren Anzeigetabelle verbunden ist, kann in den meisten Fällen statt der Tastatureingabe ein Antippen der Tabelle, der Felder, Zeilen oder Spalten erfolgen.

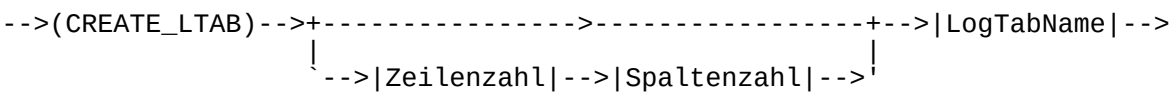
Ausnahmen bilden die integrierten Funktionen „LTAB_TITLES, LTAB_ROWS, LTAB_COLUMNS, READ_LTAB“ sowie „CREATE_LTAB“. Das Verfahren ist auch nicht möglich, wenn der angetippte Tabellenbereich mit Belegungstext versehen ist. In diesem Fall wird nämlich der Belegungstext als Eingabe für die gerade aktive Funktion verwendet, was in aller Regel zu einem Fehler führt.

Logische Tabellen können gegen Löschen und gegen Verändern gesichert werden (-> Kap.15.2.5). Bei allen Operationen, die Logische Tabellen verändern, z.B. Beschreiben, Erweitern, Löschen von Zeilen, erfolgt automatisch eine vorübergehende Sperrung gegen Löschen.

15.2.1 Anlegen

Logische Tabellen werden mit CREATE_LTAB unter einem globalen Namen angelegt. Der Namen wird beim Anlegen in die systemdefinierte Logische Tabelle "sys_user_tables" eingetragen. Die Tabelle kann danach Daten der Datentypen "STRING" oder "NUMBER" aufnehmen. Sie ist systemweit bekannt, kann also von allen Makros und auch vom Benutzer selbst verwendet werden.

CREATE_LTAB (Interruptfunktion)



|Zeilenzahl| und |Spaltenzahl| bestimmen die Anzahl der Zeilen und Spalten der logischen Tabelle. Es sind Maximalwerte von 16000 Zeilen und 1020 Spalten zulässig. Die Anzahl der Kopffelder muß nicht definiert werden. Der Name kann als Textkonstante oder Textvariable angegeben werden. Zeilen- und Spaltenzahl sollten einigermaßen richtig abgeschätzt werden. Das System vergrößert zwar bei Bedarf die Anzahl der Zeilen oder Spalten automatisch, eine geringfügige Leistungseinbuße kann jedoch die Folge sein. Wenn eine Logische Tabelle unter dem angegebenen Namen bereits existiert und nicht mit SECURE_LTAB READ_ONLY gegen Verändern gesichert wurde (-> Kap.15.2.5), werden ihre Daten ohne Warnung gelöscht. Bei gegen Verändern gesicherten Tabellen erfolgt eine Fehlermeldung.

Beispiel: Es soll eine Logische Tabelle mit 10 Zeilen und 7 Spalten unter dem Namen "Bolzendaten" angelegt werden.

```
CREATE_LTAB 10 7 'Bolzendaten'
```

15.2.2 Erweitern

In vorhandene, nicht gegen Verändern gesicherte Logische Tabellen (-> Kap.15.2.5) können an beliebigen Stellen Zeilen eingefügt werden. Bei gegen Verändern gesicherten Tabellen erfolgt eine Fehlermeldung.

INSERT_LTAB_ROW (Interruptfunktion)

```
-->(INSERT_LTAB_ROW)-->|LogTabName|-->|Zeilennummer|-->|Anzahl|-->
```

|Zeilennummer| bestimmt die Zeile, nach der die Einfügung erfolgt. Wenn die Zeilen vor der ersten Zeile eingefügt werden sollen, ist eine 0 einzugeben. |Anzahl| bestimmte die Zahl der einzufügenden Zeilen.

Beispiel: Bei der logischen Tabelle "Bolzendaten" sollen nach Zeile 2 5 Zeilen eingefügt werden.

```
INSERT_LTAB_ROW 'Bolzendaten' 2 5
```

15.2.3 Vollständiges Löschen

Logische Tabellen können gelöscht werden, wenn sie nicht gesichert oder vorübergehend gesperrt sind.

DELETE_LTAB (Interruptfunktion)

```
-->(DELETE_LTAB)-->+-->|LogTabName|-->+-->
                   |                   |
                   `------>(ALL)----->'
```

Die durch den Namen gekennzeichnete Logische Tabelle oder alle Logischen Tabellen werden gelöscht. Beim Versuch, eine gesicherte oder vorübergehend gesperrte Tabelle zu löschen, erfolgt eine Fehlermeldung.

Beispiel: Die Logische Tabelle "Bolzendaten" soll gelöscht werden.

```
DELETE_LTAB 'Bolzendaten'
```

15.2.4 Löschen einzelner Zeilen

Bei nicht gegen Verändern gesicherten logischen Tabellen (-> Kap.15.2.5) können einzelne oder alle Zeilen gelöscht werden.

DELETE_LTAB_ROW (Interruptfunktion)

```
-->(DELETE_LTAB_ROW)-->|LogTabName|-->+-->|Zeilennummer|-->+-->
                                       |                     |
                                       `------>(ALL)------->'
```

Die durch |Zeilennummer| gekennzeichnete Zeile oder alle Zeilen (ALL) werden gelöscht, wenn die Tabelle nicht gegen Verändern gesichert ist.

Beispiel: In der logischen Tabelle "Bolzendaten" sollen die ersten beiden Zeilen gelöscht werden.

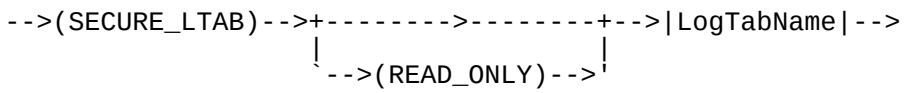
```
DELETE_LTAB_ROW 'Bolzendaten' 1
DELETE_LTAB_ROW 'Bolzendaten' 1
```

Die erste Anweisung löscht die erste Zeile, worauf die ursprünglich zweite Zeile an die erste Stelle rückt und mit der zweiten Anweisung gelöscht wird.

15.2.5 Sichern

Logische Tabellen können gegen vollständiges Löschen und gegen Verändern gesichert werden.

SECURE_LTAB (Interruptfunktion)



Mit SECURE_LTAB ohne Option wird die Tabelle gegen vollständiges Löschen, aber nicht gegen Verändern gesichert. Die Option READ_ONLY bewirkt, daß die Tabelle nur noch gelesen und abgefragt werden kann. Jede Art von Veränderung (z.B. Beschreiben, Löschen vollständig oder in Teilen, Erweitern) ist nicht mehr möglich. READ_ONLY läßt sich wieder rückgängig machen, indem die Anweisung SECURE_LTAB ohne Option auf eine Tabelle angewendet wird. Eine Sicherung gegen Löschen ist hingegen nicht mehr aufhebbar.

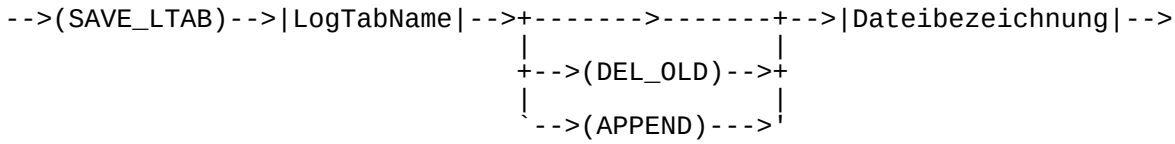
Beispiel: Die Logische Tabelle "Bolzendaten" soll gegen Verändern gesichert werden.

```
SECURE_LTAB READ_ONLY 'Bolzendaten'
```

15.2.6 Speichern

Logische Tabellen können in einer mit INPUT lesbaren Form gespeichert werden.

SAVE_LTAB (Interruptfunktion)



SAVE_LTAB überträgt die durch den Namen gekennzeichnete Logische Tabelle in die angegebene Datei. Es können auch Gerätedateien oder Programme angegeben werden (-> Kap.2.1.3). Die Ausgabe erfolgt im ASCII-Format. Mit den Optionen DEL_OLD und APPEND kann bestimmt werden, ob eine vorhandene Datei gleichen Namens überschrieben oder ergänzet werden soll. Mit der Speicheranweisung wird genaugenommen keine Logische Tabelle gespeichert, sondern eine Datei erzeugt, die Anweisungen zum Anlegen und Beschreiben einer logischen Tabelle mit den gleichen Merkmalen und Daten enthält.

Beispiel: Die Logische Tabelle "Bolzendaten" soll bei einem HP-UX-System auf dem Drucker ausgegeben werden.

```
SAVE_LTAB 'Bolzendaten' '| lp'
```

15.2.7 Abfragen der Tabellengröße

Mit den nachfolgend beschriebenen Funktionen können die Anzahl der Kopffeldeinträge, der Zeilen und der Spalten einer logischen Tabelle abgefragt werden. Bei den Funktionen handelt es sich um integrierte Funktionen, die eine Zahl als Ergebnis liefern. Die Zahl muß in einer Anweisung als Argument oder Parameter verwendet werden. Eine Anweisung, die nur aus einer integrierten Funktion besteht, ergibt keinen Sinn.

Abfragen der Anzahl der Kopffeldeinträge

LTAB_TITLES (integrierte Funktion)

```
-->(LTAB_TITLES)-->|LogTabName|-->
```

LTAB_TITLES liefert die Anzahl der Kopffeldeinträge der angegebenen logischen Tabelle.

Abfragen der Zeilenzahl

LTAB_ROWS (integrierte Funktion)

```
-->(LTAB_ROWS)-->|LogTabName|-->
```

LTAB_ROWS liefert die Anzahl der Zeilen der angegebene logischen Tabelle. Hierbei werden noch nicht belegte Zeilen nur dann mitgezählt, wenn ihnen zumindest noch eine belegte Zeile folgt. Es wird also genaugenommen die Nummer der letzten belegten Zeile abgefragt.

Abfragen der Spaltenzahl

LTAB_COLUMNS (integrierte Funktion)

```
-->(LTAB_COLUMNS)-->|LogTabName|-->
```

LTAB_COLUMNS liefert die Anzahl der Spalten der angegebene logischen Tabelle. Hierbei werden noch nicht belegte Spalten nur dann mitgezählt, wenn ihnen zumindest noch eine belegte Spalte folgt. Es wird also genaugenommen die Nummer der letzten belegten Spalte abgefragt.

Beispiel

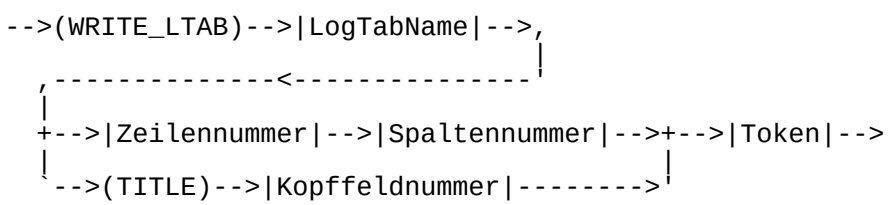
Die Anzahl der Kopffeldeinträge, der Zeilen und der Spalten der logischen Tabelle "Bolzendaten" sollen abgefragt und den Variablen "Titelzahl", "Zeilenzahl" und "Spaltenzahl" zugewiesen werden.

```
LET Titelzahl (LTAB_TITLES 'Bolzendaten')
LET Zeilenzahl (LTAB_ROWS 'Bolzendaten')
LET Spaltenzahl (LTAB_COLUMNS 'Bolzendaten')
```

15.2.8 Schreiben

Eine Logische Tabelle besitzt Kopffelder und Datenfelder, in die Texte oder Zahlen eingetragen werden können.

WRITE_LTAB (Interruptfunktion)



Das Anweisungselement (Token) wird ausgewertet und in das durch |Zeilennummer| und |Spaltennummer| gekennzeichnete Datenfeld bzw. das durch |Kopffeldnummer| gekennzeichnete Kopffeld eingetragen. Der Begriff Token wurde bereits in Kapitel 4 beschrieben. Ein Token ist ein Anweisungselement. Normalerweise handelt es sich dabei um eine Konstante oder einen Ausdruck. Es ist darauf zu achten, daß Logische Tabellen nur Zahlen oder Texte aufnehmen können. Die Auswertung des Token muß also einen Wert mit dem Datentyp NUMBER oder STRING ergeben.

Beispiel

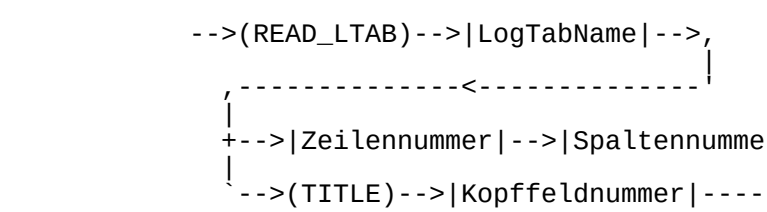
Vom Benutzer sollen Benennung, Nenndurchmesser und Länge eines Bauteils angefordert und in Felder einer logischen Tabelle mit dem Namen "Bauteile_l" eingetragen werden, die für maximal 50 Bauteile anzulegen ist. Die Daten des ersten Bauteils sind in fortlaufende Kopffelder bzw. Zeilen des Datenfelds zu schreiben. Durch Eingabe eines Fragezeichens soll das Ende der Eingabe angezeigt werden. Bei der Anforderung der Bauteilbenennung ist die Nummer des zugehörigen Datensatzes anzuzeigen.

```
DEFINE Eintragen
{Anlegen einer logischen Tabelle mit 50 Zeilen und 2 Spalten}
{Schreiben in Kopffelder und Datenfelder}
{Fischer, 01.07.91, Stand 19.07.93}

  LOCAL Bt  {Bauteilbenennung}
  LOCAL Dn  {Nenndurchmesser}
  LOCAL Lg  {Länge}
  LOCAL It  {Zähler für Anzahl der Datensätze}

  {Anlegen der logischen Tabelle, 50 Zeilen, 2 Spalten}
  CREATE_LTAB 50 2 'Bauteile_l'

  LET It 0  {Zähler wird auf 0 gesetzt}

  LOOP  {Schleife zum Einlesen der Datensätze}
    LET It (It + 1)  {Zähler wird bei jedem Durchlauf um 1 erhöht}
    {Einlesen der Bauteilbenennung (Text)}
    READ STRING
    PROMPT ('Bauteil Nr. ' + STR It + ' : Benennung (Ende = ''?'':') Bt
   EXIT_IF (Bt = '?')  {Abbruch bei Benennung = '?'}
    {Benennung wird in Kopffeld Nr. It eingetragen}
    WRITE_LTAB 'Bauteile_l' TITLE It Bt
    {Einlesen des Nenndurchmessers (Zahl)}
    READ NUMBER 'Nenndurchmesser?' Dn
    {Nenndurchmesser wird in Zeile It, Spalte 1 eingetragen}
    WRITE_LTAB 'Bauteile_l' It 1 Dn
    {Anfordern der Länge}
    READ NUMBER 'Länge?' Lg
    {Länge wird in Zeile It, Spalte 2 eingetragen}
    WRITE_LTAB 'Bauteile_l' It 2 Lg
  END_LOOP

END_DEFINE
```

15.2.9 Lesen

Eine Logische Tabelle besitzt Kopffelder und Datenfelder, aus denen die dort eingetragenen Daten gelesen werden können.

READ_LTAB (Integrierte Funktion



READ_LTAB liest entweder den durch |Kopffeldnummer| gekennzeichneten Kopffeldeintrag oder den durch |Zeilennummer| und |Spaltennummer| gekennzeichneten Datenfeldeintrag der angegebene logischen Tabelle. Bei Angabe eines nicht definierten Kopf- oder Datenfeldes wird eine leere Zeichenfolge ('') gelesen. READ_LTAB ist eine integrierte Funktion, deren Funktionswert ein Text oder eine Zahl ist. Der Funktionswert muß in einer Anweisung als Argument oder Parameter verwendet werden. Eine Anweisung, die nur aus einer integrierten Funktion besteht, ergibt keinen Sinn.

Beispiel

Es ist ein universell verwendbares Makro "Kontroll_ltab" zu schreiben, das die Daten einer logischen Tabelle liest, in die sequentielle Datei "kontroll.fil" schreibt und die Datei anzeigt. Das Makro soll für Logische Tabellen mit bis zu 999 Zeilen und bis zu 999 Spalten geeignet sein.

```
DEFINE Kontroll_ltab
{Lesen der Daten einer logischen Tabelle}
{Ausschreiben in eine sequentielle Datei}
{Fischer, 01.07.91, Stand 19.07.93}

  PARAMETER Lt_Name  {Name der log. Tabelle als Parameter}
  LOCAL Nt           {Anzahl der Kopf-Zeichenfolgen}
  LOCAL Nr           {Anzahl der Zeilen}
  LOCAL Nc           {Anzahl der Spalten}
  LOCAL It           {Zähler für Nr. der Kopf-Zeichenfolge}
  LOCAL Ir           {Zähler für Zeilennummer}
  LOCAL Ic           {Zähler für Spaltennummer}
  LOCAL It_txt       {Nr. Kopf-Zeichenfolge als Text}
  LOCAL Ir_txt       {Zeilennummer als Text}
  LOCAL Ic_txt       {Spaltennummer als Text}
  LOCAL Wert         {Gelesener Wert}

  {Abfrage der Anzahl der Kopffeldeinträge, der Zeilenzahl, der Spaltenzahl}
  LET Nt (LTAB_TITLES Lt_name)
  LET Nr (LTAB_ROWS Lt_name)
  LET Nc (LTAB_COLUMNS Lt_name)

  {Abbruch, wenn Zeilenzahl oder Spaltenzahl > 999. Grund: Ausgabeformat}
  IF ((Nr > 999) OR (Nc > 999))
    BEEP DISPLAY ('Angegebene logische Tabelle zu groß!') END
  END_IF

  {Öffnen der sequentiellen Datei zum Schreiben}
  OPEN_OUTFILE 6 DEL_OLD 'kontroll.fil'

  {Schreiben einer Überschrift und der Tabellendaten}
  WRITE_FILE 6 ('Logische Tabelle ' + Lt_name)
  WRITE_FILE 6 ''
  WRITE_FILE 6 ('Anzahl der Kopf-Zeichenfolgen : ' + STR Nt)
  WRITE_FILE 6 ('Anzahl der Datenzeilen        : ' + STR Nr)
  WRITE_FILE 6 ('Anzahl der Datenspalten       : ' + STR Nc)
  WRITE_FILE 6 ''

  WRITE_FILE 6 'Kopffeldeinträge'
  WRITE_FILE 6 '================'

  {Lesen der Kopffeldeinträge und Schreiben in die Datei}
  LET It 0  {Anfangswertzuweisung für Schleifenzähler}
  {Lesen und Schreiben, bis Zähler = Anzahl der Kopf-Zeichenfolgen}
  WHILE (It < Nt)
    LET It (It + 1)
    WRITE_FILE 6 (READ_LTAB Lt_name TITLE It)
  END_WHILE
  WRITE_FILE 6''

  {Leerzeile und Überschrift für Inhalte der Datenfelder}
  WRITE_FILE 6 ''
  WRITE_FILE 6 'Datenfelder'
  WRITE_FILE 6 '==========='
  WRITE_FILE 6 'Zeile Spalte Wert'
  WRITE_FILE 6 '----- ------ ----'
```

```
  {Lesen der Datenfelder mit zwei geschachtelten WHILE-Schleifen}
  {Äußere Schleife steuert Zeilennummer, innere Schleife steuert Spaltennummer}

  LET Ir 0  {Anfangswertzuweisung für Zähler äußere Schleife}
  WHILE (Ir < Nr)
    LET Ir (Ir + 1)
    {IF-Anweisung erzeugt passendes Ausgabeformat für Zeilennummer}
    IF (Ir < 10)
      LET Ir_txt (STR Ir + '     ')  {Linksbündige Ausgabe mit 5 Leerstellen}
     ELSE_IF (Ir < 100)
      LET Ir_txt (STR Ir + '    ')   {Linksbündige Ausgabe mit 4 Leerstellen}
     ELSE_IF (Ir < 1000)
      LET Ir_txt (STR Ir + '   ')    {linksbündige Ausgabe mit 3 Leerstellen}
    END_IF

    LET Ic 0  {Anfangswertzuweisung für Zähler innere Schleife}
    WHILE (Ic < Nc)
      LET Ic (Ic + 1)
      {IF-Anweisung erzeugt passendes Ausgabeformat für Spaltennummer}
      IF (Ic < 10)
        LET Ic_txt (STR Ic + '      ')  {Linksbündige Ausgabe mit 6 Leerstellen}
       ELSE_IF (Ir < 100)
        LET Ic_txt (STR Ic + '     ')   {Linksbündige Ausgabe mit 5 Leerstellen}
       ELSE_IF (Ic < 1000)
        LET Ic_txt (STR Ir + '    ')    {Linksbündige Ausgabe mit 4 Leerstellen}
      END_IF

      {Lesen des Datenfelds Nr. Ir/Ic und Schreiben einer Textzeile}
      {in die Datei}
      LET Wert (READ_LTAB Lt_name Ir Ic)
      WRITE_FILE 6 (Ir_txt + Ic_txt + STR (Wert))

    END_WHILE  {Ende innere WHILE-Schleife}
  END_WHILE   {Ende äußere WHILE-Schleife}

  {Schließen der Datei und Anzeigen der Datei auf dem Textbildschirm}
  CLOSE_FILE 6
  EDIT_FILE 'kontroll.fil'

END_DEFINE
```

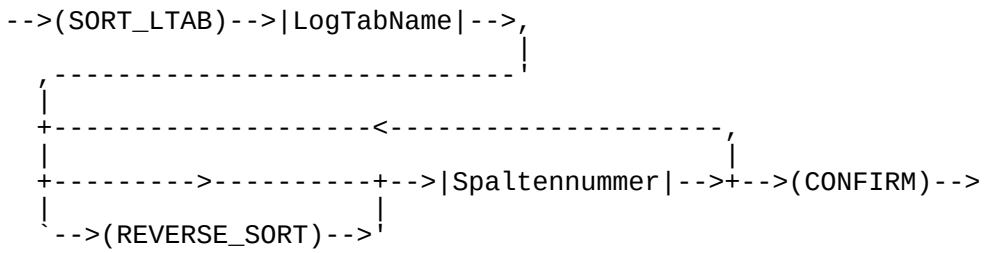
Die Anwendung des Makros könnte beispielsweise bei einer logischen Tabelle folgendes Ergebnis liefern:

```
Logische Tabelle Bauteile_l

Anzahl der Kopf-Zeichenfolgen : 4
Anzahl der Datenzeilen        : 4
Anzahl der Datenspalten       : 2

Kopffeldeinträge
================
Antriebswelle
Antriebswelle
Schraube DIN 912
Kegelstift DIN 1


Datenfelder
===========
Zeile Spalte Wert
----- ------ ----
1     1      5
1     2      250
2     1      75
2     2      300  (usw.)
```

15.2.10 Sortieren

SORT_LTAB ermöglicht das Sortieren von Zeilen in logischen Tabellen. Die Tabellen dürfen nicht gegen Verändern gesichert sein. Der Sortiervorgang benutzt eine oder mehrere Spalten als Kriterium.

SORT_LTAB (Interruptfunktion)



Bei einem Sortiervorgang werden die in einer logischen Tabelle zugelassenen Datentypen in folgende Gruppen geordnet:

- Nullwerte (Felder ohne Eintrag)
- Zahlen
- Texte

Die Reihenfolge der Gruppen kann nicht geändert werden. Ohne besondere Angaben werden die Daten einer Gruppe in aufsteigender Folge sortiert. Die Option REVERSE_SORT bewirkt eine absteigende Reihenfolge. Sie wirkt nur auf die nächste Spalte, muß also gegebenenfalls bei mehreren Spalten wiederholt werden. Wenn mehrere Spalten als Sortierkriterium angegeben werden, haben zuerst genannte Spalten eine höhere Priorität als danach genannte Spalten.

Beispiel

Die Logische Tabelle "Schraubenliste_l" besteht aus 10 Zeilen und 3 Spalten. Die erste Spalte enthält DIN-Bezeichnungen, die zweite Spalte Durchmesser und die dritte Spalte Längen:

```
DIN 912    10   30
DIN 84     10   80
DIN 912    6    50
DIN 84     4    30
DIN 84     10   50
DIN 84     10   40
DIN 84     6    25
DIN 912    8    40
DIN 912    8    25
DIN 912    8    20
```

Es soll eine Sortierung nach folgenden Kriterien vorgenommen werden: Steigende DIN-Bezeichnung, innerhalb einer DIN-Gruppe fallende Durchmesser, innerhalb einer Durchmessergruppe steigende Längen. Die Anweisung müßte lauten:

```
SORT_LTAB 'Schraubenliste_l' 1 REVERSE_SORT 2 3 CONFIRM
```

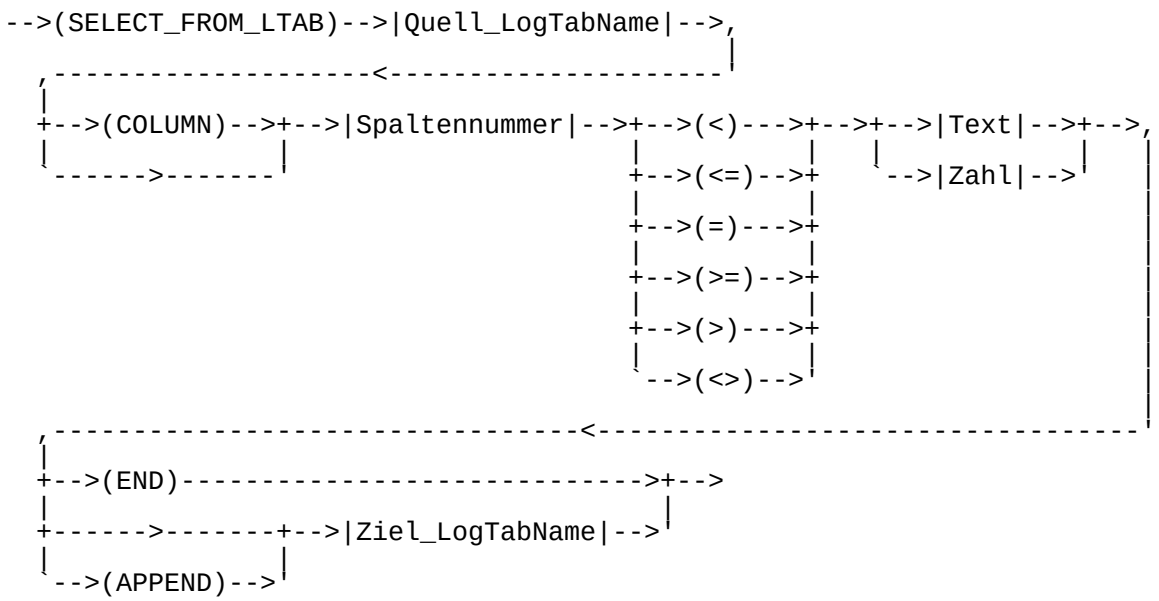
Nach dieser Anweisung liegen die Daten wie folgt vor:

```
DIN 84     10   40
DIN 84     10   50
DIN 84     10   80
DIN 84     6    25
DIN 84     4    30
DIN 912    10   30
DIN 912    8    20
DIN 912    8    25
DIN 912    8    40
DIN 912    6    50
```

15.2.11 Selektieren

MIT SELECT_FROM_LTAB lassen sich Logische Tabellen nach bestimmten Kriterien durchsuchen. Die Nummern der Zeilen, auf die das Kriterium zutrifft, werden automatisch in die systemdefinierte Logische Tabelle "sys_select" geschrieben. Zeilen, auf die das Kriterium zutrifft, können wahlweise in eine andere Logische Tabelle geschrieben werden. Dabei ist ein Überschreiben oder ein Ergänzen dieser Zieltabelle möglich.

SELECT_FROM_LTAB (Interruptfunktion)



|Quell_LogTabName| kennzeichnet die Logische Tabelle, in der die Selektion vorgenommen werden soll. Das Selektierbedingung besteht aus der Angabe einer Spaltennummer (COLUMN *Spaltennummer*) als erstem Vergleichsoperanden, aus einem Vergleichsoperator (<, <= usw.) und aus einem Text bzw. einer Zahl als zweitem Vergleichsoperanden. Vergleichsoperator und zweiter Vergleichsoperand stellen also das Kriterium dar, nach dem selektiert wird.

Zeilen, bei denen die Bedingung zutrifft, bei denen also in der angegebenen Spalte ein Wert vorhanden ist, der dem Kriterium genügt, werden selektiert. Ihre Zeilennummern werden automatisch in die systemdefinierte Logische Tabelle "sys_select" geschrieben. "sys_select" besitzt keinen Kopfbereich und nur einen einspaltigen Datenbereich. Wenn lediglich die Zeilennummern interessieren und der Inhalt der selektierten Zeilen nicht in

eine andere Logische Tabelle geschrieben werden soll, wird die Anweisung mit END beendet. Wird hingegen statt END der Name einer logischen Tabelle angegeben, werden die Zeilen dort hineinkopiert. Eine bereits vorhandene Zieltabelle wird dabei normalerweise überschrieben. Die Option APPEND bewirkt ein Ergänzen vorhandener Zieltabellen. Quelltabelle und Zieltabelle können auch die gleichen Namen haben.

Eine als zweiter Vergleichsoperand verwendeter Text darf auch Joker enthalten. Folgende Jokerzeichen sind zulässig:

Joker	Bedeutung
*	Jede Zeichenfolge einschließlich einer leeren Zeichenfolge
?	Ein einzelnes, beliebiges Zeichen
[..]	Jedes einzelne, in der Klammer aufgeführte Zeichen. Ein durch einen Strich (-) getrenntes Zeichenpaar bezeichnet ein Intervall, zu dem auch die Zeichen des Zeichenpaares gehören.
[!..]	Jedes einzelne Zeichen außer dem bzw. einer der in der Klammer bezeichneten
\	Annuliert die Sonderfunktion jedes Jokers

Beispiel

Eine Logische Tabelle "Teileliste_l" soll folgende Einträge enthalten:

Gestell	St-37	ZN 12-454-119	1	50
Lagerbock	St-37	ZN 12-454-117	2	4.7
Paßfeder	St-70	DIN 6885 -B 12 x 8 x 40	4	0.03
Welle	16MnCr5	ZN 12-300-007	1	2.7
Welle	16MnCr5	ZN 12-300-005	1	4.2
Lagerbock	St-37	ZN 12-454-116	2	5.5
Wälzlager		DIN 625 - 6211	4	0.25
Lagerdeckel	GG-18	ZN 17-229-771	4	0.18

Die erste Spalte enthält Benennungen, die zweite Spalte Werkstoffangaben, die dritte Spalte Zeichnungsnummern bzw. DIN-Bezeichnungen, die vierte Spalte Mengenangaben und die 5 Spalte Gewichtsangaben in kg. Es sind zunächst alle Zeilen zu selektieren und in eine neu anzulegende Logische Tabelle "Eigenfertigung_l" einzutragen, bei denen in der dritten Spalte eine Zeichnungsnummer (kenntlich an der vorangestellten Zeichenfolge "ZN") angegeben ist.

```
CREATE_LTAB 8 5 'Eigenfertigung_l'
SELECT_FROM_LTAB 'Teileliste_l'
  COLUMN 3 = 'ZN*' 'Eigenfertigung_l'
```

Die zweite Anweisung ist wie folgt zu interpretieren: Selektiere aus der logischen Tabelle "Teileliste_l" alle Zeilen, bei denen die in Spalte 3 eingetragenen Texte mit "ZN" beginnen und danach beliebige Zeichen aufweisen. Kopiere die gefundenen Zeilen in die Logische Tabelle "Eigenfertigung_l".

Da die Zeilen 1, 2, 4, 5, 6 und 8 der Selektionsbedingung "erste beiden Zeichen = ZN" genügen, enthält die systemdefinierte Logische Tabelle "sys_select" danach folgende Einträge:

1
2
4
5
6
8

Die Logische Tabelle "Eigenfertigung_l" hat dann folgenden Inhalt:

```
Gestell      St-37      ZN 12-454-119              1    50
Lagerbock    St-37      ZN 12-454-117              2    4.7
Welle        16MnCr5    ZN 12-300-007              1    2.7
Welle        16MnCr5    ZN 12-300-005              1    4.2
Lagerbock    St-37      ZN 12-454-116              2    5.5
Lagerdeckel  GG-18      ZN 17-229-771              4    0.18
```

Weiterhin sind aus der logischen Tabelle "Eigenfertigung_l" alle Zeilen zu selektieren, bei denen die in Spalte 4 eingetragene Menge gößer als 1 ist und der bereits vorhandenen logischen Tabelle "Gleichteile_l" anzuhängen.

```
SELECT_FROM_LTAB 'Eigenfertigung_l'
  COLUMN 4 > 1 APPEND 'Gleichteile_l'
```

Die Anweisung ist wie folgt zu interpretieren: Selektiere aus der logischen Tabelle "Eigenfertigung_l" alle Zeilen, bei denen die in Spalte 4 eingetragenen Zahlen größer als "1" sind und kopiere die gefundenen Zeilen an das Ende der logischen Tabelle "Gleichteile_l".

Da die Zeilen 2, 5 und 6 der Selektionsbedingung "Zahl größer als 1" genügen, enthält die systemdefinierte Logische Tabelle "sys_select" danach folgende Einträge:

2
5
6

Der Logischen Tabelle "Gleichteile_l" werden dann folgende Zeilen angehängt:

```
Lagerbock    St-37      ZN 12-454-117              2    4.7
Lagerbock    St-37      ZN 12-454-116              2    5.5
Lagerdeckel  GG-18      ZN 17-229-771              4    0.18
```

15.2.12 Einblenden, Ausblenden

Eine Logische Tabelle kann in einer oder mehreren mit ihr verknüpften Anzeigetabellen dargestellt werden. Die Anzeigetabellen lassen sich dabei in der Art von POP-UP-Menüs ein- und ausblenden. Anzeigetabellen sind Thema des Kapitels 15.3.

Einblenden

POP_UP_LTAB (Interruptfunktion)

```
-->(POP_UP_LTAB)-->|LogTabName|-->
```

Die Anweisung bewirkt, daß alle mit der logischen Tabelle verknüpften Anzeigetabellen am Bildschirm erscheinen, wenn der Benutzer zur nächsten interaktiven Eingabe aufgefordert wird. Sie wirkt also nicht sofort nach ihrer Eingabe, sondern erst beim nächsten Benutzerdialog. Eine ganz oder teilweise von anderen Tabellen oder Menüs verdeckte Tabelle kann mit POP_UP_LTAB nicht in der Vordergrund gebracht werden. Dies ist nur mit SHOW_TABLE ON (-> Kap.15.3.10) möglich.

Ausblenden

POP_DOWN_LTAB (Interruptfunktion)

```
-->(POP_DOWN_LTAB)-->|LogTabName|-->
```

Alle mit der angegebenen logischen Tabelle verknüpften Anzeigetabellen werden sofort ausgeblendet.

15.2.13 Scrollen

Der Datenbereich einer aus sehr vielen Zeilen bestehenden logischen Tabelle läßt sich häufig nur ausschnittsweise in einer Anzeigetabelle darstellen. Mit SCROLL_LTAB ist ein Blättern in der logischen Tabelle möglich.

SCROLL_LTAB (Interruptfunktion)

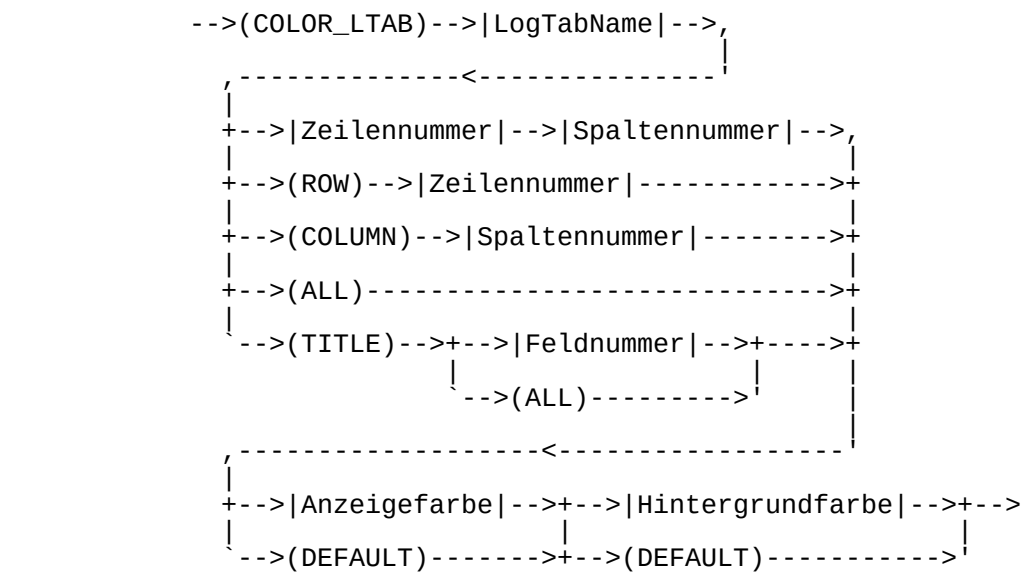
```
-->(SCROLL_LTAB)-->|LogTabName|--|Zeilennummer|-->
```

Die durch die Zeilennummer gekennzeichnete Zeile der angegebenen logischen Tabelle wird in die erste Zeile des Datenbereichs der zugeordneten Anzeigetabelle gerückt.

15.2.14 Farbe ändern

Beim Einrichten oder beim Belegen einer Anzeigetabelle werden für die Felder des Kopf- und Datenbereichs bestimmte Anzeige- und Hintergrundfarben festgelegt. Mit COLOR_LTAB kann erreicht werden, daß die Daten bestimmter Felder, Zeilen oder Spalten einer in der Anzeigetabelle dargestellten logischen Tabelle in davon abweichenden Farben ausgegeben werden. Dadurch lassen sich z.B. bevorzugt auszuwählende Daten kennzeichnen. Die Logische Tabelle darf nicht gegen Verändern gesichert sein.

COLOR_LTAB Interruptfunktion)



Bedeutung und Wirkung der Optionen und Parameter können der nachfolgenden Tabelle entnommen werden. Anzeige- und Hintergrundfarbe müssen mit den Farbbezeichnungen BLACK, BLUE usw., als RGB-Farben oder als HSL-Farben angegeben werden (siehe auch Help-Datei, Schlüsselwort "RGB_COLOR"). Bei Angabe des Schlüsselworts DEFAULT werden die Daten der gekennzeichneten Felder in der Anzeige- oder Hintergrundfarbe dargestellt, die beim Einrichten oder Belegen der Anzeigetabelle festgelegt wurden. DEFAULT hat nur dann eine Sinn, wenn zuvor eine andere Farbfestlegung getroffen wurde, da die Farben der Anzeigetabelle standardmäßig verwendet werden.

OPTIONEN, *Parameter*	Wirkung
Zeilennummer *Spaltennummer*	Anweisung wirkt auf das durch die Zeilennummer und die Spaltennummer gekennzeichnete Datenfeld
ROW *Zeilennummer*	Anweisung wirkt auf die durch die Zeilennummer gekennzeichnete Zeile
COLUMN *Spaltennummer*	Anweisung wirkt auf die durch die Spaltennummer gekennzeichnete Zeile
ALL	Die Anweisung wirkt auf die gesamte Tabelle
TITLE *Kopffeldnummer*	Die Anweisung wirkt auf das durch die Kopffeldnummer gekennzeichnete Kopffeld
TITLE ALL	Die Anweisung wirkt auf alle Kopffelder
Anzeigefarbe	Legt die Farbe des angezeigten Textes fest
Hintergrundfarbe	Legt die Farbe des Hintergrunds fest
DEFAULT	Setzt die Anzeige- oder Hintergrundfarbe auf die für die Anzeigetabelle festgelegten Werte zurück

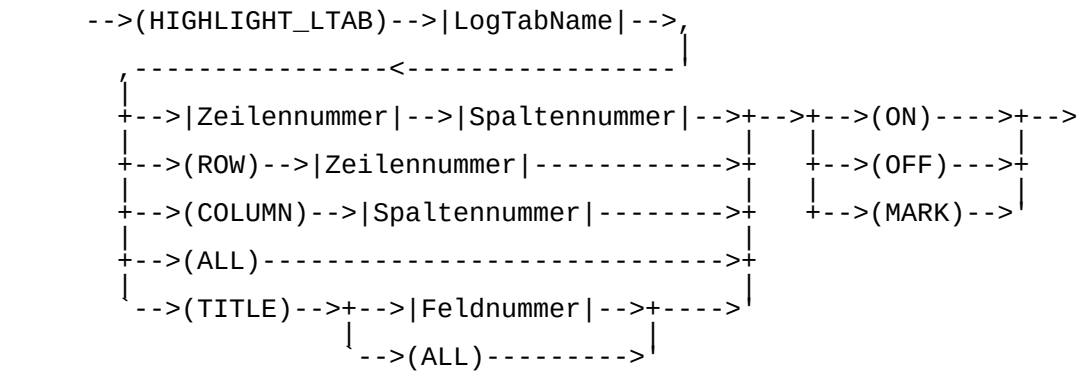
Beispiel: In der logischen Tabelle "Schraubenliste_l" sollen die in den Zeilen 2, 4 und 8 aufgeführten Ausführungen bevorzugt verwendet und daher durch schwarze Schrift auf grünem Hintergrund optisch hervorgehoben werden.

```
COLOR_LTAB 'Schraubenliste_l' ROW 2 BLACK GREEN
COLOR_LTAB 'Schraubenliste_l' ROW 4 BLACK GREEN
COLOR_LTAB 'Schraubenliste_l' ROW 8 BLACK GREEN
```

15.2.15 Hervorheben

Normalerweise werden Daten einer logischen Tabelle in den beim Einrichten oder Belegen der zugeordneten Anzeigetabelle bestimmten Farben angezeigt. HIGHLIGHT_LTAB ermöglicht eine optische Hervorhebung von Daten bestimmter Felder, Zeilen oder Spalten. Als optische Hervorhebung stehen inverse Darstellung, also eine Vertauschung von Anzeige- und Hintergrundfarbe, und Einrahmung zur Verfügung. Dadurch lassen sich z.B. bevorzugt auszuwählende Daten kennzeichnen. Die Logische Tabelle darf nicht gegen Verändern gesichert sein.

HIGHLIGHT_LTAB Interruptfunktion)



Bedeutung und Wirkung der Optionen und Parameter können der nachfolgenden Tabelle entnommen werden.

OPTIONEN, *Parameter*	Wirkung
Zeilennummer *Spaltennummer*	Anweisung wirkt auf das durch die Zeilennummer und die Spaltennummer gekennzeichnete Datenfeld
ROW *Zeilennummer*	Anweisung wirkt auf die durch die Zeilennummer gekennzeichnete Zeile
COLUMN *Spaltennummer*	Anweisung wirkt auf die durch die Spaltennummer gekennzeichnete Zeile
ALL	Die Anweisung wirkt auf die gesamte Tabelle
TITLE *Kopffeldnummer*	Die Anweisung wirkt auf das durch die Kopffeldnummer gekennzeichnete Kopffeld
TITLE ALL	Die Anweisung wirkt auf alle Kopffelder
ON	Hervorhebung in Form einer inversen Darstellung
MARK	Hervorhebung in Form einer Einrahmung
OFF	Hervorhebung wird ausgeschaltet

Beispiel: In der logischen Tabelle "Schraubenliste_l" soll das im Kopffeld 2 enthaltene Datum in Form einer inversen Darstellung optisch hervorgehoben werden.

```
HIGHLIGHT_LTAB 'Schraubenliste_l' TITLE 2 ON
```

15.2.16 Anwendungsbeispiel für Logische Tabellen

ME10 kennt leider nur einfache und keine indizierten Variablen. Es können also keine Felder (Arrays) vereinbart werden. Logische Tabellen lassen sich jedoch auch als zweidimensionale Felder verwendet. Felder mit drei und mehr Dimensionen sind aber nach wie vor nicht möglich.

Im nachfolgenden Beispiel werden zwei Logische Tabellen "A_l" und "B_l" mit den Koeffizienten zweier Matrizen [A] und [B] belegt. Die Koeffizienten werden dabei aus der sequentiellen Datei "ab.dat" gelesen. Danach erfolgt eine Matrizenmultiplikation, bei der die Koeffizienten der Ergebnismatrix [C] nach der untenstehenden Formel berechnet und in die Logische Tabelle "C_l" geschrieben werden.

$$C_{ik} = \sum_{j=1}^{n} (A_{ij} * B_{jk})$$

```
i = Zeilenindex, j = Spaltenindex,
n = Spaltenzahl von Matrix [A] und gleichzeitig auch Zeilenzahl von Matrix [B]
```

Für zwei Matrizen [A] und [B] mit jeweils zwei Zeilen und zwei Spalten würde sich folgende Rechnung ergeben:

$C_{11} = A_{11} * B_{11} + A_{12} * B_{21}$
$C_{12} = A_{11} * B_{12} + A_{12} * B_{22}$
$C_{21} = A_{21} * B_{11} + A_{22} * B_{21}$
$C_{22} = A_{21} * B_{12} + A_{22} * B_{22}$

Das Beispiel wird anhand der Matrizen [A] und [B] mit folgenden Koeffizienten erläutert:

```
[A]  =  -5.9   0.0  +0.2      [B]  =  +5.1  +6.2  -7.3  +8.4  +5.0
        -4.0  +2.8  +2.1              +2.6  -5.7  +2.8  +2.0  +3.3
        +5.7  -2.3  +1.2              -6.6  +7.7  +8.1  -9.1  +2.2
        +2.4  +3.6  -6.7
        +7.1  +2.2  +1.9
```

Zeilen- und Spaltenzahlen sowie die Koeffizienten der Matrizen [A] und [B] werden aus einer sequentiellen Datei mit folgendem Inhalt gelesen:

Zeile	Inhalt	Bedeutung
1	5	Zeilenzahl Matrix [A]
2	3	Spaltenzahl Matrix [A]
3	-5.9	Matrizenkoefizient A_{11}
4	0.0	" A_{12}
5	0.2	" A_{13}
6	-4.0	" A_{21}
7	2.8	" A_{22}
8	2.1	" A_{23}
9	5.7	" A_{31}
10	-2.3	" A_{32}
11	1.2	" A_{33}
12	2.4	" A_{41}
13	3.6	" A_{42}
14	-6.7	" A_{43}
15	7.1	" A_{51}
16	2.2	" A_{52}
17	1.9	" A_{53}

Zeile	Inhalt	Bedeutung
18	3	Zeilenzahl Matrix [B]
19	5	Spaltenzahl Matrix [B]
20	5.1	Matrizenkoefizient B_{11}
21	6.2	" B_{12}
22	-7.3	" B_{13}
23	8.4	" B_{14}
24	5.0	" B_{15}
25	2.6	" B_{21}
26	-5.7	" B_{22}
27	2.8	" B_{23}
28	2.0	" B_{24}
29	3.3	" B_{25}
30	-6.6	" B_{31}
31	7.7	" B_{32}
32	8.1	" B_{33}
33	-9.1	" B_{34}
34	2.2	" B_{35}

Makro "Mamult" stellt das Hauptmakro dar. Zunächst wird die sequentielle Datei "ab.dat" zum Lesen geöffnet. Danach werden Zeilen- und Spaltenzahl von Matrix [A] gelesen und die Logische Tabelle "A_l" mit der gleichen Zeilen- und Spaltenzahl angelegt. Das Lesen der Koeffizienten und das Eintragen der Werte in die Logische Tabelle erfolgt im Untermakro "Einles". Der Name der logischen Tabelle sowie deren Zeilen- und Spaltenzahl wird mit der Parameterliste an das Untermakro übergeben. Im Untermakro "Einles" werden die Koeffizienten von Matrix [A] gelesen und in die zugehörigen Datenfelder der logischen Tabelle "A_l" geschrieben. Die Datenfelder sind dabei mit den Indices "I" und "K" definiert, wobei "I" die Zeilennummer und "K" die Spaltennummer bezeichnet. Die Steuerung des Einlesevorgangs erfolgt mittels zweier geschachtelter Schleifen. Die innere Schleife verändert den Spaltenindex, die äußere Schleife den Zeilenindex. Die Schleifen werden abgebrochen, wenn der jeweilige Index den Grenzwert "Spaltenzahl" bzw. "Zeilenzahl" erreicht hat.

Der gleiche Vorgang wird für Matrix [B] wiederholt. Hierbei erfolgt zunächst eine Prüfung auf Durchführbarkeit der Matrizenmultiplikation. Zwei Matrizen [A] und [B] lassen sich nämlich nur miteinander multiplizieren, wenn die Spaltenzahl von [A] gleich der Zeilenzahl von [B] ist.

Nach dem Einlesevorgang sind die Datenfelder der logischen Tabellen "A_l" und "B_l" mit den Koeffizienten der Matrizen [A] und [B] belegt. Dabei entspricht ein durch Zeilennummer und Spaltennummer gekennzeichnetes Datenfeld einer logischen Tabelle dem mit der gleichen Zeilen- und Spaltennummer gekennzeichneten Koeffizienten der zugehörigen Matrix. Das Feld mit der Zeilennummer 2 und der Spaltennummer 3 der logischen Tabelle "A_l" enthält beispielsweise den Matrizenkoeffizienten A_{23}.

Das Ergebnis der Matrizenmultiplikation soll in analoger Weise in die Logische Tabelle "C_l" geschrieben werden. Für die Ergebnismatrix [C] gilt, daß ihre Zeilenzahl der Zeilenzahl von Matrix [A] und ihre Spaltenzahl der Spaltenzahl von Matrix [B] entspricht. Es wird daher eine Logische Tabelle "C_l" in entsprechender Größe angelegt. Die Matrizenmultiplikation erfolgt im Untermakro "Multator". Um das Untermakro universell verwendbar zu machen, erfolgt die Übergabe der Namen der logischen Tabellen "A_l", "B_l" und "C_l" sowie deren Zeilen- und Spaltenzahlen über eine Parameterliste.

Die Matrizenmultiplikation im Untermakro "Multator" wird mit Hilfe dreier geschachtelter Schleifen durchgeführt. Die äußere Schleife hat den Schleifenzähler "I" und steuert die Zeilennummer der Ergebnismatrix [C] bzw. der zugeordneten logischen Tabelle. Die mittlere Schleife hat den Schleifenzähler "K" und steuert die Spaltenzahl der Ergebnismatrix [C] bzw. der zugeordneten logischen Tabelle. Die innerste Schleife hat den Schleifenzähler "J" und steuert die Summation, die in der vorgenannten allgemeinen Formel dem Summationszeichen "ð" entspricht.

Nach der Matrizenmultiplikation wird das Hauptmakro fortgesetzt. Die Datenfelder der logischen Tabellen "C_l" sind nun mit den Koeffizienten der Ergebnismatrix [C] belegt. Dabei entspricht ein durch Zeilennummer und Spaltennummer gekennzeichnetes Datenfeld der logischen Tabelle dem mit der gleichen Zeilen- und Spaltennummer gekennzeichneten Koeffizienten der Matrix. Das Feld mit der Zeilennummer 2 und der

Spaltennummer 3 der logischen Tabelle "C_l" enthält beispielsweise den Matrizenkoeffizienten C_{23}.

Die Durchführung der Matrizenmultiplikation für die vorgenannten Zahlenwerte führt zu folgendem Ergebnis:

```
                  [B]  =  |  +5.1   +6.2   -7.3   +8.4   +5.0
                          |  +2.6   -5.7   +2.8   +2.0   +3.3
                          |  -6.6   +7.7   +8.1   -9.1   +2.2
          ----------------+-----------------------------------
[A]  =  -5.9   0.0  +0.2 | -31.4  -35.0  +44.7  -51.4  -29.1  =  [C]
        -4.0  +2.8  +2.1 | -27.0  -24.6  +54.1  -47.1   -6.1
        +5.7  -2.3  +1.2 | +15.2  +57.7  -38.3  +32.4  +23.6
        +2.4  +3.6  -6.7 | +65.8  -57.2  -61.7  +88.3   +9.1
        +7.1  +2.2  +1.9 | +29.4  +46.1  -30.3  +46.8  +46.9
```

Die dem Aufruf des Untermakros "Multator" folgenden Anweisungen richten Anzeigetabellen zur Anzeige der logischen Tabellen "A_l", "B_l" und "C_l" ein. Da zu ihrem Verständnis Kenntnisse des nächsten Kapitels notwendig sind, wird ihre Erläuterung erst einmal zurückgestellt.

```
DEFINE Mamult
{Multiplikation zweier Rechteckmatrizen [A] * [B) = [C]}
{Eingabedaten in Datei "ab.dat"}
{Fischer, 24.06.91, Stand 19.07.93}

  LOCAL I      {Index für Zeile}
  LOCAL J      {Summationsindex
  LOCAL K      {Index für Spalte}
  LOCAL Za     {Zeilenzahl  Matrix [A]}
  LOCAL Sa     {Spaltenzahl Matrix [A]}
  LOCAL Zb     {Zeilenzahl  Matrix [B]}
  LOCAL Sb     {Spaltenzahl Matrix [B]}
  LOCAL Zc     {Zeilenzahl  Matrix [C]}
  LOCAL Sc     {Spaltenzahl Matrix [C]}
  LOCAL Wert   {Zwischenvariable}

  OPEN_INFILE 5 'ab.dat' {Öffnen der Eingabedatei}

  READ_FILE 5 Za     {Lesen der Zeilenzahl von [A]}
  READ_FILE 5 Sa     {Lesen der Spaltenzahl von [A]}
  LET Za (VAL Za)    {Typumwandlung String --> Number}
  LET Sa (VAL Sa)

  {Anlegen einer logischen Tabelle "A_l" mit Za Zeilen und Sa Spalten}
  CREATE_LTAB Za Sa 'A_l'

  {Lesen der Matrizenkoeffizienten aus "ab.dat" und Schreiben}
  {in die logische Tablelle "A_l" mit Untermakro "Einles"}
  Einles 'A_l' Za Sa

  READ_FILE 5 Zb      {Lesen der Zeilenzahl von [B]}
  READ_FILE 5 Sb      {Lesen der Spaltenzahl von [B]}
  LET Zb (VAL Zb)     {Typumwandlung String --> Number}
  LET Sb (VAL Sb)

  {Matrizenmultiplikation nur möglich für ZB = Sa}
  IF (Zb <> Sa)
    BEEP
    DISPLAY ('Angegebene Zeilen- oder Spaltenzahlen fehlerhaft!')
    CANCEL
  END_IF

  {Anlegen einer logischen Tabelle "B_l" mit Zb Zeilen und Sb Spalten}
  CREATE_LTAB Zb Sb 'B_l'
```

```
  {Lesen der Matrizenkoeffizienten aus "ab.dat" und Schreiben}
  {in die logische Tablelle "B_l" mit Untermakro "Einles"}
  Einles 'B_l' Zb Sb

  CLOSE_FILE 5     {Schließen der Eingabedatei}

  LET Zc Za       {Zeilen- und Spaltenzahl der Matrix [C] bestimmen sich}
  LET Sc Sb       {aus den Zeilen- und Spaltenzahlen von [A] und [B]}

  {Anlegen einer logischen Tabelle "C_l" mit Zc Zeilen und Sc Spalten}
  CREATE_LTAB Zc Sc 'C_l'

  {Durchführung der Matrizenmultiplikation mit Untermakro "Multator"}
  Multator 'A_l' 'B_l' 'C_l' Za Sa Zb Sb Zc Sc

  {Aufruf des Untermakros "Zeige_x" zur Ausgabe der Matrizen in Anzeigetabellen}
  Zeige_x 'A_l' 'A_a' 'MATRIX [A]'
  Zeige_x 'B_l' 'B_a' 'MATRIX [B]'
  Zeige_x 'C_l' 'C_a' 'MATRIX [C]'

  {Anordnen der Anzeigetabellen}
  MOVE_TABLE 'C_a' RIGHT OF 'A_a' END
  MOVE_TABLE 'B_a' UPPER OF 'C_a' END
  SHOW_TABLE ON 'A_a'
  SHOW_TABLE ON 'B_a'
  SHOW_TABLE ON 'C_a'

  {Einrichten eines Ausschalter-Feldes}
  TABLE_LAYOUT 'Ausschalter_a'
    FRAME_WIDTH 2
    SCROLL_BAR (Text_slot_height + 6)
    0,0
    TITLE_LAYOUT
    (Text_slot_height + 6) '12345678901234567890123456789'
    END
  END

  TABLE_TITLE 'Ausschalter_a'
    BLACK RED ' <<<<< Tabellen Löschen >>>>>'
    ('DELETE_TABLE "A_a" DELETE_TABLE "B_a" ' +
    'DELETE_TABLE "C_a" DELETE_TABLE "Ausschalter_a"')
    1 1
  END

  MOVE_TABLE 'Ausschalter_a' UPPER OF 'A_a' END
  SHOW_TABLE ON 'Ausschalter_a'

END_DEFINE
```

```
DEFINE Einles
  {Lesen der Matrizenkoeffizienten aus "ab.dat" und Schreiben}
  {in eine logische Tablelle}
  {Fischer, 25.06.91, Stand 19.07.93}

  PARAMETER Lt_name     {Name der logischen Tabelle}
  PARAMETER N_z         {Zeilenzahl}
  PARAMETER N_s         {Spaltenzahl}

  LOCAL I               {Zeilenzähler}
  LOCAL K               {Spaltenzähler}
  LOCAL Wert            {Zwischenvariable}

  LET I 0
  LOOP
    LET I (I + 1)
    LET K 0
    LOOP
      LET K (K + 1)
      READ_FILE 5 Wert
      WRITE_LTAB Lt_name I K (VAL (Wert))
     EXIT_IF (K = N_s)
    END_LOOP
   EXIT_IF (I = N_z)
  END_LOOP
END_DEFINE
```

```
DEFINE Multator
  {Durchführung der Matrizenmultiplikation nach der allgemeinen}
  {Formel C(I,K) = Summe [A(I,J) * B(J,K)]}
  {Matrizenkoeffizienten werden aus logischen Tabellen gelesen}
  {und in logische Tabellen eingetragen}
  {Fischer, 24.06.91, Stand 19.07.93}

  PARAMETER Amat  {Matrix [A]}
  PARAMETER Bmat  {Matrix [B]}
  PARAMETER Cmat  {Matrix [C]}
  PARAMETER Ra    {Zeilenzahl  Matrix [A]}
  PARAMETER Ca    {Spaltenzahl Matrix [A]}
  PARAMETER Rb    {Zeilenzahl  Matrix [B]}
  PARAMETER Cb    {Spaltenzahl Matrix [B]}
  PARAMETER Rc    {Zeilenzahl  Matrix [C]}
  PARAMETER Cc    {Spaltenzahl Matrix [C]}

  LOCAL I         {Zeilenzähler}
  LOCAL J         {Summationszähler}
  LOCAL K         {Spaltenzähler}

  LET I 0
  LOOP
    LET I (I + 1)
    LET K 0
    LOOP
      LET K (K + 1)
      LET J 0
      LET Wert 0
        LOOP
        LET J (J + 1)
        LET Wert (Wert + (READ_LTAB Amat I J * READ_LTAB Bmat J K))
       EXIT_IF (J = Ca)
      END_LOOP
      WRITE_LTAB Cmat I K Wert
     EXIT_IF (K = Cc)
    END_LOOP
   EXIT_IF (I = Rc)
  END_LOOP

END_DEFINE


DEFINE Zeige_x
  {Anzeigen der logischen Tabellen in Anzeigetabellen}
  {Fischer, 24.06.91, Stand 19.09.91}

  PARAMETER Lt_name       {Name der logischen Tabelle}
  PARAMETER At_name       {Name der Anzeigetabelle}
  PARAMETER Ueberschrift  {Überschrift-Text}

  {Definition des Tabellen-Layouts}
  TABLE_LAYOUT At_name Lt_name
    ROWS 3
    FRAME_WIDTH 2
    HORIZONTAL BLUE SOLID
    VERTICAL BLUE SOLID
    SCROLL_BAR BLUE GREEN (Text_slot_height + 6)
    0,0
    TITLE_LAYOUT
    (Text_slot_height + 6)  '12345678901234567890123456789'
    1 '                            '
    END
    COLUMN_LAYOUT
    (Text_slot_height + 6) '12345|12345|12345|12345|12345'
    END

  {Definition der Titel-Zeile}
  TABLE_TITLE At_name BLACK GREEN Ueberschrift 'BEEP' 1 1 END

  {Belegung der Spalten mit Werten aus der Logischen Tabelle}
```

```
  TABLE_COLUMN At_name
    COLUMN 1 1 FORMAT '+0.0' RIGHT 'DISPLAY @v1'
    COLUMN 2 2 FORMAT '+0.0' RIGHT 'DISPLAY @v2'
    COLUMN 3 3 FORMAT '+0.0' RIGHT 'DISPLAY @v3'
    COLUMN 4 4 FORMAT '+0.0' RIGHT 'DISPLAY @v4'
    COLUMN 5 5 FORMAT '+0.0' RIGHT 'DISPLAY @v5'
  END

END_DEFINE
```

15.3 Anzeigetabellen

15.3.1 Anzeigetabellen und Bildschirmmenüs im Vergleich

Anzeigetabellen sind, wie zu Beginn von Kapitel 15 bereits dargelegt wurde, im Aussehen und in der Verwendung mit Bildschirmmenüs vergleichbar, besitzen aber zusätzliche Leistungsmerkmale. Die zusätzlichen Leistungsmerkmale bedingen allerdings eine größere Anzahl von Funktionen zu ihrer Realisierung. Zum Einrichten und Verwalten von Bildschirmmenüs genügen 6 Interruptfunktionen mit sehr wenigen Optionen und Parametern. Anzeigetabellen werden mit Hilfe von 12 Interruptfunktionen eingerichtet und verwaltet, von denen einige auch relativ viele Optionen und Parameter aufweisen. (Die 18 Interruptfunktionen der logischen Tabellen müssen im Prinzip auch noch dazu gezählt werden, da Anzeigetabellen in der Regel nur in Verbindung mit logischen Tabellen anwendbar sind.) Verwirrend ist zumindest am Anfang, daß beim Einrichten viele Optionen und Parameter weggelassen oder in beliebiger Reihenfolge angegeben werden können. Es empfiehlt sich, von dieser Möglichkeit nicht allzu sehr Gebrauch zu machen und lieber eine einheitliche Vorgehensweise zu entwickeln, die nur in begründeten Ausnahmefällen verlassen wird.

Anzeigetabellen und Bildschirmmenüs ist gemeinsam, daß zunächst ein Layout (also Position, Größe und Aufteilung) festgelegt werden muß, daß danach eine Belegung mit Daten erfolgt und daß sie beide ein- und ausgeblendet werden können. Anzeigetabellen lassen sich jedoch durch Unterteilung, Farbgebung und Markierung optisch aufwendiger gestalten. Sie sind auch flexibler zu handhaben als Bildschirmmenüs, da sie mit einer vertikalen Rollanzeige versehen und auch nach ihrer Definition noch beliebig vergrößert oder verkleinert werden können.

Felder von Anzeigetabellen lassen sich in gleicher Weise wie Menüfelder mit festen (also konstanten) Anzeige- und Belegungstexten versehen. Beim Antippen der Felder werden die Belegungstexte ausgewertet und an das System übergeben. Darüber hinaus besteht die Möglichkeit, Anzeigetabellen mit logischen Tabellen zu verknüpfen. Anzeige- und Belegungstexte können dann den logischen Tabellen entnommen werden. Bei Veränderung einer logischen Tabelle erfolgt automatisch eine Veränderung aller mit ihr verknüpften Anzeigetabellen. Anzeigetabellen erlauben, im Gegensatz zu Bildschirmmenüs, Zahlenwerte in einem bestimmten Ausgabeformat (Anzahl der Nachkommastellen und Vorzeichenausgabe) darzustellen.

Anzeigetabellen müssen stets mit einem Namen gekennzeichnet werden, bei Bildschirmmenüs ist dies, wie in Kapitel 14 dargelegt wurde, nur erforderlich, wenn gleichzeitig mehrere Menüs verwendet werden sollen. Die Kennzeichnung mit einem Namen ist als Vorteil anzusehen. Sie erlaubt nämlich ein sehr einfaches Verändern einer bestimmten Anzeigetabelle. Während sich Änderungsfunktionen bei Bildschirmmenüs grundsätzlich nur auf das aktuelle Menü auswirken und es zumindest für den ungeübten Benutzer nicht immer ganz leicht ist, das aktuelle Menu zu erkennen, lassen sich Anzeigetabellen einfach über ihren Namen identifizieren und dadurch gezielt verändern. Anzeigetabellen können auch in der Art von Pop-Up-Menüs ein- und ausgeblendet werden. Bei Bildschirmmenüs besteht diese Möglichkeit im Prinzip auch, sie ist jedoch umständlicher zu realisieren. Ein auszublendendes Bildschirmmenü muß zunächst zum aktuellen Menü gemacht werden, bevor es ausgeblendet werden kann.

Zusammenfassend läßt sich sagen, daß Anzeigetabellen insbesondere in Verbindung mit logischen Tabellen sehr leistungsfähige Werkzeuge zum Anzeigen und Verwenden von großen Datenmengen sind. Ihre Programmierung ist meist etwas aufwendiger als die Programmierung von Bildschirmmenüs. Sie haben Vorteile, wenn sehr große, in Bildschirmmenüs nicht unterzubringende Datenmengen vorliegen, wenn auf eine besondere optische Gestaltung und auf eine flexible Verwendung Wert gelegt wird.

Zum Schluß noch eine Vorbemerkung zu den folgenden Kapiteln. In den Syntaxdiagrammen und im Text werden die Anzeigetabellen, auf die sich die beschriebenen Anweisungen beziehen, durch |AnzTabName| gekennzeichnet. Normalerweise wird an dieser Stelle der Name der Anzeigetabelle angegeben. Sichtbare Anzeigetabellen können statt mit ihrem Namen auch durch Antipppen identifiziert werden. Bei nicht sichtbaren Anzeigetabellen besteht diese Möglichkeit natürlich nicht.

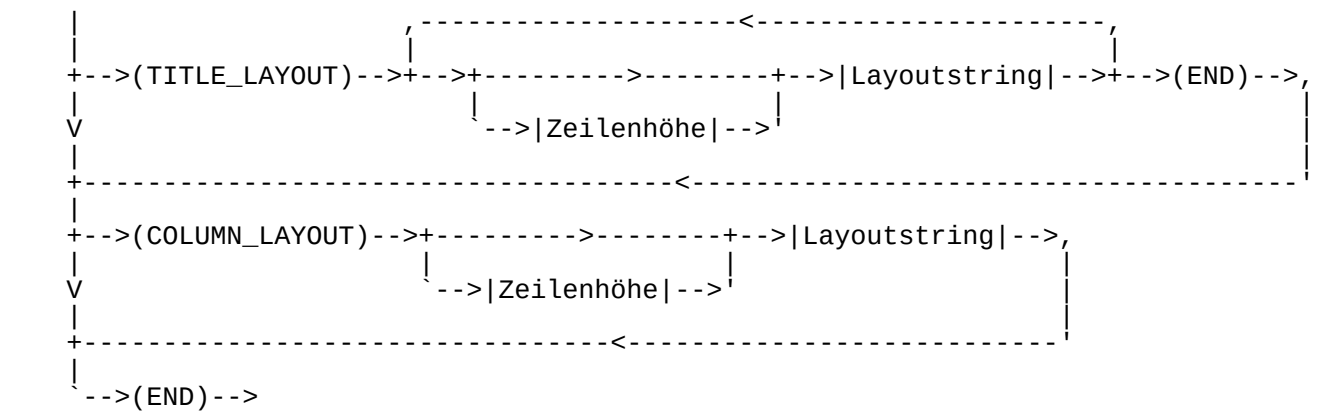
15.3.2 Anlegen und Festlegen des Layouts

Anzeigetabellen werden mit TABLE_LAYOUT unter einem globalen Namen angelegt. Sie sind dann systemweit bekannt. Da das System Anzeigetabellen und Logische Tabellen unterscheidet, kann für eine Anzeigetabelle und eine zugehörige Logische Tabelle der gleiche Name verwendet werden. Das Tabellenlayout, also Position, Größe, Aufteilung und farbige Gestaltung, kann dabei ebenfalls definiert werden. Für zahlreiche Bestimmungsgrößen sind Standardwerte vorgesehen. Eine Anzeigetabelle ist nach ihrer Definition noch nicht sichtbar, sondern sie muß mit einer geeigneten Anweisung eingeblendet werden. Sie belegt jedoch bereits einen Bereich des Bildschirms, so daß eine zuvor an der gleichen Stelle eingerichtete und eingeblendete andere Anzeigetabelle danach zwar noch sichtbar ist, aber nicht mehr ganz problemlos benutzt werden kann.

Während der Entwicklungsphase von Makros kommt es häufig vor, daß das Layout einer Anzeigetabelle geändert wird. Es ist dann dringend zu empfehlen, die zu ändernde Anzeigetabelle vor einer Neudefinition zu löschen. Eine bereits eingerichtete Anzeigetabelle behält nämlich auch bei einer Neudefinition ihre ursprüngliche Größe und Lage bei. Änderungen wirken sich dadurch nur zum Teil aus. Eine vollständige Neudefinition ist nur möglich, wenn die Tabelle zuvor gelöscht wurde.

TABLE_LAYOUT (Interruptfunktion)

(Fortsetzung von vorhergehender Seite)



Optionen, Parameter und ihre Standardwerte

Optionen und *Parameter*	Bedeutung	Standardwert
AnzTabName	Name der Anzeigetabelle	nicht vorhanden
LogTabName	Name einer mit der Anzeigetabelle verknüpften logischen Tabelle	Name der Anzeigetabelle
Anzeigefarbe	Anzeigefarbe im Datenbereich und Farbe des Tabellenrahmens (falls eingerichtet)	weiß
Hintergrundfarbe	Farbe des Feldhintergrunds im Datenbereich	MEPELOOK = 1 : blaugrau MEPELOOK = 0 : schwarz
WIDTH *Tabellenbreite*	Breite der Tabelle in Zeichen (Dezimalzahl möglich)	Zeichenzahl des Layoutstrings, sofern definiert, oder der durch 2 Tabelleneckpunkte bestimmte Wert. Breitenwert aus Tabelleneckpunkten dominiert.
ROWS *Datenzeilenzahl*	Anzahl der im Datenbereich dargestellten Zeilen (Dezimalzahl möglich)	1 Datenzeile, sofern ein Layoutstring des Datenbereichs definiert und die Tabellenhöhe nicht anderweitig festgelegt wurde. Bei zusätzlicher Angabe einer Tabellenhöhe (HEIGHT) oder der Angabe zweier Tabelleneckpunkte ergibt sich die Datenzeilenzahl aus Höhen und Anzahl der Überschriftzeilen, der Höhe einer Datenzeile und der gesamten Tabellenhöhe.

(Tabelle wird fortgesetzt)

Optionen und *Parameter*	Bedeutung	Standardwert
HEIGHT *Tabellenhöhe*	Höhe der Tabelle in Zeichen (Dezimalzahl möglich)	Ein aus der Höhe der Überschrift und aus Anzahl und Höhe der Datenzeilen berechneter Wert oder der sich aus zwei angegebenen Tabelleneckpunkten ergebende Höhenwert. (Höhenwert aus Eckpunkten dominiert.)
erster Eckpunkt *diagon.Eckpunkt*	Definition von Lage und Größe durch 2 Punkte (Angabe in Bildpunktkoordinaten oder durch Digitalisierung)	Linke untere Ecke der Tabelle gleich linke untere Ecke des Grafikbildschirms. Rechte obere Ecke durch Kombinationen von Layoutstring, Tabellenbreite (WIDTH), Zeilenzahl (ROWS) und Tabellenhöhe (HEIGHT) definiert.
FRAME_WIDTH	Breite eines die Tabelle begrenzenden Rahmens	1 Bildpunkt
HORIZONTAL VERTICAL OFF *Linienfarbe* *Linienart*	Horizontale oder vertikale Unterteilung der Tabelle. Bei Angabe der Option OFF wird eine dargestellte Unterteilung ausgeblendet.	Mit Unterteilung MEPELOOK = 1 : taubenblau MEPELOOK = 0 : weiß Linienart SOLID
SCROLL_BAR *Anzeigefarbe* *Hintergrundfarbe* *Breite*	Vertikale Rollanzeige im Datenbereich. Farbangaben und Breite (in Bildpunkten) beziehen sich auf die Darstellung der Bildlaufleiste.	Keine Rollanzeige MEPELOOK = 1 : grau/taubenblau MEPELOOK = 0 : weiß/schwarz Breite 20 Bildpunkte
TITLE_LAYOUT *Zeilenhöhe* *Layoutstring*	Einrichten einer oder mehrerer Zeilen einer Tabellenüberschrift. Zeilenhöhe in Bildpunkten. Zeichen des Layoutstrings beliebig. Einteilung in Felder durch senkrechte Striche "|".	Zeilenhöhe = Standardzeilenhöhe *).Für Layoutstring kein Standardwert vorhanden. Angabe einer Tabellenbreite durch WIDTH oder durch zwei Punkte dominiert über Layoutstring. Layoutstring wird abgeschnitten, falls die Tabellenbreite größer ist und gedehnt, falls Tabellenbreite kleiner ist. Tabellenüberschrift wird ggf. auf Breite des zugehörigen Datenbereichs erweitert.
COLUMN_LAYOUT *Zeilenhöhe* *Layoutstring*	Definition einer Zeile des Datenbereichs. Anzahl der Zeilen wird über ROWS, HEIGHT oder durch Angabe zweier Tabelleneckpunkte bestimmt. Zeilenhöhe in Bildpunkten. Zeichen des Layoutstrings beliebig. Einteilung in Spalten durch senkrechte Striche "|".	Zeilenhöhe = Standardzeilenhöhe *).Für Layoutstring kein Standardwert vorhanden. Angabe einer Tabellenbreite durch WIDTH oder durch zwei Punkte dominiert über Layoutstring. Layoutstring wird abgeschnitten, falls die Tabellenbreite größer ist und gedehnt, falls Tabellenbreite kleiner ist. Zeile wird ggf. auf die Breite der Tabellenüberschrift erweitert.

*) Standardzeilenhöhe : Hängt von der Höhe des verwendeten Standardschriftfont ab (-> Kap.14.3.2). In globaler Variablen "Text_slot_height" gespeichert.

Eine Anzeigetabelle kann bereits beim Einrichten mit einer bestimmten logischen Tabelle verknüpft werden, indem der Name der Logische Tabelle (|LogTabName|) als Parameter angegeben wird. Die Logische Tabelle muß zu diesem Zeitpunkt noch nicht existieren. Ohne explizite Angabe einer logischen Tabelle geht das System davon aus, daß die Anzeigetabelle mit einer logischen Tabelle gleichen Namens verknüpft werden soll und verwendet den Namen der Anzeigetabelle auch als Namen der logischen Tabelle. Die beim Einrichten vorgenommene Verknüpfung kann jederzeit wieder geändert werden (-> Kap.15.3.3).

Anzeige- und Hintergrundfarbe können mit den Farbbezeichnungen BLACK, BLUE usw., als RGB-Farben oder als HSL-Farben angegeben werden (siehe auch Help-Datei, Schlüsselwort "RGB_COLOR"). Diese Werte stellen die Grundfarben einer Anzeigetabelle dar. Farben, die für eine in der Tabelle angezeigte Logische Tabelle festgelegt wurden (vergl. Funktion COLOR_LTAB) oder die beim Belegen von Feldern der Anzeigetabelle (Funktionen TABLE_TITLE und TABLE_COLUMN) angegeben werden, überdecken die Grundfarben.

Die Breite einer Anzeigetabelle bestimmt sich zum einen aus der Breite eines für die Überschrift oder den Datenbereich angegebenen Layoutstrings und zum anderen aus dem durch WIDTH oder durch Angabe zweier Tabelleneckpunkte gegebenen Breitenwert. Ist die Breite des Layoutstrings größer als der Breitenwert, so wird der Layoutstring auf der rechten Seite abgeschnitten. Ist der Breitenwert größer als die Breite des Layoutstrings, wird der Layoutstring so gedehnt, daß der Breitenwert erreicht wird und die Proportionen erhalten bleiben. Der Breitenwert dominiert also in jedem Fall vor der Breite des Layoutstrings.

Beispiel: Die Anweisungen " WIDTH 11 " und " COLUMN_LAYOUT '12|12' " führen zu einer in zwei Felder unterteilte Zeile. Die Felder sind durch einen senkrechten Strich voneinander getrennt und haben jeweils eine Breite von 5 Zeichen.

Die Höhe einer Anzeigetabelle bestimmt sich zum einen aus Höhe und Anzahl der Überschrift- und Datenzeilen und zum anderen aus dem durch HEIGHT oder durch Angabe zweier Tabelleneckpunkte gegebenen Höhenwert. Der Höhenwert dominiert in jedem Fall. Eine zu große Anzahl von Zeilen wird gegebenenfalls nicht vollständig dargestellt. Um Fehler zu vermeiden, sollten entweder nur die Anzahl der Datenzeilen oder nur die Tabellenhöhe oder nur zwei Tabelleneckpunkte angegeben werden.

Die Gesamthöhe einer Anzeigetabelle darf einen Mindestwert nicht unterschreiten. Die Mindesthöhe ist gleich der Standardzeilenhöhe plus 2 Pixel (also "Text_slot_height + 2"). Der Versuch, eine Anzeigetabelle mit zu geringer Höhe einzurichten, führt zu einer Fehlermeldung.

Die verschiedenen Möglichkeiten des Syntaxdiagramms sind wahrscheinlich zunächst mehr verwirrend als hilfreich. Für den Anfang kann man sich an folgende Vorlage halten:

```
TABLE_LAYOUT
Name Anzeigetabelle
Name Logische Tabelle
ROWS Anzahl der Datenzeilen
FRAME_WIDTH 2
SCROLL_BAR
TITLE_LAYOUT  Text_slot_height Layoutstring END
COLUMN_LAYOUT Text_slot_height Layoutstring
END
```

Weiterhin besteht die Möglichkeit, vorhandene Anzeigetabellen als Muster zu verwenden, indem sie SAVE_TABLE in eine Datei gespeichert und danach editiert werden (-> Kap.15.3.8).

15.3.3 Verknüpfen mit einer logischen Tabelle

Eine Anzeigetabelle kann bereits beim Einrichten mit einer bestimmten logischen Tabelle verknüpft werden, indem der Name der logischen Tabelle (|LogTabName|) als Parameter angegeben wird. Ohne explizite Angabe einer logischen Tabelle geht das System davon aus, daß die Anzeigetabelle mit einer logischen Tabelle gleichen Namens verknüpft werden soll und verwendet den Namen der Anzeigetabelle auch als Namen der logischen Tabelle. Die beim Einrichten vorgenommene Verknüpfung kann mit CONNECT_TABLE geändert werden. Als Parameter müssen die Namen der Anzeigetabelle und der logischen Tabelle angegeben werden.

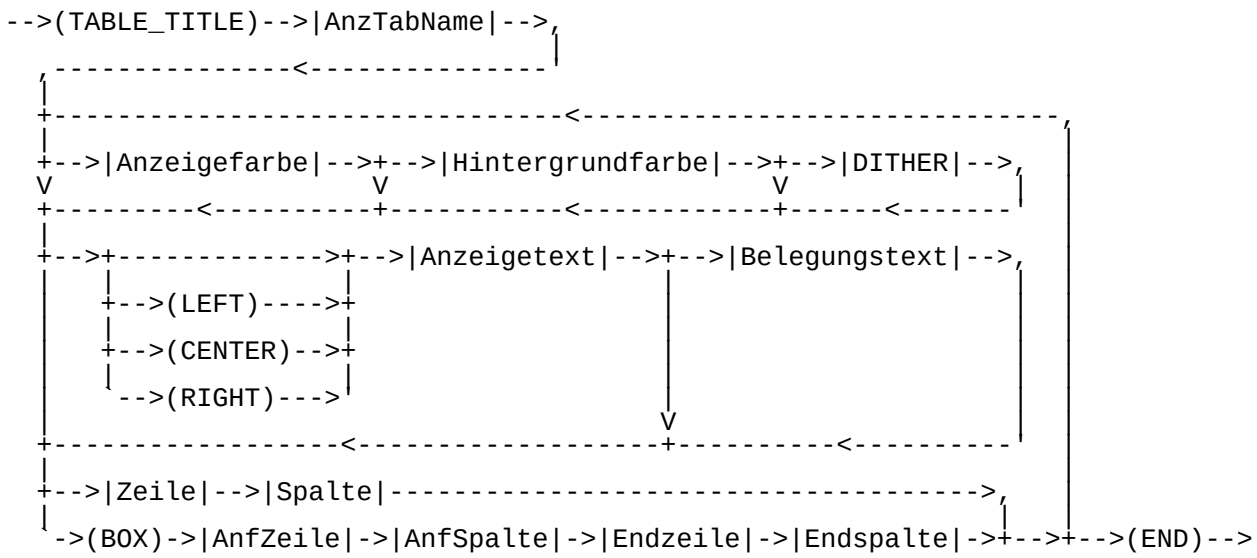
CONNECT_TABLE (Interruptfunktion)

```
-->(CONNECT_TABLE)-->|AnzTabName|-->|LogTabName|-->
```

15.3.4 Belegen der Kopffelder

Den Kopffeldern einer Anzeigetabelle werden nach ihrem Anlegen automatisch Kopffelddaten einer mit ihr verknüpften logischen Tabelle zugeordnet. Dabei findet ein Standardformat Anwendung. Kopffelder der Anzeigetabelle zeigen danach entsprechende Kopffeldeinträge der logischen Tabelle linksbündig in weißer Anzeigefarbe und schwarzer Hintergrundfarbe an und sind auch mit den dort eingetragenen Daten belegt. Eine vom Standardformat abweichende Belegung ist mit TABLE_TITLE möglich. Die Anzeigetabelle darf nicht gegen Ändern gesichert sein (-> Kap.15.2.5).

TABLE_TITLE (Interruptfunktion)



Die Optionen und Parameter haben folgende Bedeutung:

Optionen und *Parameter*	Bedeutung
AnzTabName	Name der Anzeigetabelle
Anzeigefarbe	Farbe des im Kopffeld angezeigten Textes oder Piktogramms
Hintergrundfarbe	Hintergrundfarbe des Kopffeldes
DITHER	Erzeugung von Mischfarben durch "DITHER-Technik" (nur bei Grafikkarten mit geringer Farbtiefe wirksam, vergl. Kap. 14.3.2)
CENTER/LEFT/RIGHT	Ausrichtung des Anzeigetexts
Anzeigetext	Der im Kopffeld angezeigte Text bzw. die Piktogramm-Zeichenfolge
Belegungstext	Zeichenfolge, die beim Antippen des Kopffeldes ausgewertet und danach an das System übergeben wird
Zeile	Zeilennummer des Kopffeldes
Spalte	Spaltennummer des Kopffeldes
BOX	Ermöglicht die Belegung mehrerer Kopffelder
Anf-Zeile	Zeilennummer des ersten angewählten Kopffeldes
Anf-Spalte	Spaltennummer des ersten angewählten Kopffeldes
Endzeile	Zeilennummer des letzten angewählten Kopffeldes
Endspalte	Spaltennummer des letzten angewählten Kopffeldes

Anzeige- und Hintergrundfarbe können mit den Farbbezeichnungen BLACK, BLUE usw., als RGB-Farben oder als HSL-Farben angegeben werden (siehe auch Help-Datei, Schlüsselwort "RGB_COLOR"). Ohne besondere Angaben wird der Anzeigetexte weiß und der Hintergrund bei MEPELOOK = 1 blaugrau und bei MEPELOOK = 0 schwarz dargestellt. Mit Hilfe der Option BOX können mehrere Kopffelder gleichzeitig belegt werden. Anfangszeile, Anfangsspalte, Endzeile und Endspalte bestimmen hier den Bereich. Wenn die Kopffeldbelegung interaktiv erfolgt, können statt der Angabe von Zeilen- und Spaltennummern auch die entsprechenden Felder angetippt werden.

Bei den angezeigten Daten muß es sich grundsätzlich um Text handeln. Daten mit anderem Datentyp sind entsprechend umzuwandeln. Kopffelder von Anzeigetabellen können jedoch mit allen von ME10 zugelassenen Daten (-> Kap.2.7) belegt werden. Die Bezeichnung |Belegungstext| bedeutet, daß die Daten in Textform anzugeben sind, um eine sofortige Auswertung zu unterdrücken. Sie bedeutet nicht, daß nur eine Belegung mit Text möglich ist. In Kapitel 14.3.2 wurde die Bildung von Belegungstext für Bildschirmmenüs eingehend behandelt. Die Ausführungen gelten sinngemäß auch für Anzeigetabellen.

Anzeige- und Belegungstexte des Kopfbereichs können fest vorgegeben oder Kopffeldern einer mit der Anzeigetabelle verknüpften logischen Tabelle entnommen werden. Bei fester Vorgabe werden die Texte, wie bei Bildschirmmenüs bereits gezeigt, als Stringkonstanten, Stringvariablen oder Stringausdrücke angegeben. Eine Übernahme von Kopffeldeinträgen logischer Tabellen erfolgt mit Hilfe sogenannter Stellvertreterzeichen. Ein Stellvertreterzeichen steht stellvertretend für einen Kopffeldeintrag. Feste Vorgabe und Übernahme von Kopffeldeinträgen können in beliebiger Weise miteinander kombiniert werden. So läßt sich zum Beispiel ein Anzeigetext "Schraubendaten nach DIN 84, Auswahlreihe" aus den festen Texten "Schraubendaten nach DIN ", dem Kopffeldeintrag "84" und dem festen Text ", Auswahlreihe" zusammensetzen.

Ein Stellvertreterzeichen kann an einer beliebigen Stelle eines Anzeige- oder Belegungstextes stehen. Es wird bei der Ausführung des Textes durch den Eintrag der logischen Tabelle ersetzt, den es symbolisiert. Es kann also als Textbaustein aufgefaßt werden, dessen aktueller Inhalt einer logischen Tabelle entnommen wird.

Stellvertreterzeichen bestehen aus dem Sonderzeichen "@", dem Buchstaben "s" oder "t" sowie einer Zahl. Ein gültiges Stellvertreterzeichen wäre zum Beispiel "@s5". Die Zahl gibt die Nummer des Kopffeldes der logischen Tabelle an, dessen Eintrag an der betreffenden Stelle des Anzeige- oder Belegungstextes eingesetzt werden soll. Der Buchstabe "s" bewirkt eine unverändertes Einsetzen des Kopffeldeintrags. "t" ist speziell für das Einsetzen von Text vorgesehen und bewirkt, daß Text vor dem Einsetzen in Hochkommas eingeschlossen wird. Zahlen werden stets ohne Hochkommas eingesetzt, "s" und "t" sind hier gleichwertig. "@" zeigt an, daß es sich um ein Stellvertreterzeichen handelt. Das Stellvertreterzeichen "@s5" bewirkt also eine Einsetzen des Eintrags von Kopffeld Nummer 5 in unveränderter Form.

Stellvertreterzeichen	Wirkung
@s*LogTabKopffeldnummer*	Einsetzten des Eintrags des Kopffeldes mit der Nummer *LogTabKopffeldnummer* in den Anzeige- oder Belegungs-text.
@t*LogTabKopffeldnummer*	Bei Zahlen wie @s*LogTabKopffeldnummer*. Text wird vor dem Einsetzen in Hochkommas eingeschlossen.

Beispiele

Wenn das erste Kopffeld der zweiten Kopfzeile einer Anzeigetabelle "Flansch_a" mit dem Anzeigetext "Standarddurchmesser" und mit den Zahlenwert "125" belegt werden soll, müßte die Anweisung wie folgt lauten:

```
TABLE_TITLE 'Flansch_a' 'Standarddurchmesser' '125' 2 1 END
```

Beim Antippen des so belegten Kopffeldes wird die Zahl "125" an das System übergeben.

Wenn die Anzeigetabelle mit der logischen Tabelle "Flansch_l" verbunden ist, in deren drittem Kopffeld die Zahl "125" eingetragen wurde, kann die Anweisung auch wie folgt lauten:

```
TABLE_TITLE 'Flansch_a' 'Standarddurchmesser' '@s3' 2 1 END
```

Ein Belegen mit dem Text "125" ist mit einer der folgenden beiden Anweisung möglich:

```
TABLE_TITLE 'Flansch_a' 'Standarddurchmesser' '"@s3"' 2 1 END
TABLE_TITLE 'Flansch_a' 'Standarddurchmesser' '@t3' 2 1 END
```

Ein weiteres Beispiel

Das nachfolgende, außerordentlich nützliche Makro dient der Begrüßung von neuen CAD-Benutzern in der jeweiligen Landessprache. Die Logische Tabelle "Begruessung_l' enthält Anzeige- und Belegungstexte für die aus vier Kopfzeilen bestehende Anzeigetabelle "Begruessung_a".

```
DEFINE Gruss
  {Makro zum Bgrüßen von CAD-Anwendern in ihrer Landessprache}
  {Fischer, 19.07.93, Stand 24.09.96}

  CREATE_LTAB 'Begruessung_lt'
  WRITE_LTAB 'Begruessung_lt' TITLE 1 'Alle Bayern bitte dieses Feld antippen'
  WRITE_LTAB 'Begruessung_lt' TITLE 2 'Grüß Gott'
  WRITE_LTAB 'Begruessung_lt' TITLE 3 'Die Ostfriesen bitte hier drücken'
  WRITE_LTAB 'Begruessung_lt' TITLE 4 'Moin Moin'
  WRITE_LTAB 'Begruessung_lt' TITLE 5 'Preußen bitte hier'
  WRITE_LTAB 'Begruessung_lt' TITLE 6 'Tag!'
  WRITE_LTAB 'Begruessung_lt' TITLE 7 'Der Rest der Welt darf hier'
  WRITE_LTAB 'Begruessung_lt' TITLE 8 'Hallo!'

  TABLE_LAYOUT 'Begruessung_dt' 'Begruessung_lt'
    GREEN BLACK
    FRAME_WIDTH 2
    TITLE_LAYOUT
    (Text_slot_height + 6) ' | |5   10        20        30        40   | '
    (Text_slot_height + 6) (RPT ' ' 45)
```

```
    (Text_slot_height + 6) (RPT ' ' 45)
    (Text_slot_height + 6) (RPT ' ' 45)
    (Text_slot_height + 6) (RPT ' ' 45)
   END
  END

  {Standardbelegung der Titelzeile}
  Table_control_icons 'Begruessung_dt'
  TABLE_TITLE 'Begruessung_dt'
    BLACK CYAN CENTER 'BEGRÜSSUNGTABELLE' ''1 3
    WHITE BLUE '@s1' 'DISPLAY @t2' 2 1
    BLACK GREEN '@s3' 'DISPLAY "@s4"' 3 1
    WHITE RED '@s5, aber ''n bißchen zackig!' 'DISPLAY @t6' 4 1
    WHITE BLACK '@s7' 'DISPLAY @t8' 5 1
  END

  MOVE_TABLE 'Begruessung_dt' UPPER LEFT END
  SHOW_TABLE ON 'Begruessung_dt'

END_DEFINE
```

Die zweite Kopfzeile der Anzeigetabelle mußte aus Gründen der Akzeptanz unbedingt in den Farben Weiß-Blau gestaltet werden. Es erscheint der Anzeigetext "Alle Bayern bitte dieses Feld antippen". Beim Antippen des Feldes wird "Grüß Gott" in der Hinweiszeile ausgeben. Beim Anzeigetext ist ein direktes Einsetzen des Kopffeldeintrags erforderlich, es wird daher das Stellvertreterzeichen @s1 verwendet. Beim Belegungstext muß der Parameter der Interruptfunktion "DISPLAY" in Hochkommas eingeschlossen werden, da es sich um eine Textkonstante handelt. Es wird daher das Stellvertreterzeichen @t2 verwendet. Ersatzweise könnte hier auch das in doppelte Hochkommas eingeschlossenen Stellvertreterzeichen @s2 werden (siehe Belegung Feld 2).

Der Anzeige- und Belegungstexte der Zeilen 3 und 5 werden wie bei Zeile 2 gebildet. In Zeile 4 wird der Anzeigetext aus einem konstanten Teil und einem aus der logischen Tabelle entnommenen Teil zusammengesetzt. Er lautet "Preußen bitte hier, aber 'n bißchen zackig!". (Zur Erinnerung: Innerhalb von Textkonstanten müssen Hochkommas immer doppelt angegeben werden!)

Die Anzeigetabelle kann durch Eingabe von "DELETE_TABLE Begruessung_a" wieder vom Bildschirm entfernt werden.

15.3.5 Belegung der Datenfelder

Den Datenspalten einer Anzeigetabelle werden nach ihrem Anlegen automatisch Datenspalten einer mit ihr verknüpften logischen Tabelle zugeordnet. Dabei findet ein Standardformat Anwendung. Datenfelder der Anzeigetabelle zeigen danach entsprechende Datenfeldeinträge der logischen Tabelle linksbündig in weißer Anzeigefarbe und schwarzer Hintergrundfarbe an und sind auch mit den dort eingetragenen Daten belegt. Eine vom Standardformat abweichende Belegung ist mit TABLE_COLUMN möglich. Sie erfolgt stets für ganze Spalten. Eine Belegung einzelner Felder ist nicht vorgesehen. Die Anzeigetabelle darf nicht gegen Ändern gesichert sein (-> Kap.15.2.5).

TABLE_COLUMN (Interruptfunktion)

Die Optionen und Parameter haben folgende Bedeutung:

Optionen und *Parameter*	Bedeutung
AnzTabName	Name der Anzeigetabelle
COLUMN *AnzTabSpaltennummer*	Nummer der zu belegenden Datenspalte der Anzeigetabelle
Anzeigefarbe	Farbe der anzuzeigenden Daten
Hintergrundfarbe	Hintergrundfarbe der Datenspalte
DITHER	Erzeugung von Mischfarben durch "DITHER-Technik" (nur bei Grafikkarten mit geringer Farbtiefe wirksam)
CENTER/LEFT/RIGHT	Ausrichtung der angezeigten Daten
FORMAT *Stellenzahl*	Stellenzahl von anzuzeigenden Zahlen
FORMAT *Formatstring*	Der Formatstring bestimmt das Ausgabeformat von Zahlen
LogTabSpaltennummer	Nummer der anzuzeigenden Datenspalte der log. Tabelle
Spalten-Belegungs-text	Zeichenfolge, die beim Antippen des Feldes ausgewertet und dann an das System übergeben wird

COLUMN *AnzTabSpaltennummer* bestimmt die zu belegende Datenspalte der Anzeigetabelle. Eine Belegung einzelner Felder ist nicht vorgesehen. Anzeige- und Hintergrundfarbe können mit den Farbbezeichnungen BLACK, BLUE usw., als RGB-Farben oder als HSL-Farben angegeben werden (siehe auch Help-Datei, Schlüsselwort "RGB_COLOR"). Ohne besondere Angaben wird der Anzeigetexte weiß und der Hintergrund bei MEPELOOK = 1 hellgrau und bei MEPELOOK = 0 schwarz dargestellt.

Der Anzeigetext einer Datenspalte muß grundsätzlich der Datenspalte einer zugeordneten logischen Tabelle entnommen werden. Eine Anzeige anderer Texte ist nicht möglich. Die anzuzeigende Datenspalte der logischen Tabelle wird hierbei über den Parameter *LogTabSpaltennummer* bestimmt. Texte werden rechtsbündig in dem durch die Breite der Datenspalte gegebenen Umfang ausgegeben. Die Ausrichtung läßt sich mit den Optionen CENTER (zentriert), LEFT (linksbündig), und RIGHT (rechtsbündig) beeinflussen.

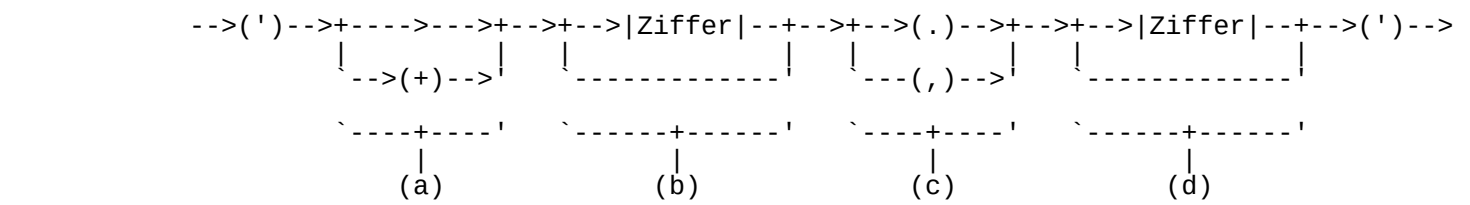
Zahlen werden, sofern die Datenspalte genügen breit ist, standardmäßig rechtsbündig und mit einer Stellenzahl ausgegeben, die der halben Spaltenbreite entspricht. Das Ausgabeformat von Zahlen wird durch FORMAT *Stellenzahl* oder FORMAT *Formatstring*, ihre Ausrichtung durch die Optionen CENTER/LEFT/RIGHT bestimmt.

Eine explizit angegebene Stellenzahl berücksichtigt die Anzahl der Vorkommastellen, einen Dezimalpunkt und die Anzahl der Nachkommastellen. Vorkommastellen werden hierbei, sofern es die Breite der Datenspalte zuläßt, stets in vollem Umfang ausgegeben, auch wenn ihre Anzahl größer als die angegebene Stellenzahl ist. Nachkommastellen werden gegebenenfalls gerundet. Wenn die Anzahl der Vorkommastellen bereits größer als die Spaltenbreite ist, erfolgt statt der Ausgabe einer Zahl die Ausgabe der Zeichen "*".

Beispiele: Bei einer Breite der Datenspalte von 10 Zeichen und linksbündiger Ausrichtung werden Zahlen, abhängig von der mit FORMAT *Stellenzahl* bestimmten Stellenzahl wie folgt ausgegeben:

Zahl	Stellenzahl	Ausgabe
1228,762	8	1228.762
1228,762	6	1228.8
1228,762	4	1229
1228,762	3	1229
1228,762	10	1228.76200
123456789012	10	**********

Ein statt einer Stellenzahl angegebener Formatstring bestimmt die Anzahl der Nachkommastellen, die Art des Dezimalzeichens und legt fest, ob positive Zahlen mit oder ohne Vorzeichen auszugeben sind und ob führende Nullen ausgegeben werden.



a) Negative Zahlen werden stets mit einem Minuszeichen gekennzeichnet. Ein Pluszeichen an der ersten Stelle des Formatstrings bewirkt, daß auch positive Zahlen mit Vorzeichen ausgegeben werden.

b) Die Anzahl der Ziffern vor dem Dezimalzeichen des Formatstrings ist ohne Bedeutung. Vorkommastellen werden, sofern es die Breite der Datenspalte zuläßt, stets in vollem Umfang ausgegeben. Entscheidend ist hingegen, ob direkt vor dem Dezimalzeichen eine Null steht oder nicht. Eine Null bewirkt, daß Zahlen < 1 mit einer führenden Null ausgegeben werden. Keine Ziffer oder eine Ziffer ungleich Null vor dem Dezimalzeichen bewirkt, daß Zahlen < 1 ohne führende Null dargestellt werden.

c) Ein Dezimalzeichen kann wahlweise als Punkt oder als Komma dargestellt werden.

d) Die Anzahl der Ziffern nach dem Dezimalzeichen bestimmt die Anzahl der auszugebenden Nachkommastellen. Wenn hierbei die letzte Ziffer eine Null ist, werden nachfolgende Nullen ausgegeben. Wenn die letzte Ziffer ungleich Null ist, werden nachfolgende Nullen unterdrückt. Beginnen die signifikanten Stellen einer Zahl erst nach der so bestimmten Anzahl von Nachkommastellen, so erfolgt die Ausgabe in Exponentialschreibweise.

Beispiele

Formatstring	Ausgabe bei linksbündiger Ausrichtung und Zahl =							
	0	1	1.2	-1.9	0,1	0,9	0,001	
'0000,0000'	0,0000	1,0000	1,2000	-1,9000	0,1000	0,9000	0,0010	
'0,0000'	0,0000	1,0000	1,2000	-1,9000	0,1000	0,9000	0,0010	
'0,00'	0,00	1,00	1,20	-1,90	0,10	0,90	1,00E-03	
'0,1111'	0	1	1,2	-1,9	0,1	0,9	0,001	
'0,0001'	0	1	1,2	-1,9	0,1	0,9	0,001	
'0,1230'	0,0000	1,0000	1,2000	-1,9000	0,1000	0,9000	0,0010	
',1230'	,0000	1,0000	1,2000	-1,9000	,1000	,9000	,0010	
'0,'	0	1	1	-2	1E-01	1	1E-03	
'0,1'	0	1	1,2	-1,9	0,1	0,9	1E-03	
'+0,00'	0,00	+1,00	+1,20	-1,90	+0,10	+0,90	+1,00E-03	
'+0.00'	0.00	+1.00	+1.20	-1.90	+0.10	+0.90	+1.00E-03	

Datenfelder von Anzeigetabellen können grundsätzlich mit allen von ME10 zugelassenen Daten (-> Kap.2.7) belegt werden. Die Bezeichnung |Belegungstext| bedeutet, daß die Daten in Textform anzugeben sind, um eine sofortige Auswertung zu unterdrücken. Sie bedeutet nicht, daß nur eine Belegung mit Text möglich ist. (In Kapitel 14.3.2 wurde die Bildung von Belegungstext für Bildschirmmenüs eingehend behandelt. Die Ausführungen gelten sinngemäß auch für Anzeigetabellen). Der Spalten-Belegungstext kann in Form von Stringkonstanten, Stringvariablen oder Stringausdrücken fest vorgegeben oder, wie bei der Belegung der Kopffelder bereits beschrieben, ganz oder teilweise aus einer zugeordneten logischen Tabelle entnommen werden. Auch hier kommen Stellvertreterzeichen zur Anwendung.

Einträge von Datenspalten einer mit der Anzeigetabelle verknüpften logischen Tabelle werden mit Hilfe der Stellvertreterzeichen "@v*LogTabSpaltennummer*" oder "@q*LogTabSpaltennummer*" in Belegungstexte eingesetzt. "@v*LogTabSpaltennummer*" bewirkt ein Einsetzen der unveränderten Spalteneinträge. "@q*LogTabSpaltennummer*" ist speziell für das Einsetzen von Text vorgesehen und bewirkt, daß Text vor dem Einsetzen in Hochkommas eingeschlossenen wird. Zahlen werden stets ohne Hochkommas eingesetzt, die Zeichen "v" und "q" sind hier gleichwertig.

Weiterhin kann auch der Eintrag eines Kopffeldes einer mit der Anzeigetabelle verbundenen logischen Tabelle als Belegungstext für alle Felder einer Datenspalte der Anzeigetabelle verwendet werden. In diesem Falle sind die Stellvertreterzeichen @s*LogTabKopffeldnummer*" oder "@t*LogTabKopffeldnummer*" zu verwenden. "@s*LogTabKopffeldnummer*" bewirkt ein Einsetzen des unveränderten Kopffeldeintrags und "@t*LogTabKopffeldnummer*" ein Einsetzen des in Hochkommas eingeschlossenen Eintrags, sofern es sich um Text handelt. Zahlen werden stets ohne Hochkommas eingesetzt, die Zeichen "s" und "t" sind hier gleichwertig.

Stellvertreterzeichen	Wirkung
@v*LogTabSpaltennummer*	Einsetzten der Einträge der Datenspalte mit der Nummer *LogTabSpaltennummer* in den Belegungstext.
@q*LogTabSpaltennummer*	Bei Zahlen wie @v*LogTabSpaltennummer*, Text wird vor dem Einsetzen in Hochkommas eingeschlossen.
@s*LogTabKopffeldnummer*	Einsetzten des Eintrags des Kopffeldes mit der Nummer *LogTabKopffeldnummer* in den Belegungstext.
@t*LogTabKopffeldnummer*	Bei Zahlen wie @s*LogTabKopffeldnummer*. Text wird vor dem Einsetzen in Hochkommas eingeschlossen.

Beispiel

Zu Beginn des Kapitels 15 wurden eine Logische Tabelle mit Schraubendaten und eine zugeordnete Anzeigetabelle dargestellt. Die Logische Tabelle enthält 10 Kopffeldeinträge des Datentyps STRING (Text). Sie bezeichnen die Art und die Nenndurchmesser von Innensechskantschrauben. Die Datenfelder enthalten Einträge des Datentyps NUMBER (Zahl). Sie bezeichnen die zulässigen Längen der Schrauben. Hierbei enthält Spalte 1 die Längen der Schraubengröße M3, Spalte 2 die Längen der Größe M4 usw.

Die Anzeigetabelle soll insgesamt 10 Kopffelder besitzen. Die ersten beiden Kopffelder erstrecken sich über die gesamte Tabellenbreite und zeigen die Texte "Innensechskantschrauben" und "DIN 912, Längentabelle an. Danach soll eine etwas breitere Trennlinie erscheinen. Die wird durch ein weiteres, sich über die gesamte Tabellenbreite erstreckendes Kopffeld mit geringer Höhe erreicht. Die nächsten sechs Kopffelder zeigen die Schraubennenndurchmesser M3 bis M12 an. Das letzte Kopffeld dient wieder als Trennlinie. Die Farben der Kopffelder und ihre Belegungstexte sind in nachfolgender Tabelle dargestellt.

Kopffeld Zeile	Kopffeld Spalte	Belegungstext	Anzeigefarbe/ Hintergrundfarbe
1	1	ohne	Schwarz/Gelb
2	1	ohne	Schwarz/Grün
3	1	ohne	Schwarz/Cyan
4	1	Text: "Nenndurchmesser = M3"	Schwarz/Grün
4	2	Text: "Nenndurchmesser = M4"	Schwarz/Grün
4	3	Text: "Nenndurchmesser = M6"	Schwarz/Grün
4	4	Text: "Nenndurchmesser = M8"	Schwarz/Grün
4	5	Text: "Nenndurchmesser = M10"	Schwarz/Grün
4	6	Text: "Nenndurchmesser = M12"	Schwarz/Grün
5	1	ohne	Schwarz/Cyan

Die Datenspalten der Anzeigetabelle zeigen die für die jeweiligen Nenndurchmesser genormten Längen an. Sie sollen so belegt werden, daß beim Antippen eines Feldes der zugehörige Längenwert einer Variablen "Lg" zugewiesen wird. Wenn z.B. das Datenfeld in Zeile 4 und Spalte 2 angetippt wird, soll folgende Zuweisung erfolgen: "LET Lg 12". Die Längenwerte der Datenfelder seien bereits in den Datenspalten der zugeordneten logischen Tabelle als Zahlen gespeichert.

Das nachfolgende Makro "Dt_standards" definiert Standardeinstellungen für Anzeigetabellen abhängig davon, ob ME10 im "ME-PE-Look" betrieben wird oder nicht. Anweisungen zum Einrichten und Belegen der Tabellen finden sich in Makro "Dn_912" (die Belegung der logischen Tabelle wurde verkürzt wiedergegeben).

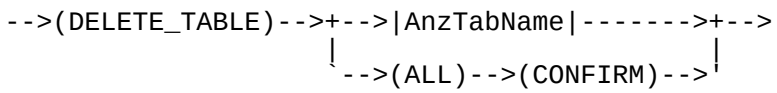
```
DEFINE Dt_standards
{Standardeinstellungen für Anzeigetabellen}
{Fischer, 24.09.96, Stand 24.09.96        }

INQ_ENV 10

 IF (INQ 6) {MEPELOOK=1}
  DEFINE Dt_border          RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_bg              RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_scrollbar_fg    RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_scrollbar_bg    RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_head_fg         RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_head_bg         RGB_COLOR 0.143 0.286 0.667 END_DEFINE
  DEFINE Dt_title_fg        RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_title_bg0       RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_title_bg1       RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_data_fg         RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_data_bg         RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_frame_width     5       END_DEFINE
  DEFINE Dt_horizontal_col WHITE   END_DEFINE
  DEFINE Dt_horizontal_typ SOLID   END_DEFINE
  DEFINE Dt_vertical_col   WHITE   END_DEFINE
  DEFINE Dt_vertical_typ   SOLID   END_DEFINE
 ELSE {MEPELOOK=0}
  DEFINE Dt_border          WHITE   END_DEFINE
  DEFINE Dt_bg              BLACK   END_DEFINE
  DEFINE Dt_scrollbar_fg    WHITE   END_DEFINE
  DEFINE Dt_scrollbar_bg    BLUE    END_DEFINE
  DEFINE Dt_head_fg         BLACK   END_DEFINE
  DEFINE Dt_head_bg         YELLOW END_DEFINE
  DEFINE Dt_title_fg        WHITE   END_DEFINE
  DEFINE Dt_title_bg0       BLACK   END_DEFINE
  DEFINE Dt_title_bg1       BLACK   END_DEFINE
  DEFINE Dt_data_fg         WHITE   END_DEFINE
  DEFINE Dt_data_bg         BLACK   END_DEFINE
  DEFINE Dt_frame_width     1       END_DEFINE
  DEFINE Dt_horizontal_col WHITE   END_DEFINE
  DEFINE Dt_horizontal_typ SOLID   END_DEFINE
  DEFINE Dt_vertical_col   WHITE   END_DEFINE
  DEFINE Dt_vertical_typ   SOLID   END_DEFINE
END_IF

END_DEFINE

{Erzeugen der Standardeinstellungen für Anzeigetabellen}
Dt_standards
```

```
DEFINE Dn_912
{Schraubenabmessungen DIN 912}
{Fischer, 19.09.91, Stand 24.09.96}

 CREATE_LTAB 16 8 'In'

 WRITE_LTAB 'In' TITLE  1 'Innensechskantschrauben'
 WRITE_LTAB 'In' TITLE  2 'DIN 912'
 WRITE_LTAB 'In' TITLE  3 'M3'
 WRITE_LTAB 'In' TITLE  4 'M4'
 WRITE_LTAB 'In' TITLE  5 'M5'
 WRITE_LTAB 'In' TITLE  6 'M6'
 WRITE_LTAB 'In' TITLE  7 'M8'
 WRITE_LTAB 'In' TITLE  8 'M10'
 WRITE_LTAB 'In' TITLE  9 'M12'
 WRITE_LTAB 'In' TITLE 10 'M16'

 WRITE_LTAB 'In' 1 1 5     WRITE_LTAB 'In' 1 2 6
 WRITE_LTAB 'In' 1 3 8     WRITE_LTAB 'In' 1 4 10
 WRITE_LTAB 'In' 1 5 12    WRITE_LTAB 'In' 1 6 16
 WRITE_LTAB 'In' 1 7 20    WRITE_LTAB 'In' 1 8 25

 WRITE_LTAB 'In' 2 1 6     WRITE_LTAB 'In' 2 2 8
 WRITE_LTAB 'In' 2 3 10    WRITE_LTAB 'In' 2 4 12
 WRITE_LTAB 'In' 2 5 16    WRITE_LTAB 'In' 2 6 20
 WRITE_LTAB 'In' 2 7 25    WRITE_LTAB 'In' 2 8 30

 WRITE_LTAB 'In' 3 1 8     WRITE_LTAB 'In' 3 2 10
 WRITE_LTAB 'In' 3 3 12    WRITE_LTAB 'In' 3 4 16
 WRITE_LTAB 'In' 3 5 20    WRITE_LTAB 'In' 3 6 25
 WRITE_LTAB 'In' 3 7 30    WRITE_LTAB 'In' 3 8 35
.
.
 WRITE_LTAB 'In' 16 7 ''   WRITE_LTAB 'In' 16 8 160

 TABLE_LAYOUT 'Innensechskant_a' 'In'
  Dt_border Dt_bg
  ROWS 13
  FRAME_WIDTH Dt_frame_width
  HORIZONTAL Dt_horizontal_col Dt_horizontal_typ
  VERTICAL Dt_vertical_col Dt_vertical_typ
  SCROLL_BAR Dt_scrollbar_fg Dt_scrollbar_bg
  TITLE_LAYOUT      ' | |                  | '
                    '                      '
                    '                      '
                  4 '                      '
                    '123|123|123|123|123|123'
                  4 '                      ' END
  COLUMN_LAYOUT     '123|123|123|123|123|123'
 END

 Table_control_icons 'Innensechskant_a'
 TABLE_TITLE 'Innensechskant_a'
  Dt_head_fg Dt_head_bg '' '' 1 3
  Dt_title_fg Dt_title_bg0 '@s1' 2 1
  Dt_title_fg Dt_title_bg0 '@s2, Längentabelle' 3 1
  Dt_title_fg Dt_title_bg1 '' '' 4 1
  Dt_title_fg Dt_title_bg1 '@s3' '"Nenndurchmesser = @s3"' 5 1
  Dt_title_fg Dt_title_bg1 '@s4' '"Nenndurchmesser = @s4"' 5 2
  Dt_title_fg Dt_title_bg1 '@s6' '"Nenndurchmesser = @s6"' 5 3
  Dt_title_fg Dt_title_bg1 '@s7' '"Nenndurchmesser = @s7"' 5 4
  Dt_title_fg Dt_title_bg1 '@s8' '"Nenndurchmesser = @s8"' 5 5
  Dt_title_fg Dt_title_bg1 '@s9' '"Nenndurchmesser = @s9"' 5 6
  '' '' 5 1
 END

 TABLE_COLUMN 'Innensechskant_a'
  COLUMN 1 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 1 'LET Lg @v1'
  COLUMN 2 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 2 'LET Lg @v2'
  COLUMN 3 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 4 'LET Lg @v4'
  COLUMN 4 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 5 'LET Lg @v5'
  COLUMN 5 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 6 'LET Lg @v6'
  COLUMN 6 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 7 'LET Lg @v7'
 END

 MOVE_TABLE 'Innensechskant_a' UPPER LEFT END
 SHOW_TABLE ON 'Innensechskant_a'

END_DEFINE
```

15.3.6 Löschen

Nicht gegen Ändern gesicherte Anzeigetabellen können gelöscht werden.

DELETE_TABLE (Interruptfunktion)



Die durch den Namen gekennzeichnete Anzeigetabelle oder alle (ALL) Anzeigetabellen werden gelöscht. Beim Versuch, eine gesicherte oder nicht vorhandene Tabelle zu löschen, erfolgt eine Fehlermeldung.

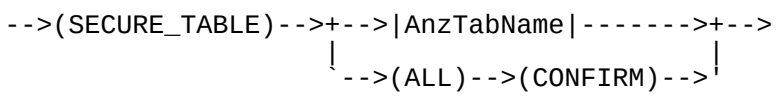
Beispiel: Die Anzeigetabelle "Innensechskant_a" soll gelöscht werden.

```
DELETE_TABLE 'Innensechskant_a'
```

15.3.7 Sichern

Anzeigetabellen können gegen Verändern und damit auch gegen Löschen gesichert werden.

SECURE_TABLE (Interruptfunktion)



Die durch den Namen gekennzeichnete Anzeigetabelle oder alle (ALL) Anzeigetabellen werden gegen Verändern gesichert. Die Sicherung ist nicht mehr aufhebbar.

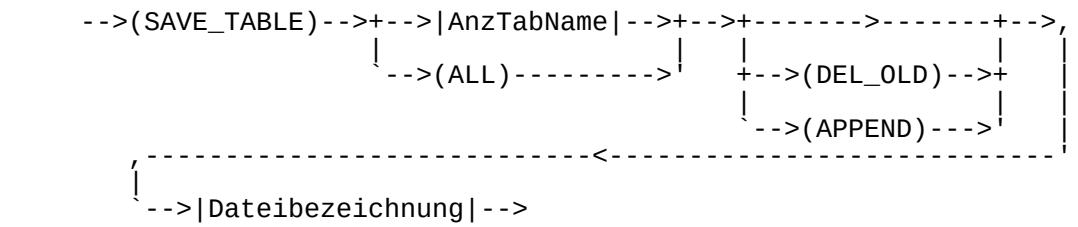
Beispiel: Die Anzeigetabelle "Innensechskant_a" soll gesichert werden.

```
SECURE_TABLE 'Innensechskant_a'
```

15.9.8 Speichern

Anzeigetabellel können in einer mit INPUT lesbaren Form gespeichert werden.

SAVE_TABLE (Interruptfunktion)



SAVE_TABLE überträgt die durch den Namen gekennzeichnete Anzeigetabelle oder alle (ALL) Anzeigetabellen in die angegebene Datei (-> Kap.2.1.3). Die Ausgabe erfolgt im ASCII-Format. Mit den Optionen DEL_OLD und APPEND kann bestimmt werden, ob eine vorhandene Datei gleichen Namens überschrieben oder ergänzt werden soll. Mit der Speicheranweisung wird genaugenommen keine Anzeigetabelle gespeichert, sondern eine Datei erzeugt, die Anweisungen zum Einrichten einer Anzeigetabelle mit den gleichen Merkmalen und Daten enthält. Auf diese Weise können vorhandene Anzeigetabellen unter einem anderen Namen gespeichert werden, um sie dann als Vorlage für neue Tabellen zu verwenden.

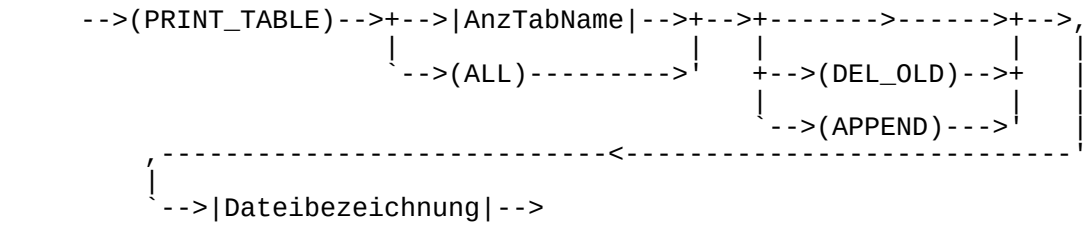
Beispiel: Die Anzeigetabelle Logische Tabelle "Innensechskant_a" soll der vorhandenen Datei "normteil.tab" angehängt werden.

```
SAVE_TABLE 'Innensechskant_a' APPEND 'normteil.tab'
```

15.3.9 Ausdrucken

Der auf dem Bildschirm sichtbare Teil von Anzeigetabellen kann in vereinfachter Form ausgedruckt oder in einer Datei gespeichert werden.

PRINT_TABLE (Interruptfunktion)



PRINT_TABLE überträgt die Bildschirmdarstellung der durch den Namen gekennzeichneten Anzeigetabelle oder alle (ALL) Anzeigetabellen in die angegebene Datei (-> Kap.2.1.3). Die Ausgabe erfolgt in vereinfachter Form mit Textzeichen und ohne Farben und Bildlaufleiste. Mit den Optionen DEL_OLD und APPEND kann bestimmt werden, ob eine vorhandene Datei gleichen Namens überschrieben oder ergänzt werden soll.

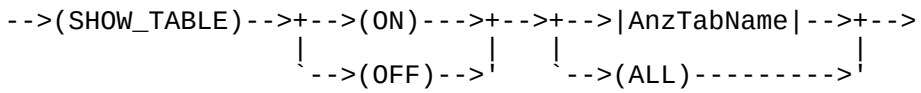
Beispiel: Der auf dem Bildschirm sichtbare Teil der Anzeigetabelle "Innensechskant_a" soll bei einem HP-UX-System auf dem Drucker ausgegeben werden.

```
PRINT_TABLE 'Innensechskant_a' '| lp'
```

15.3.10 Einblenden, Ausblenden

Anzeigetabellen können in der Art von POP-UP-Menüs ein- und ausgeblendet werden.

SHOW_TABLE (Interruptfunktion)



Die Option ON bewirkt ein Anzeigen, die Option OFF ein Ausblenden der mit dem Namen gekennzeichneten Anzeigetabelle oder aller (ALL) Anzeigetabellen. Wurden beim Anzeigen einer Tabelle Zeichnungselemente verdeckt, so werden sie nach dem Ausblenden wieder sichtbar. Tabellen, die zwar eingeblendet sind, aber ganz oder teilweise durch andere Tabellen oder Bildschirmmenüs verdeckt werden, lassen sich mit SHOW_TABLE ON vollständig anzeigen.

SHOW_TABLE wirkt auf einzelne oder alle existierenden Anzeigetabellen. Die in Zusammenhang mit logischen Tabellen beschriebenen Funktionen POP_UP_LTAB und POP_DOWN_LTAB blenden hingegen alle mit einer logischen Tabelle verbundenen Anzeigetabellen ein oder aus.

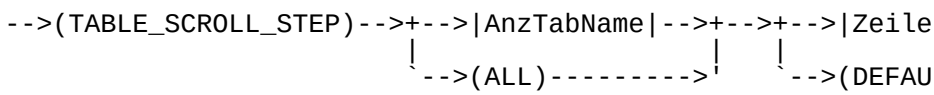
Beispiel: Der Anzeigetabelle "Innensechskant_a" soll angezeigt werden.

```
SHOW_TABLE ON 'Innensechskant_a'
```

15.3.11 Verändern der Schrittweite beim Blättern

Wenn der Datenbereich einer Anzeigetabelle als vertikale Rollanzeige eingerichtet wurde, ist ein Blättern durch Antippen der Kästchen am oberen und unteren Ende der Bildlaufleiste möglich. Normalerweise wird hierbei um eine Anzeigeseite vor- oder zurückgeblättert. Mit TABLE_SCROLL_STEP läßt sich die Schrittweite beim Blättern einstellen.

TABLE_SCROLL_STEP (Interruptfunktion)



Die Schrittweite beim Blättern der durch den Namen gekennzeichneten Anzeigetabelle oder aller (ALL) Anzeigetabellen wird auf die angegebene Zeilenzahl eingestellt. Eine negative Zeilenzahl kehrt die Blätterrichtung um. Die Angabe der Option DEFAULT bewirkt ein Zurücksetzen auf den Normalzustand, bei dem das Blättern so erfolgt, daß die letzte Zeile an den Anfang des Datenbereichs gelangt.

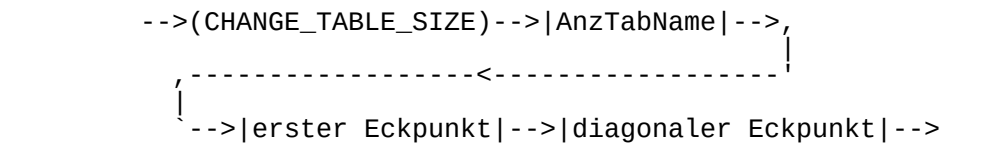
Beispiel: Die Schrittweite beim Blättern der Anzeigetabelle "Innensechskant_a" soll so eingestellt werden, daß um jeweils 8 Zeilen geblättert wird.

```
TABLE_SCROLL_STEP 'Innensechskant_a' 8
```

15.3.12 Verändern der Größe

Beim Einrichten einer Anzeigetabelle werden Breite und Höhe einer Anzeigetabelle festgelegt. In Kapitel 15.3.2 wurde gezeigt, daß dies durch unterschiedliche Angaben geschehen kann. Mit CHANGE_TABLE_SIZE kann die Größe einer nicht gegen Ändern geschützten Anzeigetabelle nachträglich verändert werden.

CHANGE_TABLE_SIZE (Interruptfunktion)



Die durch den Namen gekennzeichnete Anzeigetabelle wird so verändert, daß ihre linke obere Ecke mit der linken oberen Ecke des durch die zwei diagonal zueinander liegende Eckpunkte definierten Rechtecks fällt. Die Eckpunkte können in Bildpunkten angegeben oder durch Digitalisierung bestimmt werden.

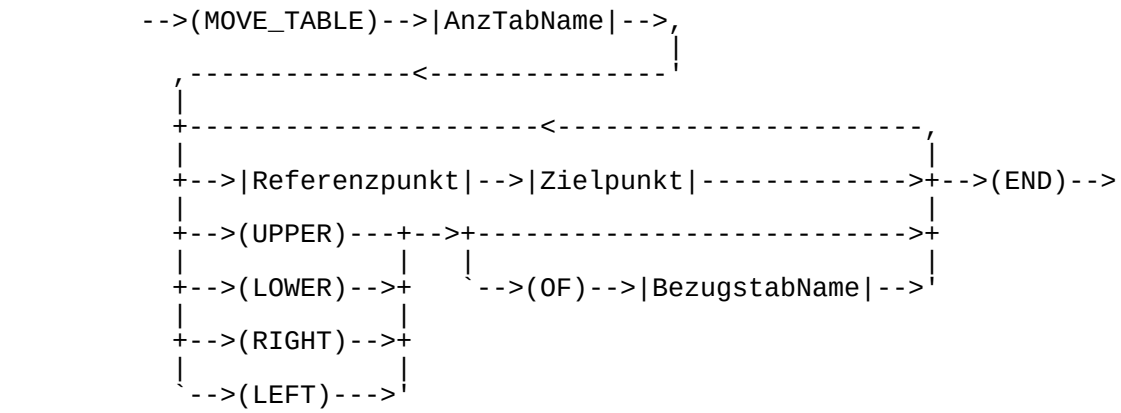
Beispiel: Die Anzeigetabelle "Innensechskant_a" soll auf eine Breite von 400 Bildpunkten und eine Höhe von 600 Bildpunkten verändert werden. Ihre linke untere Ecke soll dabei mit den Bildpunktkoordinaten x = 200 / y = 100 zusammenfallen.

```
CHANGE_TABLE_SIZE 'Innensechskant_a' 200,100 600,700
```

15.3.13 Verschieben

Beim Einrichten einer Anzeigetabelle wird auch ihre Lage festgelegt. Ohne besondere Angaben wird sie in der linken unteren Ecke des Grafikbildschirms positioniert. Mit MOVE_TABLE kann die Lage nachträglich verändert werden.

MOVE_TABLE (Interruptfunktion)



Die Optionen und Parameter haben folgende Bedeutung:

Optionen und *Parameter*	Bedeutung
AnzTabName	Name der zu verschiebenden Anzeigetabelle
Referenzpunkt	Punkt, auf den sich die Verschiebung bezieht
Zielpunkt	Zielpunkt der Verschiebung
UPPER	Verschiebung an den oberen Rand des Grafikbildschirms
LOWER	Verschiebung an den unteren Rand des Grafikbildschirms
RIGHT	Verschiebung an den rechten Rand des Grafikbildschirms
LEFT	Verschiebung an den linken Rand des Grafikbildschirms
UPPER OF *BezugstabName*	Verschiebung an den oberen Rand der Bezugstabelle
LOWER OF *BezugstabName*	Verschiebung an den unteren Rand der Bezugstabelle
RIGHT OF *BezugstabName*	Verschiebung an den rechten Rand der Bezugstabelle
LEFT OF *Bezugstabname*	Verschiebung an den linken Rand der Bezugstabelle

Die Verschiebung der durch |AnzTabName| gekennzeichneten Anzeigetabelle kann durch Angabe eines Referenzpunktes und eines Zielpunktes definiert werden. Referenz- und Zielpunkt bestimmen hierbei den Verschiebungsvektor. Sie werden in Bildpunktkoordinaten oder durch Digitalisierung angegeben. Die Optionen UPPER, LOWER, RIGHT oder LEFT bewirken ohne weitere Optionen und Parameter ein Verschieben der Anzeigetabelle an den oberen, unteren, rechten oder linken Rand des Grafikbildschirms. Eine Verwendung dieser Optionen zusammen mit der Option OF und dem Namen einer weiteren Anzeigetabelle (|BezugstabName|) als Parameter führt zum Verschieben an der oberen, unteren, rechten oder linken Rand der Bezugstabelle. Mehrere, aufeinanderfolgende Verschiebungsangaben werden addiert und erst nach Angabe der Option END ausgeführt.

Beispiel: Die Anzeigetabelle "Innensechskant_a" soll 10 Bildpunkte unterhalb der ebenfalls existierenden Anzeigetabelle "Sechskant_a" angeordnet werden.

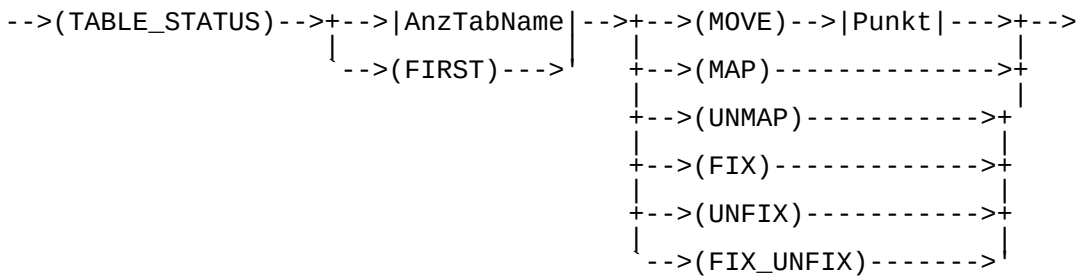
```
MOVE_TABLE 'Innensechskant_a' LOWER OF 'Sechskant_a' 0,100 0,90
END
```

Achtung: Bildpunktkoordinaten können nur positiv sein!

15.3.13 Festlegen und Ändern des Tabellenstatus

Anzeigetabellen können Hilfe der Interruptfunktion TABLE_STATUS bewegt, angezeigt, ausgeblendet, fixiert und nichtfixiert werden.

TABLE_STATUS (Interruptfunktion)



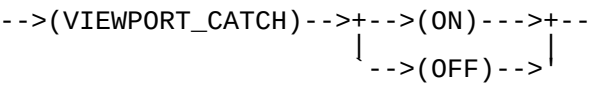
Die Optionen und Parameter haben folgende Bedeutung:

Optionen und *Parameter*	Bedeutung
AnzTabName	Name der Anzeigetabelle
FIRST	Erste Tabelle in einer systeminternen Liste
MOVE *Punkt*	Die Tabelle wird an den angegebenen Punkt verschoben
MOVE *Punkt*	Die Tabelle wird an den angegebenen Punkt verschoben
MAP	Die Tabelle wird vollständig abgebildet
UNMAP	Die Tabelle wird ausgeblendet
FIX	Die Tabelle wird an der momentanen Position fixiert
UNFIX	Die Tabelle wird an ihrer Stammposition fixiert
FIX_UNFIX	Schaltet zwischen FIX und UNFIX um

Mit Hilfe von Feldern der ersten Titelzeile können die Standard-Anzeigetabellen fixiert/nichtfixiert, bewegt und ausgeblendet werden. Dabei findet ein Teil der vorgenannten Funktionen Verwendung. Die Belegung der ersten Titelzeile erfolgt ähnlich wie bei Bildschirmmenüs mit dem Systemmakro "Table_control_icons". "Table_control_icons" benötigt den Namen der Anzeigetabelle als Parameter.

Wenn Anzeigetabellen bewegt werden, ist es manchmal wünschenswert, daß sie exakt in die Position anderer Menüs, Fenster oder Anzeigetabellen plaziert werden können. Dies läßt sich mit Hilfe der Funktion VIEWPORT_CATCH erreichen.

VIEWPORT_CATCH (Interruptfunktion)



Bei der Standardeinstellung ON rasten neu erstellte, vergrößerte, verkleinerte oder verschobene Fenster, Bildschirmmenüs oder Anzeigetabellel in die Position anderer Fenster, Bildschirmmenüs oder Anzeigetabellen ein. Der Fangbereich beträgt 8 Bildpunkte. Mit VIEWPORT_CATCH OFF wird das Einrasten ausgeschaltet.

15.4 Anwendungsbeispiele

15.4.1 Darstellung von Matrizenkoeffizienten

Kapitel 15.2.16 zeigt als Anwendungsbeispiel für Logische Tabellen ein Makro zur Multiplikation von Matrizen. Eine Beschreibung der Anweisungen zum Anzeigen der Matrizenkoeffizienten wurde zurückgestellt und soll an dieser Stelle erfolgen. Die nachstehenden Ausführungen beziehen sich auf die dem Aufruf des Untermakros "Multator" folgenden Anweisungen des Hauptmakros "Mamult" und auf das Makro "Zeige_x".

Die Koeffizienten der miteinander zu multiplizierenden Matrizen [A] und [B] und die Koeffizienten der Ergebnismatrix [C] sind in den logischen Tabellen "A_l", "B_l" und "C_l" gespeichert und werden in den Anzeigetabellen "A_a", "B_a" und "C_a" angezeigt. Hierzu wird nach Durchführung der Matrizenmultiplikation mit Hilfe des Untermakros "Multator" das zum Einrichten und Belegen der drei Anzeigetabellen vorgesehene Makro "Zeige_x" aufgerufen. Die Namen der logischen Tabellen, der Anzeigetabellen und die

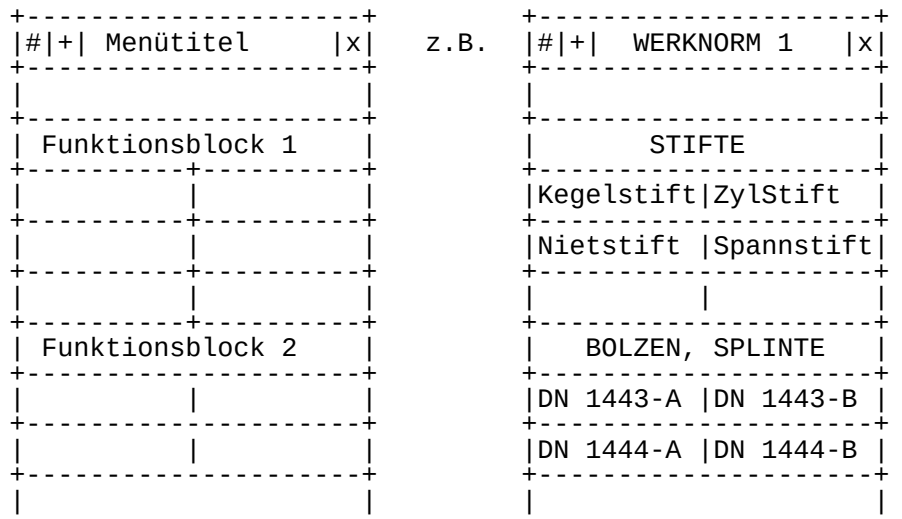
Tabellenüberschriften werden mit der Parameterliste übergeben. Kopf- und Datenfelder zeigen lediglich Daten an und sind nur mit einem Signalton belegt. Die Anzeigetabellen werden nach dem Einrichten und Belegen so verschoben, daß sich Tabelle "A_a" links und Tabelle "B_a" oberhalb von Tabelle "C_a" befindet. Danach wird eine vierte Anzeigetabelle "Ausschalter_a" zum Löschen aller vier Anzeigetabellen eingerichtet.

15.4.2 Hilfesystem

Mit den nachfolgend beschriebenen Makros wird eine Online-Hilfe für Bildschirmmenüs erzeugt.

Funktionsprinzip der Online-Hilfe

Das Funktionsprinzip wird anhand eines Beispiels erläutert, bei dem das Hilfesystem die Anwendung von Makros für Werksnormteile unterstützen soll. Die Makros sollen über zwei Bildschirmenüs "WERKNORM 1" und WERKNORM 2"bedient werden. Die Bildschirmmenüs sind wie folgt aufgebaut:



Das Menülayout und die Anzahl der Funktionsblöcke und Menüfelder unterliegen keinen Beschränkungen. Die Felder mit dem Menütitel und den Bezeichnungen der Funktionsblöcke werden mit Hilfesystem-Aufrufen belegt. Wenn also Hilfe zum Menü "WERKNORM 1" benötigt wird, ist das Titelfeld in der ersten Zeile des Menüs anzutippen. Wird Hilfe zum Funktionsblock "BOLZEN, SPLINTE" benötigt, so ist das Titelfeld dieses Funktionsblocks anzutippen. Da die Titelfelder mit Aufrufen für das Hilfesystem belegt werden, ist eine Belegung mit Funktionsaufrufen nicht zulässig.

Sämtliche Hilfetexte befinden sich in einer einzigen ASCII-Datei. Beim Laden des Hilfesystems (dies sollte zweckmäßigerweise zusammen mit dem Laden der Werknorm-Makros erfolgen) wird die Hilfedatei gelesen und in Logische Tabellen geschrieben. Die Hilfetext-Datei ist hierzu in Hilfetext-Abschnitte untergliedert. Ein Abschnitte gibt entweder einen Überblick über sämtliche Funktionen eines Menüs oder er beschreibt einen einzelnen Funktionsblock.

Jeder Abschnitt wird in eine eigene Logische Tabelle geschrieben. Damit ist jedem Titelfeld eine Logische Tabelle zugeordnet, die den zugehörigen Hilfetext enthält. Beim

Laden des Hilfesystems wird weiterhin eine Anzeigetabelle erzeugt. Sie wird zunächst nicht belegt oder eingeblendet. Erst beim Antippen eines Menü-Titelfeldes wird die dem Feld zugeordnete Logische Tabelle mit der Anzeigetabelle verbunden und die Anzeigetabelle wird eingeblendet. Der Hilfetext erscheint im Datenbereich der Tabelle.

Anforderungen an die Hilfetext-Datei

Um eine korrekte Zuordnung zwischen einem Titelfeld des Menüs, dem Textabschnitt der Hilfetext-Datei und der Logischen Tabelle zu gewährleisten, wurden folgende Regeln festgelegt:

Der Beginn eines Hilfetext-Abschnitts wird durch drei Zeilen gekennzeichnet, die im konkreten Fall wie folgt lauten können:

```
LABEL=Wb_menu_1_2
NROWS=40
TITLE=Online-Hilfe zum Menüpunkt WERKNORM 1 BOLZEN, SPLINTE
```

"Wb_menu_2" ist der Name des Bildschirmmenüs, der Zusatz "_2" bedeutet, daß der Hilfetext zum Funktionsblock Nr. 2, im Beispiel also zum Funktionsblock "BOLZEN, SPLINTE", gehört. Für das Titelfeld des Menüs selbst würde die Zeile "LABEL=Wb_menu_0 lauten.

"NROWS=40" gibt die maximale Anzahl der Hilfetextzeilen des Hilfetext-Abschnitts. Sie wird zum Anlegen der Logischen Tabelle benötigt und sollte lieber etwas größer als zu klein angegeben werden.

"TITLE=Online-Hilfe zum Menüpunkt WERKNORM 1 / BOLZEN, SPLINTE" bestimmt den Inhalt des Titelsfelds der Hilfesystem-Anzeigetabelle. Der auf das Gleichheitszeichen folgende Text darf maximal 74 Zeichen lang sein.

Die Namen der Logischen Tabellen, in den die Texte der Hilfetext-Abschnitte geschrieben werden, setzten sich aus der hinter "LABEL=" stehenden Zeichenfolge und dem Zusatz "_lt" zusammen. Im genannten Fall würde also Logische Tabelle "Wn_menu_1_2_lt" heißen.

Der Hilfetext beginnt unmittelbar nach der Zeile "TITLE=....". und endet unmittelbar vor der nächsten, mit "LABEL=...." beginnenden Zeile. Leerzeilen zwischen diesen beiden Begrenzern werden als Bestandteil des Hilfetextes angesehen und in der Anzeigetabelle dargestellt.

Der Hilfetext darf aus beliebig vielen Zeilen und aus bis zu 80 Zeichen pro Zeile bestehen. Längere Zeilen würden abgeschnitten. Das in der nachfolgenden Musterdatei am Ende einer Zeile auf Position 81 stehende Zeichen "|" ist lediglich als optische Begrenzung zum Einhalten der 80 Zeichen gedacht, es wird beim Erzeugen der Logischen Tabellen nicht verwendet und kann daher auch wegfallen.

Die Hilfetext-Datei sollte mit der Zeile "LABEL=END-OF-HELP-TEXT" enden.

Belegung der Menüfelder

Die Felder mit dem Menütitel und den Bezeichnungen der Funktionsblöcke sind wie folgt zu belegen:

```
'Online_help "LogTabName"'
```

Das Makro "Online_help" verknüpft die Anzeigetabelle "Online_help_dt" mit der als Parameter angegebenen Logischen Tabelle und blendet sie ein. Im vorgenannten Beispiel wäre also das Feld mit dem Überschrift "BOLZEN, SPLINTE" mit dem Aufruf 'Online_help "Wn_menu_1_2_lt"' zu belegen.

Die erste Titelzeile der Anzeigetabelle wird mit Piktogrammen zum Fixieren und Nichtfixieren, zum Bewegen und Ausblenden der Tabelle sowie mit einer Überschrift belegt. Die Felder der zweiten Titelzeile dienen dem Ausdrucken des sichtbaren Tabellenteils, dem Ausdrucken der gesamten Tabelle und dem Verkleinern bzw. Vergrößern der Tabelle. Weitere Hinweise zum Hilfesystem sind den Makrotexten zu entnehmen.

Hilfetext-Datei

```
LABEL=BEGIN-OF-HELP-TEXT
LABEL=Wn_menu_1_0
NROWS=40
TITLE=Online-Hilfe zum Menü WERKNORM 1
                                                                              |
Dieses Bildschirmmenü enthält Makroaufrufe zum Erzeugen von Werknormteilen.   |
                                                                              |
Menüpunkt "Stifte"                                                            |
==================                                                            |
                                                                              |
Mit Hilfe dieses Menüpunktes können Kegelstifte nach DIN 1, Zylinderstifte nach|
DIN 7, Nietstifte nach DIN 7341 und Spannstifte (Spannhülsen) nach DIN 1481   |
(schwere Aufführung) gezeichnet werden.                                       |
                                                                              |
Menüpunkt "BOLZEN, SPLINTE"                                                   |
===========================                                                   |
                                                                              |
Mit Hilfe dieses Menüpunktes können zylindrische Bolzen nach DIN 1433 Form A  |
und Form B, Bundbolzen nach DIN 1444 Form A und Form B sowie Splinte nach     |
DIN 94 gezeichnet werden.                                                     |
                                                                              |
Menüpunkt ..........                                                          |
====================                                                          |
                                                                              |
usw. usw.                                                                     |
                                                                              |

LABEL=Wn_menu_1_1
NROWS=80
TITLE=Online-Hilfe zum Menüpunkt WERKNORM 1 / STIFTE
                                                                              |
Mit Hilfe dieses Menüpunktes können Kegelstifte nach DIN 1 und Zylinderstifte |
nach DIN 7 gezeichnet werden.                                                 |
                                                                              |
Kegelstift                                                                    |
----------                                                                    |
Tippen Sie das Feld Kegelstift an. Es erscheint eine Anzeigetabelle, in der   |
Sie den Nenndurchmesser, die Länge und die Art der Ansicht (Seite = Seiten-   |
ansicht, Vorder = Kegelseite zum Betrachter, Rück = Kegelseite vom Betrachter |
weg) wählen können. Danach ist der Stift zu postionieren. Sie haben dazu      |
folgende Möglichkeiten:                                                       |
                                                                              |
usw. usw.                                                                     |
                                                                              |
```

```
Zylinderstift
-------------
Tippen Sie das Feld Zylinderstift an. Es erscheint eine Anzeigetabelle, ....


usw. usw.

LABEL=Wn_menu_1_2
NROWS=80
TITLE=ONLINE-Hilfe zum Menüpunkt WERKNORM 1 / BOLZEN, SPLINTE

Mit Hilfe dieses Menüpunktes können zylindrische Bolzen nach DIN 1433 Form A
und Form B, Bundbolzen nach DIN 1444 Form A und Form B sowie Splinte nach
DIN 94 gezeichnet werden.


DN 1443-A/B  (Form A : Ohne Splintloch, Form B : Mit Splintloch)

-----------
Tippen Sie das Feld DN 1143-A oder DN 1443-B an. Es erscheint eine Anzeige-
tabelle, ..............

usw. usw.

LABEL=END-OF-HELP-TEXT
```

Makros

```
DEFINE Start_demo
(* Online-Hilfe für Bildschirmmenüs  *)
(* Fischer, 18.09.96, Stand 22.11.96 *)

 Online_help_make_lt
 Online_help_dt_layout
 Wn_menu_layout
 Wn_menu_1

END_DEFINE


DEFINE Wn_menu_layout
(* Layout für Bildschirmmenü         *)
(* Fischer, 18.09.96, Stand 22.09.96 *)

 LOCAL N_rows  (* Zähler für Menüzeilen *)

 IF (I_port) Check_i_port) END_IF
 IF (NOT I_Port)
  CURRENT_MENU 'Wn_menu_1'
  MENU_LAYOUT
   Menu_position RIGHT
   Headline_height    ' | |              | '
   Text_slot_height   '                  '
   Text_slot_height   '                  '
   Text_slot_height   '          |       '
   Text_slot_height   '          |       '
   Text_slot_height   '          |       '
   Text_slot_height   '          |       '
   Text_slot_height   '          |       '
   Text_slot_height   '          |       '
   Text_slot_height   '          |       '
   LET N_rows 0
   REPEAT
    LET N_rows (N_rows + 1)
    Text_slot_height  '          |       '
   UNTIL (N_rows = 16)
   Bottom_slot_height '                  '
   Menu_home_point_top
  END
 END_IF
END_DEFINE


DEFINE Wn_menu_1
(* Bildschirmmenü WERKNORM 1         *)
(* Fischer, 18.09.96, Stand 19.09.96 *)
 IF (I_port) Check_i_port) END_IF
 IF (NOT I_port)
  CURRENT_MENU 'Wn_menu_1'
  Menu_control_icons
  MENU Colo0 Bcol5 CENTER 'WERKNORM 1' 'Online_help "Wn_menu_1_0_lt"'  1 3
  MENU Colo0 Bcol1 CENTER 'STIFTE' 'Online_help "Wn_menu_1_1_lt"' 3 1
  MENU 'Kegelstift' 'Empty'  4 1
  MENU 'ZylStift'   'Empty'  4 2
  MENU 'Nietstift'  'Empty'  5 1
  MENU 'Spannstift' 'Empty'  5 2
  MENU Colo0 Bcol1 CENTER 'BOLZEN, SPLINTE' 'Online_help "Wn_menu_1_2_lt"' 7 1
  MENU 'DN 1443-A' 'Empty'  8 1
  MENU 'DN 1443-B' 'Empty'  8 2
  MENU 'DN 1444-A' 'Empty'  9 1
  MENU 'DN 1444-B' 'Empty'  9 2
 END_IF
END_DEFINE


DEFINE Empty
(* Makro für leeres Menüfeld         *)
(* Fischer, 18.09.96, Stand 22.09.96 *)
 BEEP DISPLAY 'Menüfeld nicht belegt'
END_DEFINE


(* Globale Variablen *)
LET Helptext_path '.'             (* Verzeichnis der Hilfetext-Datei *)
LET Helptext_file 'wn_help.txt'   (* Datei mit den Hilfetexten       *)
```

```
DEFINE Online_help_make_lt
(* Anlegen von Log.Tabellen für das Hilfe-System *)
(* Fischer, 18.09.96, Stand 22.09.96             *)

 LOCAL Dataline    (* Eingelesene Textzeile          *)
 LOCAL Lt_name     (* Name der Log.Tabelle           *)
 LOCAL Lt_row      (* Aktuelle Zeile der Log.Tabelle *)
 LOCAL Menu_label  (* Menüfeldkennung                *)
 LOCAL Help_title  (* Titelzeile der Anzeigetabelle  *)
 LOCAL Rowsize     (* Zeilenzahl der Log.Tabelle     *)

 (* Öffnen der Hilfetext-Datei *)
 OPEN_INFILE 1 (helptext_path + '/' + helptext_file)

 DISPLAY_NO_WAIT ('Hilfesystem wird mit Texten aus Datei ' +
 helptext_path + '/' + helptext_file + ' eingerichtet.')

 (* Anfang der Hilfetexte suchen *)
 REPEAT
  READ_FILE 1 Dataline
  IF (Dataline = 'END-OF-FILE')
   BEEP
   DISPLAY ('Hilfetext-Datei ' + helptext_path + '/' + helptext_file +
   ' ist fehlerhaft, Hilfesystem wird nicht eingerichtet.')
   CANCEL
  END_IF
 UNTIL (Dataline = 'LABEL=BEGIN-OF-HELP-TEXT')

 (* Anfang des ersten Hilfetext-Abschnitts suchen *)
 REPEAT
  READ_FILE 1 Dataline
  IF (Dataline = 'END-OF-FILE')
   BEEP
   DISPLAY ('Hilfetext-Datei ' + helptext_path + '/' + helptext_file +
   ' ist fehlerhaft, Hilfesystem wird nicht eingerichtet.')
   CANCEL
  END_IF
 UNTIL (SUBSTR Dataline 1 5 = 'LABEL')

 (* Hilfetext-Abschnitte suchen und in Logische Tabellen schreiben *)
 LOOP

 EXIT_IF ((Dataline = 'LABEL=END-OF-HELP-TEXT') OR (Dataline = 'END-OF-FILE'))

  (* Tabelle anlegen und Überschrift eintragen *)
  LET Menu_label (SUBSTR Dataline 7 (LEN Dataline - 6))
  READ_FILE 1 Dataline
  IF (SUBSTR Dataline 1 5 <> 'NROWS')
   BEEP
   DISPLAY ('Hilfetext-Datei ' + helptext_path + '/' + helptext_file +
   ' ist fehlerhaft, Hilfesystem wird nicht eingerichtet.')
   CANCEL
  END_IF
  LET Rowsize (VAL SUBSTR Dataline 7 (LEN Dataline - 6))
  READ_FILE 1 Dataline
  IF (SUBSTR Dataline 1 5 <> 'TITLE')
   DISPLAY ('Hilfetext-Datei ' + helptext_path + '/' + helptext_file +
   ' ist fehlerhaft, Hilfesystem wird nicht eingerichtet.')
   CANCEL
  END_IF
  LET Help_title (SUBSTR Dataline 7 (LEN Dataline - 6))
  LET Lt_name (Menu_label + '_lt')
  CREATE_LTAB Rowsize 1 Lt_name
  WRITE_LTAB Lt_name TITLE 1 Help_title

  (* Hilfetext in Tabelle schreiben *)
  LET Lt_row 0
  READ_FILE 1 Dataline
  WHILE ((SUBSTR Dataline 1 5 <> 'LABEL') AND (Dataline <> 'END-OF-FILE'))
   LET Lt_row (Lt_row + 1)
   WRITE_LTAB Lt_name Lt_row 1 (TRIM SUBSTR Dataline 1 80)
   READ_FILE 1 Dataline
  END_WHILE

 END_LOOP

 (* Hilfetext-Datei schließen *)
 CLOSE_FILE 1

END_DEFINE
```

```
DEFINE Dt_standards
{Standardeinstellungen für Anzeigetabellen}
{Fischer, 24.09.96, Stand 24.09.96        }

INQ_ENV 10

 IF (INQ 6) {MEPELOOK=1}
  DEFINE Dt_border          RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_bg              RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_scrollbar_fg    RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_scrollbar_bg    RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_head_fg         RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_head_bg         RGB_COLOR 0.143 0.286 0.667 END_DEFINE
  DEFINE Dt_title_fg        RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_title_bg0       RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_title_bg1       RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_data_fg         RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_data_bg         RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_frame_with      5       END_DEFINE
  DEFINE Dt_horizontal_col WHITE  END_DEFINE
  DEFINE Dt_horizontal_typ SOLID  END_DEFINE
  DEFINE Dt_vertical_col   WHITE  END_DEFINE
  DEFINE Dt_vertical_typ   SOLID  END_DEFINE
 ELSE {MEPELOOK=0}
  DEFINE Dt_border          WHITE  END_DEFINE
  DEFINE Dt_bg              BLACK  END_DEFINE
  DEFINE Dt_scrollbar_fg    WHITE  END_DEFINE
  DEFINE Dt_scrollbar_bg    BLUE   END_DEFINE
  DEFINE Dt_head_fg         BLACK  END_DEFINE
  DEFINE Dt_head_bg         YELLOW END_DEFINE
  DEFINE Dt_title_fg        WHITE  END_DEFINE
  DEFINE Dt_title_bg0       BLACK  END_DEFINE
  DEFINE Dt_title_bg1       BLACK  END_DEFINE
  DEFINE Dt_data_fg         WHITE  END_DEFINE
  DEFINE Dt_data_bg         BLACK  END_DEFINE
  DEFINE Dt_frame_with      1       END_DEFINE
  DEFINE Dt_horizontal_col WHITE  END_DEFINE
  DEFINE Dt_horizontal_typ SOLID  END_DEFINE
  DEFINE Dt_vertical_col   WHITE  END_DEFINE
  DEFINE Dt_vertical_typ   SOLID  END_DEFINE
END_IF

END_DEFINE


{Erzeugen der Standardeinstellungen für Anzeigetabellen}
Dt_standards


DEFINE Online_help_dt_layout
 (* Anlegen einer Anzeigetabelle für die Online-Hilfe *)
 (* Fischer, 18.09.96, Stand 22.11.96                 *)
 (*
 +-------------------------------------------------------------------------------+
 |#|+| Online-Hilfe zum Menüpunkt .....                                      |x|
 +-------------------------------------------------------------------------------+
 |  Fensterinhalt drucken  |  Ganze Tabelle drucken   |  Tabelle klein/groß    |
 +-------------------------------------------------------------------------------+
 *)

 TABLE_LAYOUT 'Online_help_dt'
 Dt_border Dt_bg
 ROWS 25
 FRAME_WIDTH Dt_frame_width
 HORIZONTAL Dt_horizontal_col Dt_horizontal_typ
 VERTICAL Dt_vertical_col Dt_vertical_typ
 SCROLL_BAR Dt_scrollbar_fg Dt_scrollbar_bg
 TITLE_LAYOUT
  (Headline_height + 1) (' | |' + RPT ' ' 74  + '| ')
  (Headline_height + 1) (RPT ' ' 25 + '|' + RPT ' ' 26 + '|' + RPT ' ' 25)
                        2 (RPT ' ' 80)
 END
 COLUMN_LAYOUT
  (Text_slot_height + 1) (RPT ' ' 80)
 END

END_DEFINE
```

```
DEFINE Online_help
(* Anzeigen von Online-Hilfe zum Menüsystem *)
(* Fischer, 18.09.96, Stand 22.09.96        *)

 PARAMETER Lt_name  (* Name der Log.Tabelle *)

 CONNECT_TABLE 'Online_help_dt' Lt_name
 Table_control_icons 'Online_help_dt'
 TABLE_TITLE 'Online_help_dt'
  Dt_head_fg Dt_head_bg CENTER '@s1' 1 3
  Dt_title_fg Dt_title_bg1 CENTER 'Fensterinhalt drucken'
   'Online_help_print_window' 2 1
  Dt_title_fg Dt_title_bg1 CENTER 'Ganze Tabelle drucken'
   ('Online_help_print_all "' + Lt_name + '"') 2 2
  Dt_title_fg Dt_title_bg1 CENTER 'Tabelle klein/groß'
   'Online_help_dt_small_big' 2 3
  Dt_head_fg Dt_head_bg '' '' 3 1
 END

 TABLE_COLUMN 'Online_help_dt'
  COLUMN 1 Dt_data_fg Dt_data_bg LEFT 1 ''
 END
 SHOW_TABLE ON 'Online_help_dt'

END_DEFINE



DEFINE Online_help_print_window
(* Ausdrucken des sichtbaren Tabelleninhalts*)
(* Fischer, 18.09.96, Stand 22.09.96        *)

 LOCAL Printfile  (* Name der ausgedruckten Datei *)

  (* Ausgabe an Druckerspooler, wenn Betriebssystem HP-UX *)
  INQ_ENV 10
  IF (INQ 4 = 2)
   LET Printfile '| lp -onb -o12 -olm5'
  ELSE
   LET Printfile 'prt'
   DISPLAY 'Hilfetext wird in Datei "prt" geschrieben.'
  END_IF

 PRINT_TABLE 'Online_help_dt' DEL_OLD Printfile

END_DEFINE



DEFINE Online_help_print_all
(* Ausdrucken des ganzen Tabelleninhalts *)
(* Fischer, 18.09.96, Stand 22.09.96     *)

  PARAMETER Lt_name  (* Name des Log.Tabelle       *)
  LOCAL Rowsize      (* Zeilenzahl der Log.Tabelle *)
  LOCAL Lt_row       (* Zähler für die akt.Zeile   *)
  LOCAL Printfile    (* Temporäre Datei            *)

  LET Rowsize (LTAB_ROWS Lt_name)

  (* Ausgabe an Druckerspooler, wenn Betriebssystem HP-UX *)
  INQ_ENV 10
  IF (INQ 4 = 2)
   LET Printfile '| lp -onb -o12 -olm5'
  ELSE
   LET Printfile 'prt'
   DISPLAY 'Hilfetext wird in Datei "prt" geschrieben.'
  END_IF

  OPEN_OUTFILE 1 DEL_OLD Printfile
  LET Lt_row 1
  WHILE (Lt_row <= Rowsize)
   WRITE_FILE 1 (READ_LTAB Lt_name Lt_row 1)
   LET Lt_row (Lt_row + 1)
  END_WHILE
  CLOSE_FILE 1

END_DEFINE
```

```
DEFINE Online_help_dt_small_big
(* Vekleinern der Tabelle auf halbe Höhe *)
(* Vergrößern auf ursprüngliche Höhe     *)
(* Fischer, 22.09.96, Stand 22.09.96     *)

LOCAL H0  (* Stammhöhe *)
LOCAL H1  (* Akt.Höhe  *)

 INQ_TABLE 'Online_help_dt'
 LET H0 (Y_OF INQ 102 - Y_OF INQ 101)
 LET H1 (Y_OF INQ 104 - Y_OF INQ 103)
 CHANGE_TABLE_SIZE 'Online_help_dt'
 IF (H1 < H0)
  (INQ 103) (INQ 103 + (INQ 102 - INQ 101))
 ELSE
  (INQ 103) (INQ 103 + (PNT_XY (X_OF INQ 102 - X_OF INQ 101)  (0.5 * H0)))
 END_IF

END_DEFINE
```

A1 Befehle und Interruptfunktionen

Beim Schreiben von Makros müssen die englischsprachigen Namen von Befehlen, Interruptfunktionen und Optionen (sie werden in folgenden als ME10-Funktionen bezeichnet) bekannt sein. Anwender ohne Erfahrung im Programmieren von Makros kennen diese Namen meist nicht, da sie beim interaktiven Arbeiten nicht benötigt werden. Die Erfahrung zeigt allerdings, daß in Makros nur ein relativ geringer Teil der obengenannten ME10-Funktionen benötigt werden. Die meisten Funktionen sind nämlich auf die Erfordernisse des interaktiven Arbeitens zugeschnitten. Wenn der Anwender weiß, welche ME10-Funktionen er verwenden will, kann er ihre Namen wie folgt ermitteln:

Aufruf des Hilfe-Systems

Der Anwender tippt das mit "HILFE" beschriftete Feld des Menüsystems und danach das Menüfeld der gewünschten Funktion an. Er erhält den zugehörigen Eintrag der HELP-Datei am Bildschirm angezeigt.

Verwendung einer ECHO-Datei

Der Anwender öffnet eine ECHO-Datei (-> Kap.2.3.3), führt die gewünschten ME10-Funktionen interaktiv aus, schließt die Datei und schaut sich ihren Inhalt an.

Analyse von Aufrufmakros

Felder des Tablettmenüs (und der Bildschirm-Nebenmenüs, wenn ME10 mit einer Maus bedient wird) sind nicht direkt mit ME10-Funktionen, sondern mit Makros zum Aufruf dieser Funktionen belegt. Die Namen der Funktionen lassen sich daher durch Analyse der Aufrufmakros ermitteln. Der Benutzer muß dazu "EDIT_MACRO" an der Tastatur eingeben und danach das entsprechende Menüfeld antippen. In manchen Fällen rufen die Aufrufmakros allerdings selbst wieder Aufrufmakros auf, die in gleicher Weise zu analysieren sind.

Analyse von Menüdefinitionen

Felder des Bildschirmmenüs (des Bildschirm-Hauptmenüs, wenn ME10 mit einer Maus bedient wird) sind in den meisten Fällen direkt mit Befehlen, Interruptfunktionen oder Optionen belegt. Hier hilft eine Analyse der die Bildschirmmenüs definierenden Makros. Der Anwender ruft dazu das betreffende Menü auf, gibt "SAVE_MENU SCREEN" an der Tastatur ein und erhält die Menüdefinition am Bildschirm angezeigt. Sie enthält die Anzeige- und Belegungstexte aller Felder des betreffenden Menüs. Wird statt "SCREEN" ein Dateiname angegeben, erfolgt die Ausgabe in eine Datei. Wenn ein Bildschirmmenü auf diese Weise nicht aufgelistet werden kann, so handelt es sich nicht um ein Menü, sondern um eine Anzeigetabelle. (Beispiel: Die obere Hälfte des Bildschirmenüs PLOTTEN 2 enthält eine Anzeigetabelle.) Sie läßt sich wie folgt auflisten: "SAVE_TABLE" eingeben, Tabelle antippen, "SCREEN" oder ein anderes Ausgabeziel eingeben.

Die nachfolgenden Tabellen zeigen Anzeigetexte (rechte Tabelle) und Belegungstexte (linke Tabelle) der wichtigsten ME10-Bildschirmmenüs (bzw. der Bildschirm-Nebenmenüs, wenn ME10 mit einer Maus bedient wird).

ERSTELLEN 1

01	ERSTELLEN 1						
02							
03	PUNKT	NACHZEICHN		POINT		I_overdraw	
04	LINIE	2 Pkte		LINE TWO_PTS		LINE TWO_PTS	
05	Polygon	Waagerecht		LINE POLYGON		LINE HORIZONTAL	
06	Senkrecht	Parallel		LINE VERTICAL		LINE PARALLEL	
07	Lotrecht	TanBgn&Pkt		LINE PERPENDICULAR		LINE TAN_PT	
08	Tan 2 Bgn	Win & Läng		LINE TAN2		LINE PT_ANG_DIST	
09	Rechteck			LINE RECTANGLE			
10	KREIS	Mitt&Umf/R		CIRCLE CENTER		CIRCLE CENTER	
11	3 Pkte	Tan2&Pkt/R		CIRCLE THREE_PTS		CIRCLE TAN2_PT	
12	Konzentr	Durchmessr		CIRCLE PARALLEL		CIRCLE DIAMETER	
13	BOGEN	3 Pkte		ARC THREE_PTS		ARC THREE_PTS	
14	Mitt R Win	Mitt & End		ARC CEN_RAD_ANG		ARC CEN_BEG_END	
15	Konzentr	Pkt R Win		ARC PARALLEL		ARC PER_RAD_ANG	
16	Durchmessr	Kurvenzug		ARC DIAMETER		ARC SMOOTH	
17	VERSATZ			OFFSET			
18	MITTELLIN	MITLIN ÄND		CENTERLINE		CHG_CENLINE	
19	SYM LINIE	SYMLIN ÄND		SYMLINE		CHG_SYMLINE	
20	BEZUGLINIE			REFLINE		POINT MIX	
21	TRENNFKT	Ein/Aus		SPLITTING ON/OFF		SPLITTING ON/OFF	
22	TRENNEN	Elem/R-eck		SPLIT		SPLIT	
23		LängTeilen				POINT MIX	
24	VERSCHMELZ			MERGE			
25	MENÜ 2	SPLINE		Tm_create_2		Tm_create_3	
26							
27							

ERSTELLEN 2

01	ERSTELLEN 2						
02							
03	ÄQUIDIST			EQUIDISTANCE			
04	RUNDUNG			FILLET			
05	RUNDG ÄND			CHANGE_FILLET			
06	FASE	2 Pkte		CHAMFER TWO_PTS		CHAMFER TWO_PTS	
07	Abst Abst	AbstWinkel		CHAMFER DIST_DIST		CHAMFER DIST_ANG	
08	Eckpunkt			CHAMFER VERTEX			
09	ECKE	Mit Lösch		KEEP_CORNER OFF		KEEP_CORNER OFF	
10		Ohne Lösch				KEEP_CORNER ON	
11	KONVERT	Bgn&Linien		CONVERT_SPLINE		CONVERT_SPLINE	
12	Linien			CONVERT_SPLINE MAX_RADIUS 0			
13	SPL TEILEN			BSPL_PONIT_LENGTH			
14	POLYELEM	Erstellen		POLYELEM		POLYELEM	
15		Zerlegen				SMASH_POLY	
16	ELLIPSE	2Spkte&Pkt		ELL_VVP		ELL_VVP	
17	Mitt Win R	Mitt &Pkte		ELL_CAR		ELL_CPP	
18	2 Bren&Pkt	BrWinExAch		Ell_ffp		Ell_faem	
19							
20	HYPERBEL	MitBrenEnd		Hyp_cfp		Hyp_cfp	
21	MitTanBren			Hyp_ctf			
22							
23	PARABEL	MitBrenPkt		Par_cf		Par_cf	
24	BrRichtEnd			Par_fdp			
25	MENÜ 1	SPLINE		Tm_create_1		Tm_create_3	
26							
27							

SPLINE

01	SPLINE						
02							
03	INTPOL SPL			BSPLINE			
04	OPTIONEN	Grad			ORDER		
05	Offen	Geschloss		OPEN	CLOSED		
06	Anfügen	Tangente		APPEND	TANGENT		
07	STÜTZ SPL			BSPLINE CONTROL_POINTS			
08	OPTIONEN	Grad			ORDER		
09	Offen	Geschloss		OPEN	CLOSED		
10							
11	PKT ÄND			BSPL_MOVE_PNT			
12	IPKT ÄND			BSPL_MOVE_I_PNT			
13	OPTIONEN	Position			POSITION		
14		Tangente			TANGENT		
15	SPKT ÄND			BSPL_MOVE_C_PNT			
16							
17	IPKT LÖSCH	IPKT HINZU		BSPL_DEL_I_PNT	BSPL_ADD_I_PNT		
18	SPKT LÖSCH	SPKT HINZU		BSPL_DEL_C_PNT	BSPL_ADD_C_PNT		
19	TANG LÖSCH	TANG HINZU		BSPL_DEL_TANGENT	BSPL_MOVE_I_PNT TANGENT		
20							
21	STÜTZPOLYG	Zeigen Ein		SHOW_CPOLY ON	SHOW_CPOLY ON		
22		Zeigen Aus			SHOW_CPOLY OFF		
23	STUTZEN	Ein		TRIMMING ON	TRIMMING ON		
24		Aus			TRIMMING OFF		
25	MENÜ 1	MENÜ 2		Tm_create_1	Tm_create_2		
26							
27							

BEMASSEN 1

01	BEMASSEN 1						
02	Standard	Firma					
03	Filter	Ein / Aus	Filter_active_toggle Sm_show_filter		Filter_active_toggle Sm_show_filter		
04							
05	EINFACH	SYM EINFCH	DA_DIM_LINE		DA_DIM_LINE_SYM		
06	MASSL LANG	SYM LANG	DA_DIM_DATUM_LONG		DA_DIM_DATUM_LONG_SYM		
07	MASSL KURZ	KETTENMASS	DA_DIM_DATUM_SHORT		DA_DIM_CHAIN		
08	KOORDINATE	FASE	DA_DIM_COORD		DA_DIM_CHAMPFER		
09	RADIUS	DURCHMESSR	DA_DIM_RADIUS		DA_DIM_DIAMETER		
10	BOGEN	WINKEL	DA_DIM_ARC		DA_DIM_ANGLE		
11	GEO SENSIV	SCAN	DA_DIM_GEO_SENSE		DA_DIM_PD_SCAN		
12							
13	AUTO	Parallel	AUTO / MANUELL		PARALLEL		
14	Waagrecht	Sekrecht	HOIZONTAL		VERTICAL		
15	Lotrecht	Parall Zu	PERPENDICULAR		PARALLEL_TO_LINE		
16					DIM_DIAMETER NO_CENTER_LINE		
17	Präfix	Postfix Tolera	DIM_PREFIX	DIM_POSTFIX	DIM_TOLERANCE		
18	Suprfx	Subfix Normen	DIM_SUPERFIX	DIM_SUBFIX	Sm_help_style		
19							
20							
21							
22							
23							
24							
25	MENÜ 2	MENÜ 3	Sm_dam_menu_2		Sm_dam_menu_3		
26							
27							

BEMASSEN 2 / ÄNDERN

	BEMASSEN 2		[BEMASSEN 2 / ÄNDERN]	
01				
02	DB WERTE	FILTER		
03	ÄNDERN	POS AUTO		
04				
05	BEMTXT ÄND	Maß&LinBew	DA_MOVE_DIMENSION	DA_MOVE_DIMENSION
06	Maß Drehen	Maß Beweg	ROTATE_DIM_TEXT	DA_MOVE_DIMENSION TEXT_ONLY
07				
08	LINIEN ÄND	Unterbrech	MODIFY_DIM_LINES BREAK	MODIFY_DIM_LINES BREAK
09	Rücksetzen	Versetzen	MODIFY_DIM_LINES RESET	MODIFY_DIM_LINES STAGGERED
10				
11	BEM LÖSCH	Ohne Prüf	DA_DIM_DELETE NO_CHECK	DA_DIM_DELETE NO_CHECK
12		Mit Prüf		DA_DIM_DELETE CHECK
13				
14	BEM EINFÜG	Ohne Prüf	DA_DIM_INSERT NO_CHECK	DA_DIM_INSERT NO_CHECK
15		Mit Prüf		DA_DIM_INSERT CHECK
16				
17	ECKPKT ÄND	Maß Behalt	CHANGE_DIM VERTEX	CHANGE_DIM_VERTEX
18		Maß Beweg		CHANGE_DIM_VERTEX_MOVE
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 3	Sm_dam_menu_1	Sm_dam_menu_3
26				
27				

BEMASSEN 3 / LINIEN ÄND - PFEIL ÄND

					[BEMASSEN 3 / LINIEN ÄND - PFEIL ÄND]	
01	BEMASSEN 3					
02	LINIEN ÄND	TEXTE ÄND				
03	LINIEN SET	TEXTE SET				
04						
05	PFEIL ÄND	FARBE ÄND				
06	PKT ABST	LIN ABST				
07	STIFTBrÄND					
08						
09						
10	PFEIL ÄND	Beide			CHANGE_DIM_ARROW BOTH_ARROW	CHANGE_DIM_ARROW BOTH_ARROW
11	Erster	Zweiter			CHANGE_DIM_ARROW FIRST_ARROW	CHANGE_DIM_ARROW SEC_ARROW
12	OPTIONEN	Füllen Ein				FILL_ON
13	Füllen Aus	Größe Rel			FILL_OFF	RELATIVE
14	Größe Abs	Ohne Pfeil			ABSOLUTE	NONE
15	Mit Pfeil	Mit Punkt			ARROW_TYPE	DOT_TYPE
16	Mit Strich	JIS Spitze			SLASH_TYPE	JIS_TYPE
17	Innerhalb	Außerhalb			ARROW_INSIDE	ARROW_OUTSIDE
18	Auto				ARROW_AUTO	
19						
20						
21						
22						
23						
24						
25	MENÜ 1		MENÜ 2		Sm_dam_menu_1	Sm_dam_menu_2
26						
27						

BEMASSEN 3 / TEXTE ÄND - BEMTXT ÄND

	BEMASSEN 3		[BEMASSEN 3 / TEXTE ÄND - BEMTXT ÄND]	
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	BEMTXT ÄND	FORMAT ÄND		
06	EINHT ÄND	RAHMEN ÄND		
07	BEMPOS ÄND	RICHTG ÄND		
08	TOL ÄND	BEMFIX ÄND		
09	TEXT EDIT	EDIT ABBR		
10	BEMTXT ÄND	Alles	CHANGE_DIM_TEXTS DIM_ALL	CHANGE_DIM_TEXTS DIM_ALL
11	Unterer	Oberer	CHANGE_DIM_TEXTS SEC_ALL	CHANGE_DIM_TEXTS MAIN_ALL
12	Wert Unter	Wert Ober	CHANGE_DIM_TEXTS SEC_NUM	CHANGE_DIM_TEXTS MAIN_NUM
13	Tol Unter	Tol Ober	CHANGE_DIM_TEXTS SEC_TOL	CHANGE_DIM_TEXTS MAIN_TOL
14	Präfix	Postfix	CHANGE_DIM_TEXTS PREFIX	CHANGE_DIM_TEXTS POSTFIX
15	OPTIONEN	Größe Abs		ABSOLUTE
16	Schriftart	Größe Rel	FONT	RELATIVE
17	Neigung	Verh Br/Hö	SLANT	RATIO
18				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE ÄND - FORMAT ÄND

01	BEMASSEN 3			[BEMASSEN 3 / TEXTE ÄND - FORMAT ÄND]			
02	LINIEN ÄND	TEXTE ÄND					
03	LINIEN SET	TEXTE SET					
04							
05	BEMTXT ÄND	FORMAT ÄND					
06	EINHT ÄND	RAHMEN ÄND					
07	BEMPOS ÄND	RICHTG ÄND					
08	TOL ÄND	BEMFIX ÄND					
09	TEXT EDIT	EDIT ABBR					
10							
11	FORMAT ÄND	Bem Obere		CHANGE_DIM_FORMAT MAIN_ALL		CHANGE_DIM_FORMAT MAIN_ALL	
12		Bem Untere				CHANGE_DIM_FORMAT SEC_ALL	
13							
14	Haupt +	Haupt -		CHANGE_DIM_PLACES MAIN_NUM PLUS		CHANGE_DIM_PLACES MAIN_NUM MINUS	
15	Zweit +	Zweit -		CHANGE_DIM_PLACES SEC_NUM PLUS		CHANGE_DIM_PLACES SEC_NUM MINUS	
16							
17							
18							
19							
20							
21							
22							
23							
24							
25	MENÜ 1		MENÜ 2	Sm_dam_menu_1		Sm_dam_menu_2	
26							
27							

BEMASSEN 3 / TEXTE ÄND - EINHT ÄND

	BEMASSEN 3		[BEMASSEN 3 / TEXTE ÄND - EINHT ÄND]	
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	BEMTXT ÄND	FORMAT ÄND		
06	EINHT ÄND	RAHMEN ÄND		
07	BEMPOS ÄND	RICHTG ÄND		
08	TOL ÄND	BEMFIX ÄND		
09	TEXT EDIT	EDIT ABBR		
10	EINHT ÄND	Bem Obere	CONVERT_DIM_UNIT MAIN_DIM	CONVERT_DIM_UNIT MAIN_DIM
11		Bem Untere		CONVERT_DIM_UNIT SEC_DIM
12	OPTIONEN	mm	MM	MM
13	Inch	cm	INCH	CM
14	In/Bru	m	FRACT_INCH	MT
15	Ft-in/Bru	km	FT_FR_INCH	KM
16	Ft-in/BruZ	Grad	FT_FR_IN_SIGN	DEGREE
17	Ft-in/BruT	Radiant	FT_FR_IN_TEXT	RADIAN
18	Keine	Gd min sec	NONE	DEG_MIN_SEC
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE ÄND - RAHMEN ÄND

01	BEMASSEN 3			[BEMASSEN 3 / TEXTE ÄND - RAHMEN ÄND]			
02	LINIEN ÄND	TEXTE ÄND					
03	LINIEN SET	TEXTE SET					
04							
05	BEMTXT ÄND	FORMAT ÄND					
06	EINHT ÄND	RAHMEN ÄND					
07	BEMPOS ÄND	RICHTG ÄND					
08	TOL ÄND	BEMFIX ÄND					
09	TEXT EDIT	EDIT ABBR					
10							
11	RAHMEN ÄND	Aus		CHANGE_DIM_FRAME		CHANGE_DIM_FRAME OFF	
12	Ballon	Kasten		CHANGE_DIM_FRAME BALLOONED		CHANGE_DIM_FRAME BOXED	
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25	MENÜ 1		MENÜ 2	Sm_dam_menu_1		Sm_dam_menu_2	
26							
27							

BEMASSEN 3 / TEXTE ÄND - BEMPOS ÄND

	BEMASSEN 3		[BEMASSEN 3 / TEXTE ÄND - BEMPOS ÄND]	
01				
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	BEMTXT ÄND	FORMAT ÄND		
06	EINHT ÄND	RAHMEN ÄND		
07	BEMPOS ÄND	RICHTG ÄND		
08	TOL ÄND	BEMFIX ÄND		
09	TEXT EDIT	EDIT ABBR		
10				
11	BEMPOS ÄND	Auf Linie	CHANGE_DIM_TEXT_LOCATION	CHANGE_DIM_TEXT_LOCATION ON
12	Über Linie	Untr Linie	CHANGE_DIM_TEXT_LOCATION ABOVE	CHANGE_DIM_TEXT_LOCATION BELOW
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE ÄND - RICHTG ÄND

	BEMASSEN 3		[BEMASSEN 3 / TEXTE ÄND - RICHTG ÄND]	
01				
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	BEMTXT ÄND	FORMAT ÄND		
06	EINHT ÄND	RAHMEN ÄND		
07	BEMPOS ÄND	RICHTG ÄND		
08	TOL ÄND	BEMFIX ÄND		
09	TEXT EDIT	EDIT ABBR		
10				
11	RICHTG ÄND	Parallel	CHANGE_DIM_TEXT_ORIENTATION	CHANGE_DIM_TEXT_ORIENTATION PARALLEL
12	Waagrecht	Senkrecht	CHANGE_DIM_TEXT_ORIENTATION HORIZONTAL	CHANGE_DIM_TEXT_ORIENTATION VERTICAL
13	Lotrecht		CHANGE_DIM_TEXT_ORIENTATION PERPENDICULAR	
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE ÄND - TOL ÄND

	BEMASSEN 3		[BEMASSEN 3 / TEXTE ÄND - TOL ÄND]	
01				
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	BEMTXT ÄND	FORMAT ÄND		
06	EINHT ÄND	RAHMEN ÄND		
07	BEMPOS ÄND	RICHTG ÄND		
08	TOL ÄND	BEMFIX ÄND		
09	TEXT EDIT	EDIT ABBR		
10				
11	TOL ÄND	Editieren	EDIT_DIM_TOLERANCE	EDIT_DIM_TOLERANCE
12	Hinzufügen	Löschen	ADD_DIM_TOLERANCE	DELETE_DIM_TOLERANCE
13	Konvert		CONVERT_DIM_TOLERANCE	
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE ÄND - BEMFIX ÄND

	BEMASSEN 3		[BEMASSEN 3 / TEXTE ÄND - BEMFIX ÄND]	
01				
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	BEMTXT ÄND	FORMAT ÄND		
06	EINHT ÄND	RAHMEN ÄND		
07	BEMPOS ÄND	RICHTG ÄND		
08	TOL ÄND	BEMFIX ÄND		
09	TEXT EDIT	EDIT ABBR		
10				
11	PRÄFIX ÄND	Editieren	EDIT_DIM_PREFIX	EDIT_DIM_PREFIX
12	Hinzufügen	Löschen	ADD_DIM_PREFIX	DELETE_DIM_PREFIX
13	POSTFX ÄND	Editieren	EDIT_DIM_POSTFIX	EDIT_DIM_POSTFIX
14	Hinzufügen	Löschen	ADD_DIM_POSTFIX	DELETE_DIM_POSTFIX
15	SUPRFX ÄND	Editieren	EDIT_DIM_SUPERFIX	EDIT_DIM_SUPERFIX
16	Hinzufügen	Löschen	ADD_DIM_SUPERFIX	DELETE_DIM_SUPERFIX
17	SUBFIX ÄND	Editieren	EDIT_DIM_SUBFIX	EDIT_DIM_SUBFIX
18	Hinzufügen	Löschen	ADD_DIM_SUBFIX	DELETE_DIM_SUBFIX
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE ÄND - TEXT EDIT

01	BEMASSEN 3			[BEMASSEN 3 / TEXTE ÄND - TEXT EDIT]			
02	LINIEN ÄND	TEXTE ÄND					
03	LINIEN SET	TEXTE SET					
04							
05	BEMTXT ÄND	FORMAT ÄND					
06	EINHT ÄND	RAHMEN ÄND					
07	BEMPOS ÄND	RICHTG ÄND					
08	TOL ÄND	BEMFIX ÄND					
09	TEXT EDIT	EDIT ABBR					
10							
11	UNTERSTRCH	Ein / Aus		DIM_UNDERLINE_EDITED ON		DIM_UNDERLINE_EDITED ON / OFF	
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25	MENÜ 1		MENÜ 2	Sm_dam_menu_1		Sm_dam_menu_2	
26							
27							

BEMASSEN 3 / LINIEN SET - PFEIL SET

	BEMASSEN 3				[BEMASSEN 3 / LINIEN SET - PFEIL SET]			
01								
02	LINIEN ÄND	TEXTE ÄND						
03	LINIEN SET	TEXTE SET						
04								
05	PFEIL SET	FARBE SET						
06	PKT ABST	LIN ABST						
07	STIFTBrSET	MLABST SET						
08	MASSLSCHRT	MIN ABST						
09	LINIE FANG	UNTERBRECH						
10	PFEIL SET	Beide			SELECT_DIM_ARROW BOTH_ARROW		SELECT_DIM_ARROW BOTH_ARROW	
11	Erster	Zweiter			SELECT_DIM_ARROW FIRST_ARROW		SELECT_DIM_ARROW SEC_ARROW	
12	OPTIONEN	Füllen Ein					FILL_ON	
13	Füllen Aus	Größe Rel			FILL_OFF		RELATIVE	
14	Größe Abs	Ohne Pfeil			ABSOLUTE		NONE	
15	Mit Pfeil	Mit Punkt			ARROW_TYPE		DOT_TYPE	
16	Mit Strich	JIS Spitze			SLASH_TYPE		JIS_TYPE	
17	Innerhalb	Außerhalb			ARROW_INSIDE		ARROW_OUTSIDE	
18	Auto				ARROW_AUTO			
19								
20								
21								
22								
23								
24								
25	MENÜ 1		MENÜ 2		Sm_dam_menu_1		Sm_dam_menu_2	
26								
27								

BEMASSEN 3 / LINIEN SET - LINIE FANG

	BEMASSEN 3	
01		
02	LINIEN ÄND	TEXTE ÄND
03	LINIEN SET	TEXTE SET
04		
05	PFEIL SET	FARBE SET
06	PKT ABST	LIN ABST
07	STIFTBrSET	MLABST SET
08	MASSLSCHRT	MIN ABST
09	LINIE FANG	UNTERBRECH
10		
11	LINIE FANG	Ein / Aus
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25	MENÜ 1	MENÜ 2
26		
27		

[BEMASSEN 3 / LINIEN SET - LINIE FANG]	
DIM_CATCH_LINE ON	DIM_CATCH_LINE ON / OFF
Sm_dam_menu_1	Sm_dam_menu_2

BEMASSEN 3 / LINIEN SET - UNTERBRECH

	BEMASSEN 3		[BEMASSEN 3 / LINIEN SET - UNTERBRECH]	
01				
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	PFEIL SET	FARBE SET		
06	PKT ABST	LIN ABST		
07	STIFTBrSET	MLABST SET		
08	MASSLSCHRT	MIN ABST		
09	LINIE FANG	UNTERBRECH		
10				
11	UNTRBR EIN	UNTRBR AUS	DIM_BROKEN ON	DIM_BROKEN OFF
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE SET - BEMTXT SET

	BEMASSEN 3	
02	LINIEN ÄND	TEXTE ÄND
03	LINIEN SET	TEXTE SET
04		
05	BEMTXT SET	FORMAT SET
06	EINHT SET	RAHMEN SET
07	BEMPOS SET	RICHTG SET
08	TXABST SET	FANGEN SET
09		
10	BEMTXT SET	Alles
11	Unterer	Oberer
12	Wert Unter	Wert Ober
13	Tol Unter	Tol Ober
14	Präfix	Postfix
15	OPTIONEN	Größe Abs
16	Schriftart	Größe Rel
17	Neigung	Verh Br/Hö
18		
19		
20		
21		
22		
23		
24		
25	MENÜ 1	MENÜ 2
26		
27		

[BEMASSEN 3 / TEXTE SET - BEMTXT SET]	
CURRENT_DIM_TEXTS DIM_ALL	CURRENT_DIM_TEXTS DIM_ALL
CURRENT_DIM_TEXTS SEC_ALL	CURRENT_DIM_TEXTS MAIN_ALL
CURRENT_DIM_TEXTS SEC_NUM	CURRENT_DIM_TEXTS MAIN_NUM
CURRENT_DIM_TEXTS SEC_TOL	CURRENT_DIM_TEXTS MAIN_TOL
CURRENT_DIM_TEXTS PREFIX	CURRENT_DIM_TEXTS POSTFIX
	ABSOLUTE
FONT	RELATIVE
SLANT	RATIO
Sm_dam_menu_1	Sm_dam_menu_2

BEMASSEN 3 / TEXTE SET - FORMAT SET

	BEMASSEN 3		[BEMASSEN 3 / TEXTE SET - FORMAT SET]	
01				
02	LINIEN ÄND	TEXTE ÄND		
03	LINIEN SET	TEXTE SET		
04				
05	BEMTXT SET	FORMAT SET		
06	EINHT SET	RAHMEN SET		
07	BEMPOS SET	RICHTG SET		
08	TXABST SET	FANGEN SET		
09				
10	OPTIONEN	mm	MM	MM
11	Inch	cm	INCH	CM
12	In/Bru	m	FRACT_INCH	MT
13	Ft-in/Bru	km	FT_FR_INCH	KM
14	Ft-in/BruZ	Grad	FT_FR_IN_SIGN	DEGREE
15	Ft-in/BruT	Radiant	FT_FR_IN_TEXT	RADIAN
16	Keine	Gd min sec	NONE	DEG_MIN_SEC
17	Haupt +	Haupt -	DIM_DEC_PLACES PLUS END	DIM_DEC_PLACES MINUS END
18	Zweit +	Zweit -	DIM_2ND_PLACES PLUS END	DIM_2ND_PLACES MINUS END
19				
20				
21				
22				
23				
24				
25	MENÜ 1	MENÜ 2	Sm_dam_menu_1	Sm_dam_menu_2
26				
27				

BEMASSEN 3 / TEXTE SET - EINHT SET

	BEMASSEN 3	
01		
02	LINIEN ÄND	TEXTE ÄND
03	LINIEN SET	TEXTE SET
04		
05	BEMTXT SET	FORMAT SET
06	EINHT SET	RAHMEN SET
07	BEMPOS SET	RICHTG SET
08	TXABST SET	FANGEN SET
09		
10	EINHT SET	Bem Obere
11		Bem Untere
12	OPTIONEN	mm
13	Inch	cm
14	In/Bru	m
15	Ft-in/Bru	km
16	Ft-in/BruZ	Grad
17	Ft-in/BruT	Verh Br/Hö
18	Keine	Gd min sec
19		
20		
21		
22		
23		
24		
25	MENÜ 1	MENÜ 2
26		
27		

[BEMASSEN 3 / TEXTE SET - EINHT SET]	
CURRENT_DIM_UNIT MAIN_DIM	CURRENT_DIM_UNIT MAIN_DIM
	CURRENT_DIM_UNIT SEC_DIM
MM	MM
INCH	CM
FRACT_INCH	MT
FT_FR_INCH	KM
FT_FR_IN_SIGN	DEGREE
FT_FR_IN_TEXT	RADIAN
NONE	DEG_MIN_SEC
Sm_dam_menu_1	Sm_dam_menu_2

BEMASSEN 3 / TEXTE SET - RAHMEN SET

01	BEMASSEN 3			[BEMASSEN 3 / TEXTE SET - RAHMEN SET]			
02	LINIEN ÄND	TEXTE ÄND					
03	LINIEN SET	TEXTE SET					
04							
05	BEMTXT SET	FORMAT SET					
06	EINHT SET	RAHMEN SET					
07	BEMPOS SET	RICHTG SET					
08	TXABST SET	FANGEN SET					
09							
10	RAHMEN SET	Aus		DIM_FRAME OFF		DIM_FRAME OFF	
11	Ballon	Katen		DIM_FRAME BALLOONED		DIM_FRAME BOXED	
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25	MENÜ 1		MENÜ 2		Sm_dam_menu_1		Sm_dam_menu_2
26							
27							

BEMASSEN 3 / TEXTE SET - BEMPOS SET

01	BEMASSEN 3			[BEMASSEN 3 / TEXTE SET - BEMPOS SET]			
02	LINIEN ÄND	TEXTE ÄND					
03	LINIEN SET	TEXTE SET					
04							
05	BEMTXT SET	FORMAT SET					
06	EINHT SET	RAHMEN SET					
07	BEMPOS SET	RICHTG SET					
08	TXABST SET	FANGEN SET					
09							
10	BEMPOS SET	Auf Linie		DIM_TEXT_LOCATION ON		DIM_TEXT_LOCATION ON	
11	Über Linie	Untr Linie		DIM_TEXT_LOCATION ABOVE		DIM_TEXT_LOCATION BELOW	
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25	MENÜ 1		MENÜ 2	Sm_dam_menu_1		Sm_dam_menu_2	
26							
27							

BEMASSEN 3 / TEXTE SET - RICHTG SET

01	BEMASSEN 3			[BEMASSEN 3 / TEXTE SET - RICHTG SET]			
02	LINIEN ÄND	TEXTE ÄND					
03	LINIEN SET	TEXTE SET					
04							
05	BEMTXT SET	FORMAT SET					
06	EINHT SET	RAHMEN SET					
07	BEMPOS SET	RICHTG SET					
08	TXABST SET	FANGEN SET					
09							
10	RICHTG SET	Parallel		DIM_TEXT_ORIENTATION PARALLEL		DIM_TEXT_ORIENTATION PARALLEL	
11	Waagrecht	Senkrecht		DIM_TEXT_ORIENTATION HORIZONTAL		DIM_TEXT_ORIENTATION VERTICAL	
12	Lotrecht			DIM_TEXT_ORIENTATION PERPENDICUAR			
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25	MENÜ 1		MENÜ 2	Sm_dam_menu_1		Sm_dam_menu_2	
26							
27							

SCHRAFFUR

01	SCHRAFFUR						
02							
03	LÖSCHEN			DELETE_HATCH			
04							
05	START	Auto		HATCH AUTO		HATCH AUTO	
06		Manuell				HATCH MANUAL	
07							
08	SETZEN	Winkel		HATCH_ANGLE		HATCH_ANGLE	
09	Abstand	Bezugspkt		HATCH_DIST		HATCH_REF_PT	
10							
11	MUSTER SET	Neu		CURRENT_HATCH_PATTERN		CURRENT_HATCH_PATTERN	
12		Abbruch				CURRENT_HATCH_PATTERN NONE	
13							
14	ÄNDERN	Winkel		CHANGE_HATCH_ANGLE		CHANGE_HATCH_ANGLE	
15	Abstand	Bezugspkt		CHANGE_HATCH_DIST		CHANGE_HATCH_REF_PT	
16	Muster	Farbe		CHANGE_HATCH_PATTERN		CHANGE_HATCH_COLOR	
17	Linienart			CHANGE_HATCH_LINETYPE			
18	BEM FREISP			DIM_TEXT_HOLE			
19							
20	STANDARDS	Eisen		I_hatch_iron		I_hatch_iron	
21	Stahl	Kupfer		I_hatch_steel		I_hatch_copper	
22							
23							
24							
25							
26							
27							

TEXT 1

01	TEXT 1						
02							
03	EINGEBEN	Zeile		TEXT		TEXT	
04		Bildschirm				TEXT SCREEN	
05	EDITIEREN	KONV EDIT		EDIT_TEXT		EDIT_TEXT CONVERT	
06	SETZEN	Winkel		TEXT_ANGLE		TEXT_ANGLE	
07	Zeilenabst	Größe		TEXT_LINESPACE		TEXT_SIZE	
08	Verh Br/Hö	Neigung		TEXT_RATIO		TEXT_SLANT	
09	Ausrichten	Füllen Ein		TEXT_ADJUST		TEXT_FILL ON	
10		Füllen Aus				TEXT_FILL OFF	
11	ÄNDERN	Lösch Alt		MODIFY TEXTS		MODIFY TEXTS	
12		Ohne Lösch				MODIFY COPY TEXTS	
13	DREHEN	BEWEGEN		ROTATE		MOVE	
14							
15	HINWEISLIN	Start		LEADER_LINE		LEADER_LINE	
16	Unterbrech	Abschluß		BREAK		END	
17	BEGRENZUNG	Größe		LEADER_ARROW		LEADER_ARROW	
18	Dreieck	Pfeil		LEADER_ARROW TRIANGLE_TYPE		LEADER_ARROW ARROW_TYPE	
19	Punkt	Strich (/)		LEADER_ARROW DOT_TYPE		LEADER_ARROW SLASH_TYPE	
20	JIS	Keine		LEADER_ARROW JIS_TYPE		LEADER_ARROW NONE	
21	SCHRIFTART	ISO 3098		I_set_font		I_set_font "hp_i3098_v"	
22	Block	DIN 17		I_set_font "hp_block_v"		I_set_font "hp_d17_v"	
23	RAHMEN SET	Aus		TEXT_FRAME OFF		TEXT_FRAME OFF	
24	Ballon	Kasten		TEXT_FRAME BALLOON		TEXT_FRAME BOX	
25		MENÜ 2				Tm_text_2	
26							
27							

TEXT 2

01	TEXT 2						
02							
03	ÄNDERN	Winkel		CHANGE_TEXT_ANGLE		CHANGE_TEXT_ANGLE	
04	Zeilenabst	Größe		CHANGE_TEXT_LINESPACE		CHANGE_TEXT_SIZE	
05	Verh Br/Hö	Neigung		CHANGE_TEXT_RATIO		CHANGE_TEXT_SLANT	
06	Füllen Ein	Füllen Aus		CHANGE_TEXT_FILL ON		CHANGE_TEXT_FILL OFF	
07	Ausrichten			CHANGE_TEXT_ADJUST			
08	RAHMEN ÄND	Aus		CHANGE_TEXT_FRAME OFF		CHANGE_TEXT_FRAME OFF	
09	Ballon	Kasten		CHANGE_TEXT_FRAME BALLOON		CHANGE_TEXT_FRAME BOX	
10	SCHFT LST			LIST_FONTS			
11	OPTIONEN	Bildschirm		SCREEN		SCREEN	
12	Lösch Alt	Drucker		DEL_OLD		PRT	
13	Anfügen			APPEND			
14	SCHFT ÄND	ISO 3098		CHANGE_TEXT_FONTNAME		CHANGE_TEXT_FONTNAME "hp_i3098_v"	
15	Block	DIN 17		CHANGE_TEXT_FONTNAME "hp_block_v"		CHANGE_TEXT_FONTNAME "hp_d17_v"	
16							
17	SCHRFTEDIT			Font_editor			
18							
19							
20							
21							
22							
23							
24							
25		MENÜ 1				Tm_text_1	
26							
27							

DATEI 1

01	DATEI 1						
02							
03	LADEN	Zchn/Teil		LOAD		LOAD	
04	'ArbDatei'	Einzelteil		"workfile"		LOAD SUBPART	
05	ABSOLUT			ABSOLUTE			
06	SPEICHERN	Alles		STORE ALL		STORE ALL	
07	'ArbDatei'	Teil		DEL_OLD "workfile"		STORE	
08	SPEICH MI	Alles		STORE MI ALL		STORE MI ALL	
09	'ArbDatei'	Teil		DEL_OLD "workfile"		STORE MI	
10							
11	DATEILISTE	Mit Lösch		Fbt_dtabs_all_on 0		Fbt_dtabs_all_on 0	
12		Ohne Lösch				Fbt_dtabs_all_on 1	
13	VERZ AKT			CURRENT_DIRECTORY			
14	KOPIEREN			COPY_FILE			
15	LÖSCHEN			PURGE_FILE			
16	DATEI EDIT			EDIT_FILE			
17	EINGABE			INPUT			
18	SICHERN			SAVE			
19							
20	OPTIONEN	Lösch Alt		DEL_OLD		DEL_OLD	
21	Drucker	Anfügen		PRT		APPEND	
22	Rekursiv			RECURSIV			
23							
24							
25	Datei 2	ASSDOK	ÄNDSTD	Tm_file_2	Tm_file_3		Tm_file_4
26							
27							

DATEI 2

01	DATEI 2			
02				
03	KATALOG	Aktuell		
04		Name		
05	LIST ALLES			
06				
07	AUSWÄHLEN			
08	AUFW SORT	ABW SORT		
09	∩PHYS_NAM∩	∩DAT_NAM∩		
10	∩DAT_TYP∩	∩DAT_BESC∩		
11	∩DAT_GRÖß∩			
12	∩ERST_DAT∩	∩DAT_ÄND∩		
13	∩ZUGR_DAT∩	∩ZUGR_NUM∩		
14				
15	OPTIONEN	Bildschirm		
16	Lösch Alt	Drucker		
17	Anfügen			
18				
19	NEUERUNGEN	EXTRAS		
20				
21				
22				
23				
24				
25	Datei 1	ASSDOK	ÄNDSTD	
26				
27				

CATALOG " "	CATALOG " "		
	CATALOG		
DETAIL INFO			
SELECT			
SORT	RESERVE_SORT		
"PHYS_NAME"	"FILE_NAME"		
"FILE_SIZE"	"FILE_DESC"		
"FILE_SIZE"			
"CREATE_DATE"	"MODIFY_DATE"		
"ACCESS_DATE"	"NUM_ACCESS"		
SCREEN	SCREEN		
DEL_OLD	Prt		
APPEND			
New_in_7_macro	Goodies		
Tm_file_1	Tm_file_3	Tm_file_4	

ASSOZ DOKUMENT

01	ASSOZ DOKUMENT					
02						
03	LAD & AKT	AllAnsicht	Adu_load_update		Adu_load_update	
04		EinzelAns			Adu_load_update_single_view	
05	FARBE SET	Aktualis2D			Adu_upd_color_and_menu "Upd"	
06	Übertr 2D	Rgener 2D	Adu_upd_color_and_menu "Trans"		Adu_upd_color_and_menu "Regen"	
07	Äqu 3D-Geo	Neu 3D-Geo	Adu_upd_color_and_menu "Ident"		Adu_upd_color_and_menu "New"	
08						
09	RÜCKS ZEIG	DIFF ZEIG	SHOW GLOBAL ALL ON		Adu_show_diff	
10						
11	AKZEPTIERE	ÄNDERN	ADU_CONFIRM_ANNOS		ADU_UPDATE_ANNOS	
12						
13	BEM BEWEGE	BEM RCKSET	Adu_move_dim		Adu_reset_dim	
14						
15	LAD & AKT	Automatik	Adu_ask_user_for_update		Adu_ask_user_for_update	
16						
17	SPEICH AKT	workfile	Adu_store_current_drawing Adu_update_store_current		Adu_store_current_drawing Adu_update_store_current	
18						
19						
20						
21						
22						
23						
24						
25	Datei 1	Datei2	ÄNDSTD	Tm_file_1	Tm_file_2	Tm_file_4
26						
27						

ÄNDSTKONTROLLE

01	ÄNDSTKONTROLLE				
02					
03	DATEIN LAD		Rc_load		
04					
05	VERGLEICH	Geometrie	Rc_comp SOFT	Rc_comp SOFT	
06	+ Doku	+ Teile	Rc_comp MEDIUM	Rc_comp HARD	
07					
08	ANZEIGEN	Zchg 1	Rc_view "TopX"	Rc_view "TopX"	
09	Beide	Zchg 2	Rc_view "Top"	Rc_view "TopY"	
10					
11	RÜCKS ZEIG	DIFF ZEIG	SHOW GLOBAL ALL ON	Rc_show_diff	
12					
13	FORTSETZEN	Mit Zchg 1		Rc_clean_up "TOPY" "TOPX"	
14		Mit Zchg 2		Rc_clean_up "TOPX" "TOPY"	
15					
16	FARBE SET	Diff in 1		Rc_upd_color_and_menu "Only1"	
17	Identisch	Diff in 2	Rc_upd_color_and_menu "Ident"	Rc_upd_color_and_menu "Only2"	
18					
19					
20					
21					
22					
23					
24					
25	Datei 1	Datei2	ASSDOK	Tm_file_1	Tm_file_2	Tm_file_3
26							
27							

PLOT

01	PLOT						
02							
03	POSITION	Mitte		Set_sys_plot_center ON Sm_plot		Set_sys_plot_center ON Sm_plot	
04		Links				Set_sys_plot_center OFF Sm_plot	
05	MASSTAB	1		Plot_enter_plotscale_and_update		Plot_enter_plotscale_and_update	
06		Einpassen				Set_sys_plot_plotscale 0	
07	AUSGABE	Alles		Set_sys_plot_source ALL Sm_plot		Set_sys_plot_source ALL Sm_plot	
08	Kasten	DarstFenst		Plot_enter_two_pts_and_update		Set_sys_plot_source CURRENT WINDOW Sm_plot	
09	INHALT	Zeichnung		Set_sys_plot_as_displayed 0 Sm_plot		Set_sys_plot_as_displayed 0 Sm_plot	
10		Zeigen Akt				Set_sys_plot_as_displayed 1 Sm_plot	
11	DREH WINKL	0		Plot_enter_rotate_and_update		Plot_enter_rotate_and_update	
12	LINARTLÄNG			PLOT_LINETYPE_LENGTH			
13	PAPIER FMT	Ben Format		Plot_enter_user_format_and_update		Plot_enter_user_format_and_update	
14	A	A4		Set_sys_plot_format "A" Sm_plot		Set_sys_plot_format "A4" Sm_plot	
15	B	A3		Set_sys_plot_format "B" Sm_plot		Set_sys_plot_format "A3" Sm_plot	
16	C	A2		Set_sys_plot_format "C" Sm_plot		Set_sys_plot_format "A2" Sm_plot	
17	D	A1		Set_sys_plot_format "D" Sm_plot		Set_sys_plot_format "A1" Sm_plot	
18	E	A0		Set_sys_plot_format "E" Sm_plot		Set_sys_plot_format "A0" Sm_plot	
19	START PLOT	nLaserplot		Plot_do		Plot_do	
20	BILDS DUMP	Bildschirm		Dump_screen_screen		Dump_screen_screen	
21	Kasten	Fenster		Dump_screen_box		Dump_screen_port	
22	DUMP OPTN						
23	Farbe	Sch&Weiß		Set_sys_dump_screen_color ON Sm_plot		Set_sys_dump_screen_color OFF	
24	MitHinterg	OhnHinterg		Set_sys_dump_screen_background ON Sm_plot		Set_sys_dump_screen_background OFF Sm_plot	
25		KONFIG				Sm_plot_configuration	
26							
27							

ÄNDERN1

01	ÄNDERN 1						
02							
03	ÄNDERN	Lösch Alt		MODIFY DEL_OLD		MODIFY DEL_OLD	
04		Ohne Lösch				MODIFY COPY	
05							
06	BEWEGEN	2 Pkte		MOVE		MOVE TWO_PTS	
07	Waagerecht	Senkrecht		MOVE HORIZONTAL		MOVE VERTICAL	
08	Mehrfach			MOVE MULTIPLE			
09	DREHEN	Mitte		ROTATE		ROTATE CENTER	
10		2 Pkte				ROTATE TWO_PTS	
11	SPIEGELN	2 Pkte		MIRROR		MIRROR TWO_PTS	
12	Waagerecht	Senkrecht		MIRROR HORIZONTAL		MIRROR VERTICAL	
13	Mitte			MIRROR CENTER			
14	MASSTAB	Mitte		SCALE		SCALE CENTER	
15		2 Pkte				SCALE TWO_PTS	
16	ÄHNLICH			SIMILAR			
17							
18	FORMÄNDERN			AFFINE			
19							
20	DEHNEN	Lösch Alt		STRETCH DEL_OLD		STRETCH DEL_OLD	
21		Ohne Lösch				STRETCH COPY	
22	Waagerecht	Senkrecht		MOVE HORIZONTAL		MOVE VERTICAL	
23							
24	ISOMETRIE			ISOMETRIC			
25		MENÜ 2				Tm_modify_2	
26							
27							

ÄNDERN 2

01	ÄNDERN 2						
02							
03	BEREINIGEN	Stutz zwei		TRIM_TWO		TRIM_TWO	
04	Stutz eins	Mitte raus		TRIM_ONE		CUT_MIDDLE	
05	OPTIONEN	Folge				CHAIN	
06	KONTUR					CONTOUR	
07							
08	ENTFERN	Punkte		CLEAN_DRAWING CLEAN_CLOSE_POINTS		CLEAN_DRAWING CLEAN_CLOSE_POINTS	
09	Doppelt	Überlagert		CLEAN_DRAWING CLEAN_DUPLICATE_GEOMETRY		CLEAN_DRAWING CLEAN_STACKED_GEOMETRY	
10							
11	BEMZEIGEN	BEM VERDCK		PD_MAKE_DIMENSIONS		PD_HIDE_DIMENSIONS	
12							
13	BEM ÄNDERN	Sofort		PD_MODIFY_DIMENSION IMMEDIATE		PD_MODIFY_DIMENSION IMMEDIATE	
14		Später				PD_MODIFY_DIMENSION DEFER	
15							
16	BEM TAUSCH			PD_USE_DIMENSION			
17	BEM BEWEGE			MOVE_DIMENSION			
18							
19							
20							
21							
22							
23							
24							
25	MENÜ 1			Tm_modify_1			
26							
27							

TEILE 1

01	TEILE 1					
02						
03	EDITIEREN	Teil		EDIT_PART		EDIT_PART
04	Top	Oberbaugrp		EDIT_PART TOP		EDIT_PART PARENT
05	BEGINNEN	Teil		INIT_PART		INIT_PART
06		Einzelteil				INIT_SUBPART
07	ENDE			END_PART		
08	ERSTELLEN	Einzelteil		CREATE_SUBPART		CREATE_SUBPART
09		Detail				CREATE_DETAIL
10	ELEM BIND	Lösch Alt		GATHER		GATHER
11		Kopieren				GATHER COPY
12	TEIL BIND	Lösch Alt		GATHER PART		GATHER PART
13		Kopieren				GATHER COPY PART
14	ZEIGEN	Teil		SHOW_PART		SHOW_PART
15	Alles	Teil []		SHOW_PART ALL		SHOW_PART PART_BOX
16	ET INTEGR			SMASH_SUBPART		
17	OPTIONEN	DatenBehal				KEEP_DATA
18		AnzeigBeha				KEEP_DISPLAY
19	MEHRF ABB	EIGENST TL		SHARE_PART		UNSHARE_PART
20				PARTS_LIST		PARTS_LIST
21	TL EDITOR			PRT_EDITOR		
22	OPTIONEN	Drucker		PRT		PRT
23	Lösch Alt	Anfügen		DEL_OLD		APPEND
24						
25		MENÜ 2				Tm_parts_2
26						
27						

TEILE 2

01	TEILE 2						
02							
03	ANZEIGEN	Teil		VIEW		VIEW	
04	Top	Oberbaugrp		VIEW TOP		VIEW PARENT	
05	HERVORHEBG	Ein		SPOTLIGHT ON		SPOTLIGHT ON	
06		Aus				SPOTLIGHT OFF	
07	SYMBOL	Ein		SYMBOL_PART		SYMBOL_PART	
08		Aus				SYMBOL_PART OFF	
09	BZPKT SET			CHANGE_PART_REF_PT			
10							
11	UMBENENNEN			RENAME_PART			
12							
13	SKAL TEIL	Teil+ETeil		PRT_DRW_SCALE		PRT_DRW_SCALE	
14		Nur Teil				PRT_DRW_SCALE ONLY	
15							
16	SKAL BZPKT	Mitte		PRT_DRW_SCALE_REF CENTER Sm_update_scale		PRT_DRW_SCALE_REF CENTER	
17	Ursprung	Bezugspunk		PRT_DRW_SCALE PART_ORIGIN		PRT_DRW_SCALE_REF REF_PNT	
18							
19	TEILELISTE	Aufzählen		PARTS_LIST		PARTS_LIST	
20		Baum				PARTS_LIST TREE	
21	OPTIONEN	Bildschirm		SCREEN		SCREEN	
22	Lösch Alt	Drucker		DEL_OLD		PRT	
23	Anfügen			APPEND			
24							
25	MENÜ 1			Tm_parts_1			
26							
27							

STANDARDS

01	STANDARDS						
02	KONFIG EDI	KONFIG SP		EDIT_ENVIRONMENT		SAVE_ENVIRONMENT	
03							
04	EINHEITEN	Inch		UNITS INCHES		UNITS INCHES	
05	Mikro-In	Fuß		UNITS UINCHES		UNITS FEET	
06	Yards	Meilen		UNITS YARDS		UNITS MILES	
07	mil	mm		UNITS MILS		UNITS MM	
08	cm	m		UNITS CM		UNITS METERS	
09	Mikrometer	km		UNITS UM		UNITS KM	
10	Grad	Neugrad		UNITS DEG		UNITS GRD	
11	Radiant	Benutzer-		UNITS RAD		User_units_def	
12							
13	RAHMEN LAD						
14	A	A4		LOAD SUBPART "format_A"		LOAD SUBPART "format_A4"	
15	B	A3		LOAD SUBPART "format_B"		LOAD SUBPART "format_A3"	
16	C	A2		LOAD SUBPART "format_C"		LOAD SUBPART "format_A2"	
17	D	A1		LOAD SUBPART "format_D"		LOAD SUBPART "format_A1"	
18	E	A0		LOAD SUBPART "format_E"		LOAD SUBPART "format_A0"	
19							
20	ZCHN MASST	1:1		Sm_get_drawing_scale		Sm_get_drawing_scale	
21							
22	PARAMETRIC	Ein		Enable_parametric_design		Disable_parametric_design / Enable...	
23	COPILOT	Ein/Aus		EnableCopilot		DisableCopilot / Enable...	
24	Fangabstnd	1		Sm_dist_grid_def		Sm_dist_grid_def	
25	Fangwinkel	15		Sm_angle_grid_def		Sm_angle_grid_def	
26							
27							

SYMBOLE 1

01	SYMBOLE 1						
02	ART	Normal					
03		Zusamgeszt					
04	FORM	Gerade					
05	Eben	Rund					
06	Zylinder						
07	PROFIL	Zeile					
08		Fläch Form					
09	ORT	Position					
10	Symmetrie	Konzentr					
11	RICHTUNG	Neigung					
12	Lotrecht	Parallel					
13	LAUFTOL	Rund/Plan					
14		Summenlauf					
5	ZS SYMBOLE						
16							
17	TEXTANFÜG	Zeile Neu					
18	Rückschrit	∩|∩					
19	Rahmen End						
20							
21	BEZUGSELEM	Symbol					
22		Punkt					
23	ÄNDRGSTAND	Dreieck					
24	KONV EDIT						
25		MENÜ 2					
26							
27							

		Symbols_normal Sm_update_type	
		Symbols_composite Sm_update_type	
Symbol_init (CHR 173)		Symbol_init (CHR 173)	
Symbol_init (CHR 174)		Symbol_init (CHR 175)	
Symbol_init (CHR 176)			
Symbol_init (CHR 177)		Symbol_init (CHR 177)	
		Symbol_init (CHR 178)	
Symbol_init (CHR 182)		Symbol_init (CHR 182)	
Symbol_init (CHR 172)		Symbol_init (CHR 183)	
Symbol_init (CHR 179)		Symbol_init (CHR 179)	
Symbol_init (CHR 180)		Symbol_init (CHR 181)	
Symbol_init (CHR 184)		Symbol_init (CHR 184)	
		Symbol_init (CHR 185)	
Modifiers_pop_up			
Symbol_add_text		Symbol_add_newline	
Symbol_back_space		Symbol_add_string "|"	
Symbol_end_frame			
Symbol_init (CHR 194)		Symbol_init (CHR 194)	
		Symbol_init (CHR 195)	
Symbol_init (CHR 199)		Symbol_init (CHR 199)	
EDIT_TEXT CONVERT		Symbols_point	
		Tm_symbols_2	

INFO

01	INFO						
02							
03	WAHL HINZU	Element		ADD_ELEM_INFO		ADD_ELEM_INFO	
04		Bildschirm				SCREEN	
05	AKT HINZU			ADD_CURRENT_INFO			
06							
07	EDITIEREN	Element		EDIT_ELEM_INFO		EDIT_ELEM_INFO	
08		Aktuell				EDIT_CURRENT_INFO	
09	LÖSCHEN	Element		DELETE_ELEM_INFO		DELETE_ELEM_INFO	
10		Aktuell				DELETE_CURRENT_INFO	
11	ÄNDERN	Element		CHANGE_ELEM_INFO		CHANGE_ELEM_INFO	
12	Global	Aktuell		CHANGE_GLOBAL_INFO		CHANGE_CURRENT_INFO	
13	AUFLISTEN			LIST_GLOBAL_INFO			
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							

VERDECKTE LINIEN 1

01	2D VERDECKTE LINIEN 1						
02							
03	POSINTEIL	Elem&Fläch		HL_GEN_ALL_PART ALL		HL_GEN_ALL_PART ALL	
04		Mit Löcher				HL_GEN_ALL_PART HOLES	
05							
06	FLÄCH ERST	EinfchAuto		HL_GENERATE_FACE AUTO		HL_GENERATE_FACE AUTO	
07	Alles Teil	EinfchManl		HL_GENERATE_FACE ALL		HL_GENERATE_FACE MANUAL	
08	Mit Löcher			HL_GENERATE_FACE HOLES			
09							
10	Z-WERT SET	Elemente		HL_SET_Z_VALUE ELEMENT		HL_SET_Z_VALUE ELEMENT	
11		Mehrf Teil				HL_SET_Z_VALUE SHARED_PART	
12							
13	Z-WERT AKT			HL_SET_CURR_Z_VALUE		HL_SET_CURR_Z_VALUE	
14	Z-WERT ABF	Absolut		HL_INQ_Z_VALUE		HL_INQ_Z_VALUE	
15	Minimum	Maximum		HL_INQ_Z_VALUE MINIMUM		HL_INQ_Z_VALUE MAXIMUM	
16	MTeil Abt	MTeilElem		HL_INQ_Z_VALUE SHARED_PART		HL_INQ_Z_VALUE RELATIVE	
17	z-Wert Akt	z-Wert Lad		HL_INQ_CURR_Z_VALUE		HL_INQ_LOAD_VALUE	
18	Z-WERT LAD			HL_SET_LOAD_VALUE		HL_SET_LOAD_VALUE	
19	ABST AKT			HL_SET_RELATION_OFFSET		HL_SET_RELATION_OFFSET	
20							
21	OPTIONEN	Darüber		ABOVE		ABOVE	
22	Gleich	Darunter		SAME		BELOW	
23	Keine	Zwischen		NONE		BETWEEN	
24	VERCK EIN	VERDCK AUS		HL_REDRAW_MODE ON		HL_REDRAW_MODE OFF	
25		MENÜ 2				Sm_2d_hidden_line_2	
26							
27							

VERDECKTE LINIEN 2

01	2D VERDECKTE LINIEN 2						
02							
03	FLÄCHFARBE	Ändern		HL_SET_FACE_COLOR		HL_SET_FACE_COLOR	
04	Setzen	Rücksetzen		HL_DEFAULT_FACE_COLOR		HL_DEFAULT_FACE_COLOR RESET	
05							
06	FLÄCHLÖSCH			HL_DELETE_FACE			
07							
08	EBENE ZEIG	ElemInZchn		Hl_show_z_level 1		Hl_show_z_level 1	
09	FlächInZch	ElemInTeil		Hl_show_faces 1		Hl_show_z_level 0	
10	FlächInTl			Hl_show_faces 0			
11							
12	ZEIG ALLES			SHOW_GLOBAL ALL ON			
13							
14	OPTIONEN	Alles		ALL		ALL	
15							
16	VERD ERST	Mit S-Kop		HL_GENERATE_HIDDEN		HL_GENERATE_HIDDEN	
17		Ohne S-Kop				HL_GENERATE_HIDDEN NO_BACKUP	
18	S-KOP LAD			LOAD "workfile.hid"			
19							
20	VERD LINIE	Zeigen Ein		Hl_show_hidden ON		Hl_show_hidden ON	
21		Zeigen Aus				Hl_show_hidden OFF	
22	Farbe Set	Typ Setzen		HL_SET_COLOR		HL_SET_LINETYPE	
23	Farbe Änd	Art Ändern		Hl_change_color		Hl_change_ltype	
24	Farbe Rset			HL_SET_COLOR RESET			
25	MENÜ 1			Sm_2d_hidden_line_1		Sm_2d_hidden_line_2	
26							
27							

A2 Häufige Fehler

- Das Editieren eines Makrotextes wurde mit der BREAK-Taste oder der ESC-Taste statt mit der Tastenkombination <CTRL>+<D> beendet. Vorgenommene Änderungen sind dann verloren.
- Ein Makrofile wurde geändert, aber nicht mit INPUT geladen. Im Speicher befindet sich noch die alte Version.
- Häufig wird der Makrotext zunächst mit EDIT_MACRO geschrieben, auf Platte gespeichert, mit INPUT geladen und ausgetestet. Der Benutzer stellt einen Fehler fest, ruft EDIT_MACRO auf, verbessert den Fehler und speichert das Makro mit Hilfe der Tastenkombination <CTRL>+<D> ab. Die verbesserte Version befindet sich jetzt im Hauptspeicher und könnte ohne vorheriges Laden ausgetestet werden. Auf der Platte befindet sich noch die alte, fehlerhafte Version. Der Benutzer ist sich aber über diese Tatsache nicht im klaren und lädt den auf der Platte befindlichen Makrotext in den Hauptspeicher. Dadurch wird das zwischenzeitlich verbesserte Makro vom fehlerhaften Makro überschrieben.
- In einem Ausdruck stimmt die Anzahl der öffnenden Klammern nicht mit der Anzahl der schließenden Klammern überein.
- Bei einem Kommentar fehlt das den Kommentarbeginn oder das Kommentarende kennzeichnende Symbol.
- Bei einer Textkonstanten wurde das abschließende Hochkomma vergessen.
- Daten unterschiedlichen Typs werden in unzulässiger Weise miteinander verknüpft. (Beispiel: Addition eines Vektors und einer Zahl).
- Eine Variable wurde mit einen reservierten Schlüsselwort gekennzeichnet. Der Ladevorgang wird in diesem Fall mit einem akustischen Signal abgebrochen. TRACE liefert in diesem Fall das Schlüssenwort LOCAL als letzte Anweisung.
- Eine noch nicht definierte Variable wird benutzt. Das System meldet in diesem Fall: "Das Makro *makroname* ist nicht definiert".
- In einer Schleife wird das Schlüsselwort "EXIT_IF" ohne den Unterstrich, also als "EXIT IF" geschrieben. Das Makro wird dann ohne Fehlermeldung abgebrochen.
- Innerhalb einer IF-Anweisungsfolge wird statt ELSE_IF ein weiteres IF benutzt.
- Eine IF-Anweisungsfolgen wird nicht mit END_IF abgeschlossen.
- Eine LOOP-, REPEAT- oder WHILE-Schleife wird nicht mit END_LOOP, UNTIL (Boolscher Ausdruck) oder END_WHILE abgeschlossen.
- Geschachtelte Schleifen überlappen sich.
- Innerhalb einer Schleife wird die Anzahl der Durchläufe mit einer Anweisung wie z.B. "LET N (N + 1)" gezählt. Es wurde jedoch vergessen, der Variablen "N" zuvor einen Anfangswert (z.B. "0") zuzuweisen.
- Das ein Makro abschließende "END_DEFINE" wurde vergessen. Bei Laden mit "INPUT" erscheint dann die Meldung "Makrodefinition eingeben".

- Eine zu Beginn einer Makrotextdatei stehende Kommentarklammer wurde nicht geschlossen. Der ganze nachfolgende Makrotext wird dadurch als Kommentar interpretiert. Beim Laden des Makrotextes mittels INPUT hängt sich der Ladeprozeß auf und ME10 reagiert auf keine normalen Benutzereingaben. Abbruch durch Betätigung der BREAK-Taste (bei MS-DOS-Systemen CTRL- und BREAK-Taste) möglich.
- Eine Kommentarklammer innerhalb des Makrotextes wurde nicht geschlossen. Alle bis zur nächsten öffnenden Kommentarklammer folgenden Anweisungen werden dadurch ignoriert.

A3 Übungsaufgaben

Aufgabe 3_1

Geben Sie folgendes Makro mit Hilfe des ME10-Texteditors ein, speichern Sie das Makro unter dem Namen "pfeil.mac" ab und starten Sie den Makrolauf.

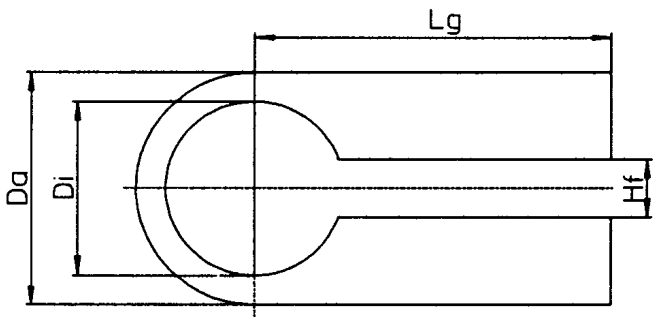
```
DEFINE Pfeil
 LOCAL P1
 LOCAL P2
 LOCAL P3
 LOCAL P4
 LOCAL P5
 LOCAL P6
 LOCAL P7
 LOCAL P8
 LOCAL H
 LOCAL W
 LOCAL T
 LOOP
  READ PNT 'Punkt für Pfeilspitze bestimmen' P1
  READ PNT 'Punkt für Pfeilende bestimmen' RUBBER_LINE P1 P2
  LET H (0.6 * (P2 - P1))
  LET W (0.3 * (ROT H 90))
  LET T (0.4 * W)
  LET P3 (P1 + H + W)
  LET P6 (P1 + H - W)
  LET P4 (P1 + H + T)
  LET P5 (P1 + H - T)
  LET P7 (P2 + T)
  LET P8 (P2 - T)
  LINE POLYGON WHITE SOLID P1 P3 P4 P7 P8 P5 P6 P1
  END
 END_LOOP
END_DEFINE
```

Aufgabe 4_1

Schreiben Sie ein Makro, das die nachfolgend abgebildete Schelle in folgenden Abmessungen zeichnet:

Di = 30 Da = 40 Lg = 60 Hf = 10

Die Schelle ist in waagrechter Lage zu zeichnen. Dabei soll der Mittelpunkt der Bohrung im Ursprung des Eingabekoordinatensystems liegen.



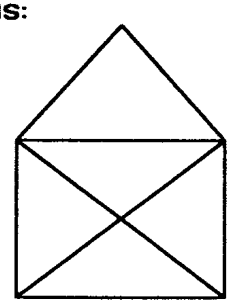
Aufgabe 6_1

Ändern Sie das Makro aus Aufgabe 4_1 so ab, daß die Maße Di, Da, Lg und Hf im Dialog abgefragt werden. (Zur Lösung der Aufgabe wird die in Kapitel 9 beschriebene Vektorfunktion "PNT_XY" benötigt.)

Aufgabe 6_2

Schreiben Sie ein Makro zum Zeichnen eines Nikolaus-Hauses. Zunächst soll der Bildschirm gelöscht und eine Zeichnung mit der Spielbeschreibung geladen werden (siehe Bild). Danach kann der Benutzer selbst ein Haus zeichnen.

Kennen Sie das Spiel vom Haus des Nikolaus? Bei diesem Spiel geht es darum, ein Haus in einem Zug zu zeichnen. Dabei darf keine Linie doppelt gezeichnet werden. Das Nikolaus-Haus sieht so aus:



Sie sollen nun das Haus zeichnen.
Bestimmen Sie zunächst den Startpunkt Ihres Linienzugs.
In der Dialogzeile erscheint danach der Nikolaus-Text. Geben Sie nach jeder Silbe den nächsten Punkt des zu zeichnenden Linienzugs ein.

Zusatzaufgabe (nur bei Hardware mit Frequenzgenerator möglich): Begleiten Sie jede Benutzereingabe mit einem Ton (Funktion TONE). Die Töne sollten möglichst eine kleine Melodie ergeben. Hierzu können Sie die folgenden Frequenzangaben verwenden:

1. Oktave:

Ton	c	#c	d	#d	e	f	#f	g	#g	a	#a	h	c
f[Hz]	523	554	587	622	659	698	740	784	831	880	932	988	1047

2. Oktave:

Ton	c	#c	d	#d	e	f	#f	g	#g	a	#a	h	c
f[Hz]	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976	2093

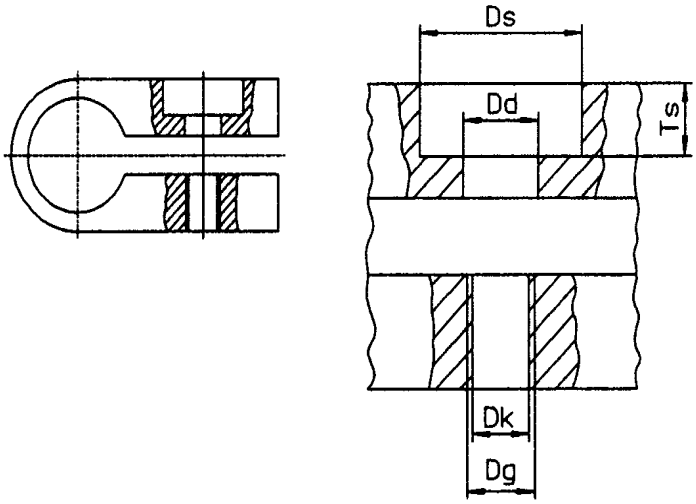
Aufgabe 9_1

Ändern Sie das Makro aus Aufgabe 6_1 so ab, daß die Schelle an beliebigen Stellen der Zeichnung in beliebiger Winkellage gezeichnet werden kann.

Aufgabe 10_1

Ändern Sie das Makro aus Aufgabe 9_1 wie folgt ab:

- Die Schenkel der Schelle werden mit einer Bohrung zur Aufnahme einer Spannschraube nach DIN 6912 versehen (siehe Abbildung). Die Mittelachse der Bohrung soll dabei die Klemmfuge in etwa zwei gleich lange Abschnitte unterteilen.
- Die Abmessungen der Spannschraubenbohrung sind im Dialog einzulesen.
- Die Bohrung ist aufgebrochen darzustellen. Dabei sollen die Bruchlinien mit einem universell verwendbaren Untermakro gezeichnet werden.



Aufgabe 10_2

Ändern Sie das Makro aus Aufgabe 10_1 so ab, daß lediglich Position, Winkellage und Innendurchmesser der Schelle im Dialog eingelesen werden. Hierbei sollen nur bestimmte Durchmesser akzeptiert werden. Die übrigen Abmessungen der Schelle sollen, abhängig vom Innendurchmesser, mit Hilfe von Steueranweisungen wie folgt festgelegt werden:

Di	Da	Lg	Hf	Dg	Dk	Dd	Ds	Ts
16	25	31.5	4	5	4.13	5.5	13	5.5
25	40	50	6.3	8	6.65	9	20	8
40	63	80	10	12	10.11	13.5	26	11

Nach dem Zeichnen einer Schelle soll der Benutzer zu einer erneuten Eingabe aufgefordert werden.

Aufgabe 10_3

a) Ein grüner Planet soll eine gelbe Sonne umkreisen. Die Position der Sonne und die Anzahl der Planetenumläufe sind vom Benutzer abzufragen.

b) Erweitern Sie Ihr Makro so, daß ein weiterer grüner Planet die Sonne in einem etwas größeren Abstand und mit geringerer Umlaufgeschwindigkeit umkreist.

c) Schreiben Sie ein weiteres Makro, bei dem der zweite Planet den ersten Planeten pro Sonnenumlauf "n" mal umkreist. Die Anzahl der Sonnenumläufe des ersten Planeten und "n" sollen vom Benutzer gewählt werden können. Außerdem soll der Benutzer bestimmen können, ob die Planetenbahnen während des Umlaufs gelöscht oder beibehalten werden.

Aufgabe 11_1

Ändern Sie das Makro aus Aufgabe 10_2 so ab, daß die vom Bohrungsdurchmesser abhängigen Daten aus einer sequentiellen Datei gelesen werden. Hierbei soll gelten:

Di	Da	Lg	Hf	Dg	Dk	Dd	Ds	Ts
10	16	20	2.5	3	2.46	3.4	8	3.3
12.5	20	25	3.15	4	3.24	4.5	10	5.5
16	25	31.5	4	5	4.13	5.5	13	5.5
20	31	40	5	6	4.92	6.6	13	6.5
25	40	50	6.3	8	6.65	9	20	8
31.5	50	63	8	10	8.38	11	24	9.5
40	63	80	10	12	10.11	13.5	26	11
50	80	100	12.5	16	13.84	17.5	33	15
63	100	125	16	20	17.29	22	40	17.5
80	125	160	20	24	20.75	26	46	20.5
100	160	200	25	30	26.21	33	61	25.5

Aufgabe 12_1

In Ihrem Verzeichnis befindet sich ein lauffähiges C-Programm mit dem Namen "hohlwell". Das Programm berechnet die Biegelinie einer an den Enden mit einem Loslager und einem Festlager gelagerten Hohlwelle. Die Hohlwelle wird dabei an einer beliebigen Stelle zwischen den Lagern mit einer Einzelkraft belastet.

Das Programm benötigt folgende Werte in der Standardeingabe:

- Rohraußendurchmesser [mm]
- Rohrinnendurchmesser [mm]
- Elastizitätsmodul des Rohres [N/mm¨]
- Lagerabstand [mm]
- Kraft [N]
- Abstand zwischen linkem Lager und Kraftangriffspunkt [mm]
- Anzahl der Zwischenpunkte, für die die Durchbiegung berechnet werden soll [/]

Die nachfolgende Tabelle zeigt Ausgaben des Programms in der Standardausgabe und Eingaben, die der Benutzer in der Standardeingabe vornimmt. Benutzereingaben sind kursiv dargestellt. Die Zeilen- und Spaltennummerierung sowie die sie begrenzenden senkrechten und waagerechten Striche werden nicht vom Programm ausgegeben, sondern dienen nur der besseren Lesbarkeit im vorliegenden Text.

Die Zeilen 1 bis 5 werden sofort nach dem Programmstart ausgegeben. Danach wird der Benutzer zur Eingabe des Außendurchmessers (Zeile 6), des Innendurchmessers (Zeile 7) usw. aufgefordert. Nachdem er die Anzahl der zu berechnenden Zwischenpunkte eingegeben hat (Zeile 12), führt das Programm die Berechnung durch. Die Ausgabe der Ergebnisse beginnt mit einer Leerzeile (Zeile 13) und endet mit der Ausgabe des letzten

Wellenpunktes und dessen Absenkung (Zeile 33). Die Anzahl der ausgegebenen Durchbiegungswerte ist stets um 2 größer als die Anzahl der Zwischenpunkte, die der Benutzer eingegeben hat. Die Zwischenpunkte werden vom Programm gleichmäßig über den Lagerabstand verteilt.

```
      |Spalte
      |         10        20        30        40        50        60
Zeile |12345678901234567890123456789012345678901234567890123456789012345678890
------+------------------------------------------------------------------------
  1   |****************************************
  2   |* Durchbiegung einer mit einem Festlager  *
  3   |* und einem Loslager gelagerten Hohlwelle *
  4   |****************************************
  5   |
  6   |Aussendurchmesser [mm]    : 100
  7   |Innendurchmesser [mm]     : 80
  8   |E-Modul [N/mm^2}          : 210000
  9   |Lagerabstand [mm]         : 1000
 10   |Kraft [N]                 : 850
 11   |Kraftangriff bei x = [mm] : 350
 12   |Anzahl Zwischenpunkte     : 9
 13   |
 14   |Flaechentraegheitsmoment [mm^4]           : 2898058
 15   |Durchbiegung am Kraftangriffspunkt [mm]   : 0.0241
 16   |Ort der maximalen Durchbiegung x_max [mm] :  459.2
 17   |Maximale Durchbiegung w_max [mm]          : 0.0258
 18   |
 19   |Zwischenwerte:
 20   |
 21   |  x [mm]    Durchbiegung [mm]
 22   |--------    -----------------
 23   |     0.0    0.0000
 24   |   100.0    0.0086
 25   |   200.0    0.0163
 26   |   300.0    0.0221
 27   |   400.0    0.0253
 28   |   500.0    0.0256
 29   |   600.0    0.0234
 30   |   700.0    0.0192
 31   |   800.0    0.0136
 32   |   900.0    0.0071
 33   |  1000.0    0.0000
```

Schreiben Sie ein Makro, das folgende Anforderungen erfüllt:

- Die zur Berechnung der Biegelinie erforderlichen Daten werden im Makro eingelesen und auf Zulässigkeit (z.B. Innendurchmesser kleiner Außendurchmesser) geprüft.
- Die Berechnung findet im C-Programm statt.
- Die Datenaustausch zwischen Makro und C-Programm findet über eine Plattendatei statt.
- Das Makro lädt eine Zeichnung, die eine Prinzipskizze der unbelasteten Welle enthält und zeichnet die Biegelinie in einem gut sichtbaren Maßstab ein. Die Kraft wird am Angriffspunkt als Pfeil dargestellt und entsprechend ihrem Zahlenwert beschriftet. Ort und Größe der maximalen Durchbiegung werden ebenfalls dargestellt und beschriftet.

Die Aufgabe kann ohne Kenntnisse des C-Programms bearbeitet werden. Für interessierte Leser ist das Programm nachfolgend aufgelistet.

```
/* C-Programm "hohlwelle.c" zur Berechnung der */
/* Biegelinie einer mit einem Festlager und einem */
/* Loslager gelagerten Hohlwelle */
/* Fischer, 02.10.91, Stand 02.10.91 */

#include <stdio.h>
#include <math.h>
#define pi 3.1415265

main ()
{
 float da,di,emod,l,a,b,f,itraeg,step,x,x_max,w_x,w_a,w_max;
 int nz,i;

 printf("*****************************************\n");
 printf("* Durchbiegung einer mit einem Festlager  *\n");
 printf("* und einem Loslager gelagerten Hohlwelle *\n");
 printf("*****************************************\n");

 printf("\nAussendurchmesser [mm]    : ");
 scanf("%f",&da);
 printf("\nInnendurchmesser [mm]     : ");
 scanf("%f",&di);
 printf("\nE-Modul [N/mm^2}          : ");
 scanf("%f",&emod);
 printf("\nLagerabstand [mm]         : ");
 scanf("%f",&l);
 printf("\nKraft [N]                 : ");
 scanf("%f",&f);
 printf("\nKraftangriff bei x = [mm] : ");
 scanf("%f",&a);
 printf("\nAnzahl Zwischenpunkte     : ");
 scanf("%d",&nz);

 b=l-a;
 itraeg=pi*(da*da*da*da-di*di*di*di)/64;
 printf("\n\nFlaechentraegheitsmoment [mm^4]           : %6.0f\n",itraeg);
 w_a = f*a*a*b*b/3/emod/itraeg/l;
 printf("Durchbiegung am Kraftangriffspunkt [mm]   : %6.4f\n",w_a);
 if(a>b)
 {
  x_max=sqrt((l*l-b*b)/3);
  w_max=f*b*sqrt((l*l-b*b)*(l*l-b*b)*(l*l-b*b))/15.59/emod/itraeg/l;
 }
 else
 {
  x_max=l-sqrt((l*l-a*a)/3);
  w_max=f*a*sqrt((l*l-a*a)*(l*l-a*a)*(l*l-a*a))/15.59/emod/itraeg/l;
 }
 printf("Ort der maximalen Durchbiegung x_max [mm] : %6.1f\n",x_max);
 printf("Maximale Durchbiegung w_max [mm]          : %6.4f\n",w_max);
 step=l/((float)nz+1);

 printf("\nZwischenwerte:\n\n");
 printf("  x [mm]    Durchbiegung [mm]\n");
 printf("--------    -----------------\n");
 for(i=0; i<=(nz+1); i++)
 {
  x = (float)i*step;
   if((a<=0.001)||((l-a)<0.001))
    w_x = 0;
   else if(x<a)
    w_x=f*a*b*b/6/emod/itraeg*((1+l/b)*x/l-x*x*x/a/b/l);
   else
    w_x=f*a*a*b/6/emod/itraeg*((1+l/a)*(l-x)/l-(l-x)*(l-x)*(l-x)/a/b/l);
  printf("%8.1f    %6.4f\n",x,w_x);
 }
}
```

Aufgabe 12_2

Schreiben Sie das Makro aus Aufgabe 12_1 so um, daß der Datenaustausch zwischen dem Makro und dem C-Programm über benannte Pipes erfolgt.

Aufgabe 13_1

Schreiben Sie ein Makro zum Unterteilen einer Strecke nach der Regel des goldenen Schnitts. Die Strecke ist am Unterteilungspunkt aufzutrennen. Der Unterteilungspunkt soll mit einer normal zur Strecke verlaufenden Hilfsgeometrieline gekennzeichnet werden.

Anmerkung: Der goldene Schnitt soll ein vom Betrachter als besonders ausgewogen empfundenes Teilungsverhältnis liefern. Er besagt, daß sich die Gesamtlänge einer Strecke zur Länge des großen Teilstücks wie die Länge des großen Teilstücks zur Länge des kleinen Teilstücks verhält. Die Berechnung des Verhältnisses von Länge des großen Teilstücks zur Gesamtlänge ergibt einen Wert von etwa 0,61803399.

Aufgabe 13_2

Schreiben Sie ein Makro zum Zeichnen von Mittelkreuzen in vorhandene Kreise. Die Mittelkreuze sollen in der Farbe Gelb und in der Linienart "Strichpunktlinie" gezeichnet werden. Die vor dem Aufruf des Makros eingestellten Werte für Farbe und Linienart sollen nach der Makroausführung wieder vorliegen.

Aufgabe 13_3

Schreiben Sie ein Makro zum Trimmen von Linien. Unter Trimmen versteht man das Verlängern oder Verkürzen einer Linie bis zu einem angegebenen Punkt. Der Benutzer soll die zu trimmende Linie auswählen und dabei gleichzeitig das zu trimmende Ende bestimmen. Farbe und Linienart der zu trimmenden Linie sind beizubehalten. Nach Beendigung des Makros sollen die zuvor eingestellten Systemwerte für Farbe und Linienart wieder eingestellt sein. Bei einem vorzeitigen Abbruch des Makros durch den Benutzer muß die ursprüngliche Linie noch vorhanden sein.

Lösungshinweis: Die zu trimmenden Linie wird zunächst gelöscht. Danach wird, ausgehend vom bisherigen Anfangspunkt, eine neue Linie in gleicher Richtung in der vom Benutzer bestimmten Länge gezeichnet.

Aufgabe 13_4

Schützen Sie (falls noch nicht geschehen), ihr Makro vor einer Fehlbedienung durch den Benutzer. Folgende Fehler sollten abgefangen werden:

- Bei der Elementauswahl wird kein Element gefunden
- Das ausgewählte Element ist keine Linie
- Der Benutzer versucht, ein Element auf die Länge Null zu verkürzen (Anfangs- und Endpunkt fallen zusammen).

Aufgabe 13_5

Erweitern Sie das vorangegangene Makro so, daß damit auch Kreisbögen getrimmt werden können.

Aufgabe 14_1

Binden Sie das Makro zum Trimmen von Linien an geeigneter Stelle in das Menüsystem ein. Geeignet wäre z.B. ein freies Feld des Bildschirmmenüs "ERSTELLEN" oder ein freies Tablettfeld. Das Einbinden soll sich auf die aktuelle ME10-Sitzung beschränken. Verändern Sie also keine Systemdateien.

Aufgabe 14_2

Schreiben Sie ein Makro, das die Hauptwerte der dezimalgeometrischen Normzahlenreihen R5, R10 und R20 (DIN 323) in einem Bildschirmmenü anzeigt und eine Übernahme als Eingabewerte durch Antippen ermöglicht. Die Normzahlen sind in nachfolgender Tabelle angegeben.

Hauptwerte		
R5	R10	R20
1,0	1,0	1,0
		1,12
	1,25	1,25
		1,4
1,6	1,6	1,6
		1,8
	2,0	2,0
		2,24
2,5	2,5	2,5
		2,8
	3,15	3,15
		3,55

Hauptwerte		
R5	R10	R20
4,0	4,0	4,0
		4,5
	5,0	5,0
		5,6
6,3	6,3	6,3
		7,1
	8,0	8,0
		9,0
10,0	10,0	10,0

Aufgabe 15_1

In Kapitel 14.3.3 wurde ein Makro zum Zeichnen von Kegelstiften nach DIN 1 beschrieben. Der Benutzerdialog findet dabei über ein Bildschirmmenü statt. Verändern Sie das Makro so, daß der Dialog mit Hilfe von Anzeigetabellen geführt wird. Im einzelnen sollen folgende Bedingungen erfüllt werden:

Der Benutzer soll nur eine eingeschränkte Auswahl von Kegelstiften in folgender Art angeboten bekommen:

Nennduchmesser [mm]	Längen [mm]					
4	16	20	24	30	40	55
5	20	24	28	36	50	70
6	24	28	32	45	60	90
8	28	32	40	55	80	110
10	32	40	50	70	100	130

Eine logische Tabelle soll geeignete Überschriften der Anzeigetabelle sowie die Kuppenradien und alle in DIN 1 genormten Längen der Durchmesser 4, 5, 6, 8 und 10 enthalten. Die Beschränkung auf bestimmte Längen erfolgt dadurch, daß beim Belegen der Anzeigetabelle nur bestimmte Spalten der logische Tabelle verwendet werden. Kuppenradien "r" und Längen "l" der angegebenen Nenndurchmesser "d" werden in nachstehender Tabelle noch einmal aufgeführt.

d	l		r
	von	bis	
4	16	60	4
5	20	70	6
6	24	100	6
8	28	120	10
10	32	140	10

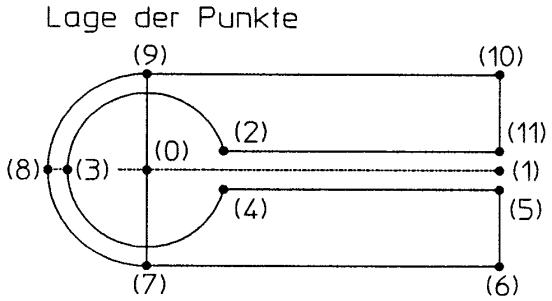
Stufung der Länge				
12	14	16	18	20
22	24	26	28	30
32	36	40	45	50
55	60	70	80	90
100	110	120	130	140

A4 Musterlösungen

Aufgabe 4_1

```
DEFINE Schelle1
{Zeichnen einer Schelle, Lage im Ursprung und horizontal}
{Maße fest vorgegeben}
{Fischer, 01.03.91, Stand 01.11.91}

ARC THREE_PTS WHITE SOLID
 14.14,-5 14.14,5 -15,0 0,-20 0,20 -20,0
LINE POLYGON
 14.14,-5 60,-5 60,-20 0,-20
LINE POLYGON
 0,20 60,20 60,5 14.14,5
END
LINE YELLOW DOT_CENTER
 -22,0 62,0 0,22 0,-22
WHITE SOLID
END

END_DEFINE
```

Aufgabe 6_1




```
DEFINE Schelle2
{Zeichnen einer Schelle, Lage im Ursprung und horizontal}
{Maße werden vom Benutzer eingegeben}
{Fischer, 01.03.91, Stand 25.10.91}

LOCAL Di  LOCAL Da  LOCAL Lg  LOCAL Hf
LOCAL P2  LOCAL P3  LOCAL P4  LOCAL P5   LOCAL P6
LOCAL P7  LOCAL P8  LOCAL P9  LOCAL P10  LOCAL P11

{Einlesen der Geometriedaten}
READ NUMBER 'Länge der Schelle' Lg
READ NUMBER 'Innendurchmesser?' Di
READ NUMBER 'Außendurchmesser?' Da
READ NUMBER 'Höhe der Klemmfuge?' Hf

{Berechnung der Konturpunkte}
LET P2 (PNT_XY (0.5 * SQRT (Di^2 - Hf^2)) (Hf / 2))
LET P3 (PNT_XY (-Di / 2) 0)
LET P4 (PNT_XY (0.5 * SQRT (Di^2 - Hf^2)) (-Hf / 2))
LET P5 (PNT_XY Lg (-Hf / 2))
```

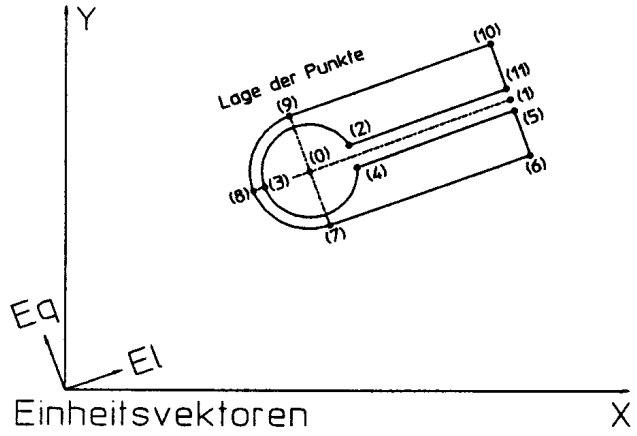
```
LET P6 (PNT_XY Lg (-Da / 2))
LET P7 (PNT_XY 0 (-Da / 2))
LET P8 (PNT_XY (-Da / 2 ) 0)
LET P9 (PNT_XY 0 (Da / 2))
LET P10 (PNT_XY Lg (Da / 2))
LET P11 (PNT_XY Lg (Hf / 2))

{Zeichnen der Kontur}
ARC THREE_PTS  WHITE SOLID
 P4 P2 P3 P7 P9 P8
LINE POLYGON
 P4 P5 P6 P7
LINE POLYGON
 P9 P10 P11 P2
END

{Zeichnen der Mittellinien}
LINE YELLOW DOT_CENTER
 (PNT_XY (Lg + 2) 0) (PNT_XY (-Da / 2 - 2) 0)
 (PNT_XY 0 (Da / 2 + 2)) (PNT_XY 0 (-Da / 2 - 2))
 WHITE SOLID
END

END_DEFINE
```

Aufgabe 6_2

```
DEFINE Niki
{Zeichnen eines Nikolaus-Hauses}
{Fischer, 08.01.91, Stand 18.11.91}

 LOCAL P1  LOCAL P2

 DELETE ALL CONFIRM
 LOAD 'nikidraw.mi'
 TONE 523 0.2 0.5
 TONE 659 0.2 0.5
 TONE 784 0.2 0.5
 TONE 1047 0.2 0.5
 END
 CATCH PERMANENT OFF
 READ PNT 'Startpunkt?' P1
 READ PNT 'Das ..' RUBBER_LINE P1 P2
 TONE 784 0.2 0.5
 LINE YELLOW SOLID P1 P2 END
 READ PNT 'ist ..' RUBBER_LINE P2 P1
 TONE 1047 0.2 0.5
 LINE P2 P1 END
 READ PNT 'das ..' RUBBER_LINE P1 P2
 TONE 1319 0.2 0.5
 LINE P1 P2 END
 READ PNT 'Haus ..' RUBBER_LINE P2 P1
 TONE 1568 0.2 0.5
 LINE P2 P1 END
 READ PNT 'vom ..' RUBBER_LINE P1 P2
 TONE 1319 0.2 0.5
 LINE P1 P2 END
 READ PNT 'Ni ..' RUBBER_LINE P2 P1
 TONE 1047 0.2 0.5
 LINE P2 P1 END
 READ PNT 'ko ..' RUBBER_LINE P1 P2
 TONE 1047 0.2 0.5
 LINE P1 P2 END
 READ PNT 'laus!' RUBBER_LINE P2 P1
 TONE 1047 0.5 0.5
 WAIT 0.5
 LINE P2 P1 WHITE SOLID END
 TONE 523 0.2 0.5
```

```
 TONE 659 0.2 0.5
 TONE 784 0.2 0.5
 TONE 1047 0.2 0.5
 END
 CATCH ALL
END_DEFINE
```

Aufgabe 9_1

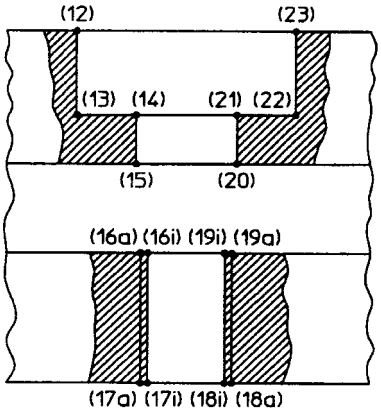



```
DEFINE Schelle3
{Zeichnen einer Schelle an beliebigen Stellen in beliebiger Lage}
{Maße werden vom Benutzer eingegeben}
{Fischer, 28.02.91, Stand 01.11.91}

LOCAL Di  LOCAL Da  LOCAL LS  LOCAL Hf  LOCAL El  LOCAL Eq
LOCAL P0  LOCAL Pr  LOCAL P1  LOCAL P2  LOCAL P3  LOCAL P4
LOCAL P5  LOCAL P6  LOCAL P7  LOCAL P8  LOCAL P9  LOCAL P10
LOCAL P11

{Einlesen der Geometriedaten}
READ PNT 'Mitte Bohrung?' P0
READ PNT 'Richtung der Klemmfuge?' RUBBER_LINE P0 Pr
READ NUMBER 'Länge der Schelle' Lg
READ NUMBER 'Innendurchmesser?' Di
READ NUMBER 'Außendurchmesser?' Da
READ NUMBER 'Höhe der Klemmfuge?' Hf

{Berechnung der Einheitsvektoren längs und quer}
LET El ((Pr - P0) / (LEN (Pr - P0)))
LET Eq (ROT El 90)

{Berechnung der Konturpunkte}
LET P1 (P0  + El * Lg)
LET P2 (P0 + (ROT (El * Di / 2) (ARCSIN (Hf / Di))))
LET P3 (P0 - El * Di / 2)
LET P4 (P0 + (ROT (El * Di / 2) (-ARCSIN (Hf / Di))))
LET P5 (P1 - Eq * Hf / 2)
LET P6 (P1 - Eq * Da / 2)
LET P7 (P0 - Eq * Da / 2)
LET P8 (P0 - El * Da / 2)
LET P9 (P0 + Eq * Da / 2)
LET P10 (P1 + Eq * Da / 2)
LET P11 (P1 + Eq * Hf / 2)

{Zeichnen der Kontur}
ARC THREE_PTS WHITE SOLID
 P4 P2 P3 P7 P9 P8
LINE POLYGON
 P4 P5 P6 P7
LINE POLYGON
```

```
 P9 P10 P11 P2 END
{Zeichnen der Mittellinien}
LINE YELLOW DOT_CENTER
 (P8 - 2 * El) (P1 + 2 * El)
 (P7 - 2 * Eq) (P9 + 2 * Eq)
 WHITE SOLID
END

END_DEFINE
```

Aufgabe 10_1

Lage der zusätzlichen Punkte




```
DEFINE Schelle4
{Zeichnen einer Schelle mit Spannschraubenbohrung}
{Maße werden vom Benutzer eingegeben}
{28.02.91, Stand 01.11.91}

LOCAL Lg    LOCAL Di    LOCAL Da    LOCAL Hf    LOCAL Dg
LOCAL Dk    LOCAL Dd    LOCAL Ds    LOCAL Ts    LOCAL Lf
LOCAL Sm    LOCAL P0    LOCAL Pr    LOCAL P1    LOCAL P2
LOCAL P3    LOCAL P4    LOCAL P5    LOCAL P6    LOCAL P7
LOCAL P8    LOCAL P9    LOCAL P10   LOCAL P11   LOCAL P12
LOCAL P13   LOCAL P14   LOCAL P15   LOCAL P16a  LOCAL P16i
LOCAL P17a  LOCAL P17i  LOCAL P18a  LOCAL P18i  LOCAL P19a
LOCAL P19i  LOCAL P20   LOCAL P21   LOCAL P22   LOCAL P23
LOCAL El    LOCAL Eq

{Einlesen der Geometriedaten}
READ PNT 'Mitte Bohrung?' P0
READ PNT 'Richtung der Klemmfuge?' RUBBER_LINE P0 Pr
READ NUMBER 'Länge der Schelle' Lg
READ NUMBER 'Innendurchmesser?' Di
READ NUMBER 'Außendurchmesser?' Da
READ NUMBER 'Höhe der Klemmfuge?' Hf
READ NUMBER 'Gewinde-Nenndurchmesser?' Dg
READ NUMBER 'Gewinde-Kerndurchmesser?' Dk
READ NUMBER 'Durchmesser des Durchgangslochs?' Dd
READ NUMBER 'Durchmesser Schraubenkopfsenkung?' Ds
READ NUMBER 'Tiefe Schraubenkopfsenkung?' Ts

{Berechnung der Einheitsvektoren längs und quer}
LET El ((Pr - P0) / (LEN (Pr - P0)))
LET Eq (ROT El 90)

{Berechnung der Konturpunkte}
LET P1 (P0 + El * Lg)
LET P2 (P0 + (ROT (El * Di / 2) (ARCSIN (Hf / Di))))
```

```
LET P3 (P0 - El * Di / 2)
LET P4 (P0 + (ROT (El * Di / 2) (-ARCSIN (Hf / Di))))
LET P5 (P1 - Eq * Hf / 2)
LET P6 (P1 - Eq * Da / 2)
LET P7 (P0 - Eq * Da / 2)
LET P8 (P0 - El * Da / 2)
LET P9 (P0 + Eq * Da / 2)
LET P10 (P1 + Eq * Da / 2)
LET P11 (P1 + Eq * Hf / 2)
LET Lf (LEN (P1 - P0) - Di / 2)
LET Sm (Lf / 2 + Di / 2)
LET P12 (P9 + El * (Sm - Ds / 2))
LET P13 (P12 - Eq * Ts)
LET P14 (P13 + El * ((Ds - Dd) / 2))
LET P15 (P14 - Eq * ((Da - Hf) / 2 - Ts))
LET P16a (P0 - Eq * Hf / 2 + El * (Sm - Dg / 2))
LET P16i (P0 - Eq * Hf / 2 + El * (Sm - Dk / 2))
LET P17a (P0 - Eq * Da / 2 + El * (Sm - Dg / 2))
LET P17i (P0 - Eq * Da / 2 + El * (Sm - Dk / 2))
LET P18a (P17a + El * Dg)
LET P18i (P17i + El * Dk)
LET P19a (P16a + El * Dg)
LET P19i (P16i + El * Dk)
LET P20 (P15 + El * Dd)
LET P21 (P14 + El * Dd)
LET P22 (P13 + El * Ds)
LET P23 (P12 + El * Ds)

{Zeichnen der Kontur}
SPLITTING ON
ARC THREE_PTS WHITE SOLID
 P4 P2 P3 P7 P9 P8
LINE POLYGON
 P4 P5 P6 P7
LINE POLYGON
 P9 P10 P11 P2
LINE POLYGON
 P12 P13 P22 P23 END
LINE
 P14 P15 P21 P20 P16i P17i P19i P18i
END

{Zeichnen der Bruchlinien}
Bruch (0.35 * P10 + 0.65 * P23) (0.5 * P11 + 0.5 * P20)
Bruch (0.35 * P5 + 0.65 * P19a) (0.35 * P6 + 0.65 * P18a)
Bruch (P12 - 0.35 * (P10 - P23)) (P15 - 0.5 * (P11 - P20))
Bruch (P16a - 0.35 * (P5 - P19a)) (P17a - 0.35 * (P6 - P18a))

{Schraffieren des Aufbruchs}
HATCH_DIST 2.5
HATCH_REF_PT P12
HATCH_ANGLE (ANG El + 45)
HATCH AUTO YELLOW SOLID
 (P12 - El * 0.5 - Eq * 0.5) (P23 + El * 0.5 - Eq * 0.5)
 (P16i - El * 0.5 - Eq * 0.5) (P19i + El * 0.5 - Eq * 0.5)
END

{Zeichnen der Gewinde-Außenlinien}
LINE YELLOW SOLID
 P16a P17a P19a P18a

{Zeichnen der Mittellinien}
LINE YELLOW DOT_CENTER
 (P8 - 2 * El) (P1 + 2 * El) (P7 - 2 * Eq) (P9 + 2 * Eq)
 ((P17a + P18a) / 2 - 2 * Eq) ((P12 + P23) / 2 + 2 * Eq)
 WHITE SOLID
END

HATCH_ANGLE 45

END_DEFINE
```

```
DEFINE Bruch
{Makro zum Zeichnen von Bruchlinien zwischen zwei Punkten}
{Fischer, 28.02.91, Stand 01.11.91}
{Nstep = Anzahl der Schritte, Amp = Amplitudenfaktor}
{Amplitude = Schrittweite * Amp}

PARAMETER Start  PARAMETER Ziel

LOCAL Nstep  LOCAL Amp  LOCAL Sl  LOCAL Sq
LOCAL Sig    LOCAL Ps   LOCAL Is

LET Nstep 7
LET Amp 0.35
LET Sl ((Ziel - Start) / Nstep)
LET Sq (ROT (Sl * Amp) 90)
LET Sig -1
LET Ps Start
LET Is 0

SPLINE YELLOW SOLID
 Start
 WHILE (Is < (Nstep - 1))
  LET Is (Is + 1)
  LET Sig (-Sig)
  LET Ps (Start + Is * Sl + Sig * RND * Sq)
  Ps
 END_WHILE
 Ziel
 END
 WHITE SOLID
END

END_DEFINE
```

Aufgabe 10_2

```
DEFINE Schelle5
{Zeichnen einer Schelle}
{Festlegung von abhhängigen Maßen mit Steueranweisungen}
{Fischer, 28.02.91, Stand 01.11.91}

LOCAL Lg    LOCAL Di    LOCAL Da    LOCAL Hf    LOCAL Dg
LOCAL Dk    LOCAL Dd    LOCAL Ds    LOCAL Ts    LOCAL Lf
LOCAL Sm    LOCAL P0    LOCAL Pr    LOCAL P1    LOCAL P2
LOCAL P3    LOCAL P4    LOCAL P5    LOCAL P6    LOCAL P7
LOCAL P8    LOCAL P9    LOCAL P10   LOCAL P11   LOCAL P12
LOCAL P13   LOCAL P14   LOCAL P15   LOCAL P16a  LOCAL P16i
LOCAL P17a  LOCAL P17i  LOCAL P18a  LOCAL P18i  LOCAL P19a
LOCAL P19i  LOCAL P20   LOCAL P21   LOCAL P22   LOCAL P23
LOCAL El    LOCAL Eq    LOCAL Nochmal

LOOP {Schleife zur wiederholten Makroausführung}

{Einlesen der Geometriedaten}
READ PNT 'Mitte Bohrung?' P0
READ PNT 'Richtung der Klemmfuge?' RUBBER_LINE P0 Pr
 LOOP
  READ NUMBER 'Innendurchmesser?' Di
  {Prüfen der Eingabe und Festlegen der übrigen Maße}
  IF (Di = 16)
   LET Da 25
   LET Lg 31.5
   LET Hf 4
   LET Dg 5
   LET Dk 4.13
   LET Dd 5.5
   LET Ds 13
   LET Ts 5.5
```

```
  LET Nochmal FALSE
 ELSE_IF (Di = 25)
  LET Da 40
  LET Lg 50
  LET Hf 6.3
  LET Dg 8
  LET Dk 6.65
  LET Dd 9
  LET Ds 20
  LET Ts 8
  LET Nochmal FALSE
 ELSE_IF (Di = 40)
  LET Da 63
  LET Lg 80
  LET Hf 10
  LET Dg 12
  LET Dk 10.11
  LET Dd 13.5
  LET Ds 26
  LET Ts 11
  LET Nochmal FALSE
 ELSE
  LET Nochmal TRUE
  BEEP
  DISPLAY ('Zulässig nur Di = 16, 25 oder 40!')
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP

{Berechnung der Einheitsvektoren längs und quer}
LET El ((Pr - P0) / (LEN (Pr - P0)))
LET Eq (ROT El 90)

SPLITTING ON

{Berechnung der Konturpunkte}
LET P1 (P0 + El * Lg)
LET P2 (P0 + (ROT (El * Di / 2) (ARCSIN (Hf / Di))))
LET P3 (P0 - El * Di / 2)
LET P4 (P0 + (ROT (El * Di / 2) (-ARCSIN (Hf / Di))))
LET P5 (P1 - Eq * Hf / 2)
LET P6 (P1 - Eq * Da / 2)
LET P7 (P0 - Eq * Da / 2)
LET P8 (P0 - El * Da / 2)
LET P9 (P0 + Eq * Da / 2)
LET P10 (P1 + Eq * Da / 2)
LET P11 (P1 + Eq * Hf / 2)
LET Lf (LEN (P1 - P0) - Di / 2)
LET Sm (Lf / 2 + Di / 2)
LET P12 (P9 + El * (Sm - Ds / 2))
LET P13 (P12 - Eq * Ts)
LET P14 (P13 + El * ((Ds - Dd) / 2))
LET P15 (P14 - Eq * ((Da - Hf) / 2 - Ts))
LET P16a (P0 - Eq * Hf / 2 + El * (Sm - Dg / 2))
LET P16i (P0 - Eq * Hf / 2 + El * (Sm - Dk / 2))
LET P17a (P0 - Eq * Da / 2 + El * (Sm - Dg / 2))
LET P17i (P0 - Eq * Da / 2 + El * (Sm - Dk / 2))
LET P18a (P17a + El * Dg)
LET P18i (P17i + El * Dk)
LET P19a (P16a + El * Dg)
LET P19i (P16i + El * Dk)
LET P20 (P15 + El * Dd)
LET P21 (P14 + El * Dd)
LET P22 (P13 + El * Ds)
LET P23 (P12 + El * Ds)

{Anlegen eines Teils mit der DIN-Bezeichnung}
INIT_PART ('Schelle ' + STR Di + ' x ' + STR Lg)

{Zeichnen der Kontur}
ARC THREE_PTS WHITE SOLID
 P4 P2 P3 P7 P9 P8
```

```
 LINE POLYGON
  P4 P5 P6 P7
 LINE POLYGON
  P9 P10 P11 P2
 LINE POLYGON
  P12 P13 P22 P23
 LINE
  P14 P15 P21 P20 P16i P17i P19i P18i
 END

 {Zeichnen der Bruchlinien}
 Bruch (0.35 * P10 + 0.65 * P23) (0.5 * P11 + 0.5 * P20)
 Bruch (0.35 * P5 + 0.65 * P19a) (0.35 * P6 + 0.65 * P18a)
 Bruch (P12 - 0.35 * (P10 - P23)) (P15 - 0.5 * (P11 - P20))
 Bruch (P16a - 0.35 * (P5 - P19a)) (P17a - 0.35 * (P6 - P18a))

 {Schraffieren des Aufbruchs}
 HATCH_DIST 2.5
 HATCH_REF_PT P12
 HATCH_ANGLE (ANG El + 45)
 HATCH AUTO YELLOW SOLID
  (P12 - El * 0.5 - Eq * 0.5) (P23 + El * 0.5 - Eq * 0.5)
  (P16i - El * 0.5 - Eq * 0.5) (P19i + El * 0.5 - Eq * 0.5)
 END

 {Zeichnen der Gewinde-Außenlinien}
 LINE YELLOW SOLID
  P16a P17a P19a P18a

 {Zeichnen der Mittellinien}
 LINE YELLOW DOT_CENTER
  (P8 - 2 * El) (P1 + 2 * El) (P7 - 2 * Eq) (P9 + 2 * Eq)
  ((P17a + P18a) / 2 - 2 * Eq) ((P12 + P23) / 2 + 2 * Eq)
  WHITE SOLID
 END
  HATCH_ANGLE 45

 END_PART

 END_LOOP

END_DEFINE
```

Aufgabe 10_3 a

```
DEFINE Planet_a
{Ein gruener Planet umkreist eine gelbe Sonne}
{Fischer, 07.03.91, Stand 18.09.91}

 LOCAL Csonne  LOCAL Cplanet1  LOCAL Rplanet1
 LOCAL Nmax    LOCAL Nu        LOCAL Alpha

 DELETE ALL CONFIRM
 READ PNT 'Sonnenmitte?' Csonne
 CIRCLE YELLOW SOLID
  Csonne 20
 END
 HATCH_DIST 0.1
 HATCH_ANGLE 0
 HATCH AUTO YELLOW SOLID
  Csonne
 END
 READ NUMBER 'Zahl der Umlaeufe des Planeten?' Nmax
 LET Nu 1
 WHILE (Nu <= Nmax)
  LET Alpha 0
  WHILE (Alpha < 360)
   LET Cplanet1 (Csonne + (PNT_RA 60 Alpha))
   CIRCLE GREEN SOLID
    Cplanet1 5
   HATCH AUTO GREEN SOLID
    Cplanet1
   END
   LET Rplanet1 (Csonne + (PNT_RA 55 Alpha))
   DELETE
    Rplanet1
   END
   LET Alpha (Alpha + 5)
  END_WHILE
  LET Nu (Nu + 1)
 END_WHILE
 LINE WHITE SOLID
 END
 HATCH_COLOR YELLOW
 HATCH_DIST 3.5
END_DEFINE
```

Aufgabe 10_3 b

```
DEFINE Planet_b
{Zwei Planeten umkreisen eine Sonne mit unterschiedlicher Geschwindigkeit}
{Fischer, 07.03.91, Stand 18.09.91}

 LOCAL Csonne    LOCAL Cplanet1  LOCAL Cplanet2
 LOCAL Rplanet1  LOCAL Rplanet2  LOCAL Nmax
 LOCAL Nu        LOCAL Alpha

 DELETE ALL CONFIRM
 READ PNT 'Sonnenmitte?' Csonne
 CIRCLE YELLOW SOLID
  Csonne 20
 END
 HATCH_DIST 0.1
 HATCH_ANGLE 0
 HATCH AUTO YELLOW SOLID
  Csonne
 END
 READ NUMBER 'Zahl der Umlaeufe des inneren Planeten?' Nmax
 LET Nu 1
 WHILE (Nu <= Nmax)
```

```
  LET Alpha 0
  WHILE (Alpha < 360)
   LET Cplanet1 (Csonne + (PNT_RA 60 Alpha))
   LET Cplanet2 (Csonne + (PNT_RA 80 (2 * Alpha)))
   CIRCLE GREEN SOLID
    Cplanet1 5
   CIRCLE
    Cplanet2 5
   HATCH AUTO GREEN SOLID
    Cplanet1
    Cplanet2
   END
   LET Rplanet1 (Csonne + (PNT_RA 55 Alpha))
   LET Rplanet2 (Csonne + (PNT_RA 75 (0.63 * Alpha)))
   DELETE
    Rplanet1 Rplanet2
   END
   LET Alpha (Alpha + 5)
  END_WHILE
 LET Nu (Nu + 1)
 END_WHILE
 LINE WHITE SOLID
 END
 HATCH_COLOR YELLOW
 HATCH_DIST 3.5
END_DEFINE
```

Aufgabe 10_3 c

```
DEFINE Planet_c
{Der erste Planet wird bei seinen Sonnenumlaeufen}
{von einem zweiten Planeten umkreist}
{Fischer, 07.03.91, Stand 27.07.92}

 LOCAL Csonne    LOCAL Cplanet1  LOCAL Cplanet2
 LOCAL Rplanet1  LOCAL Rplanet2  LOCAL Nerde
 LOCAL Nmond     LOCAL Loesch    LOCAL Delta
 LOCAL Nu        LOCAL Alpha1

 DELETE ALL CONFIRM
 READ PNT 'Sonnenmitte?' Csonne
 CIRCLE YELLOW SOLID
  Csonne 20
 END
 HATCH_DIST 0.1
 HATCH AUTO YELLOW SOLID
  Csonne
 END
 READ NUMBER 'Zahl der Erdumlaeufe?' Nerde
 READ NUMBER 'Zahl der Mondumlaeufe pro Erdumlauf?' Nmond
 READ STRING 'Bahnen loeschen (j/n)' DEFAULT 'j' Loesch
 LET Loesch (UPC (Loesch))
 LET Delta 2
 LET Nu 1
 WHILE (Nu <= Nerde)
  LET Alpha1 0
  WHILE (Alpha1 < 360)
   LET Cplanet1 (Csonne + (PNT_RA 60 Alpha1))
   LET Cplanet2 (Cplanet1 + (PNT_RA 20 (Nmond * Alpha1)))
   CIRCLE CYAN SOLID
    Cplanet1 5
   CIRCLE GREEN SOLID
    Cplanet2 2.5
   END
   IF (Loesch = 'J')
    LET Rplanet1 (Csonne + (PNT_RA 55 Alpha1))
    LET Rplanet2 (Cplanet1 + (PNT_RA 17.5 (Nmond * Alpha1)))
    DELETE Rplanet1
```

```
    DELETE Rplanet2
   END_IF
   END
   LET Alpha1 (Alpha1 + Delta)
  END_WHILE
  LET Nu (Nu + 1)
 END_WHILE
 LINE WHITE SOLID
 END
 HATCH_DIST 3.5
END_DEFINE
```

Aufgabe 11_1

```
DEFINE Schelle6
{Zeichnen einer Schelle}
{Lesen von abhängigen Maßen aus Datei "SCHELLEN.DAT}
{Fischer, 01.03.91, Stand 01.11.91}

LOCAL Lg    LOCAL Di    LOCAL Da    LOCAL Hf    LOCAL Dg
LOCAL Dk    LOCAL Dd    LOCAL Ds    LOCAL Ts    LOCAL Lf
LOCAL Sm    LOCAL P0    LOCAL Pr    LOCAL P1    LOCAL P2
LOCAL P3    LOCAL P4    LOCAL P5    LOCAL P6    LOCAL P7
LOCAL P8    LOCAL P9    LOCAL P10   LOCAL P11   LOCAL P12
LOCAL P13   LOCAL P14   LOCAL P15   LOCAL P16a  LOCAL P16i
LOCAL P17a  LOCAL P17i  LOCAL P18a  LOCAL P18i  LOCAL P19a
LOCAL P19i  LOCAL P20   LOCAL P21   LOCAL P22   LOCAL P23
LOCAL El    LOCAL Eq    LOCAL Treffer  LOCAL Inzeile

LOOP {Schleife 0: Wiederholtes Ausführen des Makros}

{Benutzereingaben}
READ PNT 'Mitte Bohrung?' P0
READ PNT 'Richtung der Klemmfuge?' RUBBER_LINE P0 Pr

LOOP {Schleife 1: Lesen der Abmessungen aus Datei}
 {Öffnen der Abmessungsdatei und Überlesen der ersten 3 Zeilen}
 OPEN_INFILE 1 'schellen.dat'
 READ_FILE 1 Inzeile
 READ_FILE 1 Inzeile
 READ_FILE 1 Inzeile

 {Eingabe des Innendurchmessers, Prüfen der Zulässigkeit und}
 {Suche der zugehörigen Daten in Abmessungsdatei}
 READ NUMBER 'Innendurchmesser?' Di
 LET Treffer FALSE
 REPEAT
  READ_FILE 1 Inzeile
  IF ((VAL (SUBSTR Inzeile 1 5)) = Di)
   LET Treffer TRUE
   LET Da (VAL (SUBSTR Inzeile 7 5))
   LET Lg (VAL (SUBSTR Inzeile 13 5))
   LET Hf (VAL (SUBSTR Inzeile 19 5))
   LET Dg (VAL (SUBSTR Inzeile 25 5))
   LET Dk (VAL (SUBSTR Inzeile 31 5))
   LET Dd (VAL (SUBSTR Inzeile 37 5))
   LET Ds (VAL (SUBSTR Inzeile 43 5))
   LET Ts (VAL (SUBSTR Inzeile 49 5))
  END_IF
 UNTIL ((Treffer = TRUE) OR (Inzeile = 'END-OF-FILE'))
 CLOSE_FILE 1
 EXIT_IF (Treffer = TRUE)
 BEEP
 DISPLAY ('Innendurchmesser unzulässig, Neueingabe oder Abbruch')
END_LOOP {Ende Schleife 1}

{Berechnung der Einheitsvektoren längs und quer}
LET El ((Pr - P0) / (LEN (Pr - P0)))
```

```
LET Eq (ROT El 90)

{Berechnung der Konturpunkte}
LET P1 (P0 + El * Lg)
LET P2 (P0 + (ROT (El * Di / 2) (ARCSIN (Hf / Di))))
LET P3 (P0 - El * Di / 2)
LET P4 (P0 + (ROT (El * Di / 2) (-ARCSIN (Hf / Di))))
LET P5 (P1 - Eq * Hf / 2)
LET P6 (P1 - Eq * Da / 2)
LET P7 (P0 - Eq * Da / 2)
LET P8 (P0 - El * Da / 2)
LET P9 (P0 + Eq * Da / 2)
LET P10 (P1 + Eq * Da / 2)
LET P11 (P1 + Eq * Hf / 2)
LET Lf (LEN (P1 - P0) - Di / 2)
LET Sm (Lf / 2 + Di / 2)
LET P12 (P9 + El * (Sm - Ds / 2))
LET P13 (P12 - Eq * Ts)
LET P14 (P13 + El * ((Ds - Dd) / 2))
LET P15 (P14 - Eq * ((Da - Hf) / 2 - Ts))
LET P16a (P0 - Eq * Hf / 2 + El * (Sm - Dg / 2))
LET P16i (P0 - Eq * Hf / 2 + El * (Sm - Dk / 2))
LET P17a (P0 - Eq * Da / 2 + El * (Sm - Dg / 2))
LET P17i (P0 - Eq * Da / 2 + El * (Sm - Dk / 2))
LET P18a (P17a + El * Dg)
LET P18i (P17i + El * Dk)
LET P19a (P16a + El * Dg)
LET P19i (P16i + El * Dk)
LET P20 (P15 + El * Dd)
LET P21 (P14 + El * Dd)
LET P22 (P13 + El * Ds)
LET P23 (P12 + El * Ds)

 {Anlegen eines Teils mit der DIN-Bezeichnung}
 INIT_PART ('Schelle ' + STR Di + ' x ' + STR Lg)

{Zeichnen der Kontur}
SPLITTING ON
ARC THREE_PTS WHITE SOLID
 P4 P2 P3 P7 P9 P8
LINE POLYGON
 P4 P5 P6 P7
LINE POLYGON
 P9 P10 P11 P2
LINE POLYGON
 P12 P13 P22 P23
LINE
 P14 P15 P21 P20 P16i P17i P19i P18i
END

{Zeichnen der Bruchlinien}
Bruch (0.35 * P10 + 0.65 * P23) (0.5 * P11 + 0.5 * P20)
Bruch (0.35 * P5 + 0.65 * P19a) (0.35 * P6 + 0.65 * P18a)
Bruch (P12 - 0.35 * (P10 - P23)) (P15 - 0.5 * (P11 - P20))
Bruch (P16a - 0.35 * (P5 - P19a)) (P17a - 0.35 * (P6 - P18a))

{Schraffieren des Aufbruchs}
HATCH_DIST 2.5
HATCH_REF_PT P12
HATCH_ANGLE (ANG El + 45)
HATCH AUTO YELLOW SOLID
 (P12 - El * 0.5 - Eq * 0.5) (P23 + El * 0.5 - Eq * 0.5)
 (P16i - El * 0.5 - Eq * 0.5) (P19i + El * 0.5 - Eq * 0.5)
END

{Zeichnen der Gewinde-Außenlinien}
LINE YELLOW SOLID
 P16a P17a P19a P18a

{Zeichnen der Mittellinien}
LINE YELLOW DOT_CENTER
 (P8 - 2 * El) (P1 + 2 * El) (P7 - 2 * Eq) (P9 + 2 * Eq)
```

```
 ((P17a + P18a) / 2 - 2 * Eq) ((P12 + P23) / 2 + 2 * Eq)
 WHITE SOLID
END

HATCH_ANGLE 45

END_LOOP  {Ende Schleife 0}

END_DEFINE
```

Aufgabe 12_1

```
DEFINE Hw_file
{Biegelinie einer Hohlwelle mit Fest- Loslagerung}
{Berechnung der Durchbiegung in C-Programm "hohlwell"}
{Datenaustausch über Plattendateien}
{Fischer, 02.10.91, Stand 21.10.91}

LOCAL Da       {Außendurchmesser}
LOCAL Di       {Innendurchmesser}
LOCAL Emodul   {E-Modul}
LOCAL Lg       {Länge}
LOCAL Kraft    {Kraft}
LOCAL Af       {Kraftangriffspunkt}
LOCAL Nz       {Anzahl der Zwischenpunkte}
LOCAL I        {Zählindex}
LOCAL Zeile    {Gelesener Datensatz}
LOCAL I_traeg  {Flächenträgheitsmoment}
LOCAL W_kraft  {Durchbiegung am Kraftangriffspunkt}
LOCAL X_max    {Ort der max.Durchbiegung}
LOCAL W_max    {Max.Durchbiegung}
LOCAL X_wert   {X-Wert für Durchbiegung}
LOCAL W_wert   {Durchbiegung bei X = X-Wert}
LOCAL X_faktor {Skalierungsfaktor Länge}
LOCAL W_faktor {Skalierungsfaktor Durchbiegung}
LOCAL Nochmal  {Steuervariable für Eingabe-Kontrollschleifen}

{Laden der Prinzipskizze}
LOAD 'hohlwell.mi'
CS_SET THREE_PTS ABSOLUTE 0,0 ABSOLUTE 1,0 ABSOLUTE 0,1
CS_REF_PT ABSOLUTE -60,30
ORIGIN ON

LOOP {Eingabekontrolle}
 LET Nochmal FALSE
 READ NUMBER 'Außendurchmesser [mm] ?' Da
 READ NUMBER 'Innendurchmesser [mm] ?' Di
 IF (Di >= Da)
  LET Nochmal TRUE
  BEEP
  DISPLAY 'Di >= Da unzulässig!'
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP

READ NUMBER 'E-Modul [N/mm**2] ?' DEFAULT 210000 Emodul

LOOP {Eingabekontrolle}
 LET Nochmal FALSE
 READ NUMBER 'Lagerabstand [mm] ?' Lg
 IF (Lg <= 0)
  LET Nochmal TRUE
  BEEP
  DISPLAY 'Lagerabstand muß größer als "0" sein!'
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP
```

```
READ NUMBER 'Kraft [N] ?' Kraft

LOOP {Eingabekontrolle}
 LET Nochmal FALSE
 READ NUMBER 'Kraftangriff bei x = [mm] ?' Af
 IF ((Af < 0) OR (Af > Lg))
  LET Nochmal TRUE
  BEEP
  DISPLAY '0 < Lagerabstand <= Länge erforderlich!'
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP

LET Nz 24 {Anzahl der Zwischenpunkte}

{Öffnen der Ausgabedatei und Eintragen der Wellendaten}
OPEN_OUTFILE 6 DEL_OLD 'wellout.dat'
WRITE_FILE 6 Da
WRITE_FILE 6 Di
WRITE_FILE 6 Emodul
WRITE_FILE 6 Lg
WRITE_FILE 6 Kraft
WRITE_FILE 6 Af
WRITE_FILE 6 Nz
CLOSE_FILE 6

{Aufruf des C-Programm mit Umadressierung}
RUN GRAPHIC 'hohlwell < wellout.dat > wellin.dat'

{Öffnen der Eingabedatei}
OPEN_INFILE 5 'wellin.dat'

LET I 0

{Überlesen der ersten 13 Datensätze}
REPEAT
 LET I (I + 1)
 READ_FILE 5 Zeile
UNTIL (I >= 13)

READ_FILE 5 Zeile
LET I_traeg (VAL (SUBSTR Zeile 45 (LEN Zeile)))
READ_FILE 5 Zeile
LET W_kraft (VAL (SUBSTR Zeile 45 (LEN Zeile)))
READ_FILE 5 Zeile
LET X_max   (VAL (SUBSTR Zeile 45 (LEN Zeile)))
READ_FILE 5 Zeile
LET W_max   (VAL (SUBSTR Zeile 45 (LEN Zeile)))
LET X_faktor (200 / Lg)
IF (W_max = 0)
 LET W_faktor 1
ELSE
 LET W_faktor (25 / W_max)
END_IF

{Einzeichnen der Kraft}
LEADER_LINE
 RED
 (PNT_XY (X_faktor * Af) 25)
 (PNT_XY (X_faktor * Af) 0)
END
TEXT_SIZE 3.5
TEXT
 RED
 ('F = ' + (STR Kraft) + ' N')
 (PNT_XY (X_faktor * Af) 35)
 ('bei X = ' + (STR Af) + ' mm')
 (PNT_XY (X_faktor * Af) 30)

{Einzeichnen der max.Durchbiegung}
IF (W_max > 0)
 LINE YELLOW DASHED
```

```
  (PNT_XY (X_faktor * X_max) 0)
  (PNT_XY (X_faktor * X_max) -25)
 TEXT
  YELLOW
  ('W(max) = ' + (STR W_max) + ' mm')
  (PNT_XY (X_faktor * X_max) -35)
  ('bei X = ' + (STR X_max) + ' mm')
  (PNT_XY (X_faktor * X_max) -40)
  GREEN
 END
END_IF

{Überlesen der nächsten 5 Datensätze}
LET I 0
REPEAT
 LET I (I + 1)
 READ_FILE 5 Zeile
UNTIL (I >= 5)

{Zeichnen der Biegelinie}
LET I 0
SPLINE
WHITE SOLID
REPEAT
 LET I (I + 1)
 READ_FILE 5 Zeile
 LET X_wert (X_faktor * (VAL (SUBSTR Zeile 1 8)))
 LET W_wert (- W_faktor * (VAL (SUBSTR Zeile 13 6)))
 (PNT_XY X_wert W_wert)
UNTIL (I >= (Nz + 2))
CLOSE_FILE 5

{Löschen der Dateien}
PURGE_FILE 'wellout.dat' CONFIRM
PURGE_FILE 'wellin.dat' CONFIRM

END
END
END_DEFINE
```

Aufgabe 12_2

```
DEFINE Hw_pipe
{Biegelinie einer Hohlwelle mit Fest- Loslagerung}
{Berechnung der Durchbiegung in C-Programm "hohlwell"}
{Datenaustausch über benannte Pipes}
{Fischer, 02.10.91, Stand 21.10.91}

LOCAL Da       {Außendurchmesser}
LOCAL Di       {Innendurchmesser}
LOCAL Emodul   {E-Modul}
LOCAL Lg       {Länge}
LOCAL Kraft    {Kraft}
LOCAL Af       {Kraftangriffspunkt}
LOCAL Nz       {Anzahl der Zwischenpunkte}
LOCAL I        {Zählindex}
LOCAL Zeile    {Gelesener Datensatz}
LOCAL I_traeg  {Flächenträgheitsmoment}
LOCAL W_kraft  {Durchbiegung am Kraftangriffspunkt}
LOCAL X_max    {Ort der max.Durchbiegung}
LOCAL W_max    {Max.Durchbiegung}
LOCAL X_wert   {X-Wert für Durchbiegung}
LOCAL W_wert   {Durchbiegung bei X = X-Wert}
LOCAL X_faktor {Skalierungsfaktor Länge}
LOCAL W_faktor {Skalierungsfaktor Durchbiegung}
LOCAL Nochmal  {Steuervariable für Eingabe-Kontrollschleifen}

{Laden der Prinzipskizze}
LOAD 'hohlwell.mi'
CS_SET THREE_PTS ABSOLUTE 0,0 ABSOLUTE 1,0 ABSOLUTE 0,1
CS_REF_PT ABSOLUTE -60,30
ORIGIN ON

LOOP {Eingabekontrolle}
 LET Nochmal FALSE
 READ NUMBER 'Außendurchmesser [mm] ?' Da
 READ NUMBER 'Innendurchmesser [mm] ?' Di
 IF (Di >= Da)
  LET Nochmal TRUE
  BEEP
  DISPLAY 'Di >= Da unzulässig!'
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP

READ NUMBER 'E-Modul [N/mm**2] ?' DEFAULT 210000 Emodul

LOOP {Eingabekontrolle}
 LET Nochmal FALSE
 READ NUMBER 'Lagerabstand [mm] ?' Lg
 IF (Lg <= 0)
  LET Nochmal TRUE
  BEEP
  DISPLAY 'Lagerabstand muß größer als "0" sein!'
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP

READ NUMBER 'Kraft [N] ?' Kraft

LOOP {Eingabekontrolle}
 LET Nochmal FALSE
 READ NUMBER 'Kraftangriff bei x = [mm] ?' Af
 IF ((Af < 0) OR (Af > Lg))
  LET Nochmal TRUE
  BEEP
  DISPLAY '0 < Lagerabstand <= Länge erforderlich!'
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP
```

```
LET Nz 24 {Anzahl der Zwischenpunkte}

{Einrichten zweier benannter Pipes}
RUN GRAPHIC '/etc/mknod out_pipe p'
RUN GRAPHIC '/etc/mknod in_pipe p'

{Starten des C-Programm als Hintergrundprozeß mit Umadressierung}
RUN GRAPHIC 'hohlwell < out_pipe > in_pipe &'

{Öffnen der Ausgabepipe und Schreiben der Wellendaten}
OPEN_OUTFILE 6 'out_pipe'
WRITE_FILE 6 Da
WRITE_FILE 6 Di
WRITE_FILE 6 Emodul
WRITE_FILE 6 Lg
WRITE_FILE 6 Kraft
WRITE_FILE 6 Af
WRITE_FILE 6 Nz
CLOSE_FILE 6

{Öffnen der Eingabe-Pipe}
OPEN_INFILE 5 'in_pipe'

LET I 0

{Überlesen der ersten 13 Datensätze}
REPEAT
 LET I (I + 1)
 READ_FILE 5 Zeile
UNTIL (I >= 13)

READ_FILE 5 Zeile
LET I_traeg (VAL (SUBSTR Zeile 45 (LEN Zeile)))
READ_FILE 5 Zeile
LET W_kraft (VAL (SUBSTR Zeile 45 (LEN Zeile)))
READ_FILE 5 Zeile
LET X_max   (VAL (SUBSTR Zeile 45 (LEN Zeile)))
READ_FILE 5 Zeile
LET W_max   (VAL (SUBSTR Zeile 45 (LEN Zeile)))
LET X_faktor (200 / Lg)
IF (W_max = 0)
 LET W_faktor 1
ELSE
 LET W_faktor (25 / W_max)
END_IF

{Einzeichnen der Kraft}
LEADER_LINE
 RED
 (PNT_XY (X_faktor * Af) 25)
 (PNT_XY (X_faktor * Af) 0)
END
TEXT_SIZE 3.5
TEXT
 RED
 ('F = ' + (STR Kraft) + ' N')
 (PNT_XY (X_faktor * Af) 35)
 ('bei X = ' + (STR Af) + ' mm')
 (PNT_XY (X_faktor * Af) 30)

{Einzeichnen der max.Durchbiegung}
IF (W_max > 0)
 LINE YELLOW DASHED
  (PNT_XY (X_faktor * X_max) 0)
  (PNT_XY (X_faktor * X_max) -25)
 TEXT
  YELLOW
  ('W(max) = ' + (STR W_max) + ' mm')
  (PNT_XY (X_faktor * X_max) -35)
  ('bei X = ' + (STR X_max) + ' mm')
  (PNT_XY (X_faktor * X_max) -40)
  GREEN
```

```
 END
END_IF

{Überlesen der nächsten 5 Datensätze}
LET I 0
REPEAT
 LET I (I + 1)
 READ_FILE 5 Zeile
UNTIL (I >= 5)

{Zeichnen der Biegelinie}
LET I 0
SPLINE
WHITE SOLID
REPEAT
 LET I (I + 1)
 READ_FILE 5 Zeile
 LET X_wert (X_faktor * (VAL (SUBSTR Zeile 1 8)))
 LET W_wert (- W_faktor * (VAL (SUBSTR Zeile 13 6)))
 (PNT_XY X_wert W_wert)
UNTIL (I >= (Nz + 2))
CLOSE_FILE 5

{Löschen der benannten Pipes}
RUN GRAPHIC 'rm out_pipe'
RUN GRAPHIC 'rm in_pipe'

END
END
END_DEFINE
```

Aufgabe 13_1

```
DEFINE Gold
{Unterteilen einer Strecke nach dem goldenen Schnitt}
{Fischer, 20.09.91, Stand 15.11.96}

 LOCAL Vh  {Teilungsverhaeltnis nach goldenem Schnitt}
 LOCAL Px  {Identifizierungspunkt}
 LOCAL P0  {Linienanfangspunkt}
 LOCAL P1  {Linienendpunkt}
 LOCAL Pu  {Unterteilungspunkt}
 LOCAL Pg  {Endpunkt großes Teilstueck}
 LOCAL Pk  {Endpunkt kleines Teilstueck}

 {Berechnung des Unterteilungsverhaeltnisses}
 LET Vh ((SQRT 5 - 1) / 2)

 {Schleife zur wiederholten Makroausfuehrung}
 LOOP

  LOOP
   {Bestimmung der zu teilenden Linie}
   READ PNT
    'Bitte Linie in der Nähe des Unterteilungspunktes antippen' Px
   INQ_ELEM Px
  EXIT_IF ((INQ 403) = LINE)
   BEEP DISPLAY 'Keine Linie gefunden'
  END_LOOP

  {Abfrage der Elementdaten}
  LET P0 (INQ 101)
  LET P1 (INQ 102)

  {Bestimmung der Endpunkte der großen und kleinen Teilstuecke}
  IF (LEN (Px - P0) > LEN (Px - P1))
   LET Pg P0
   LET Pk P1
```

```
  ELSE
   LET Pg P1
   LET Pk P0
  END_IF
  {Berechnung des Unterteilungspunkts}
  LET Pu (Pg + Vh * (Pk - Pg))

  {Hilfsgeometrielinie am Unterteilungspunkt, normal zur Linie}
  C_LINE PERPENDICULAR Pu Pu

  {Trennen der Linie am Unterteilungspunkt}
  SPLIT Pu Pu END

 END_LOOP

END_DEFINE
```

Aufgabe 13_2

```
DEFINE Mittelkreuz
{Zeichnen von Mittelkreuzen in Kreise, Ueberstand 2 mm}
{Eingestellte Systemwerte bleiben erhalten}
{Fischer, 11.03.91, Stand 18.09.91}

 LOCAL Farbe  LOCAL Linienart  LOCAL Pp  LOCAL Rad
 LOCAL Mb     LOCAL Vk         LOCAL P1
 LOCAL P2     LOCAL P3         LOCAL P4

 {Abfragen und Speichern der Systemwerte fuer Farbe und Linienart}
 INQ_ENV 3
 LET Farbe (INQ 201)
 LET Linienart (INQ 301)
 LOOP
  READ PNT 'Bitte Bohrung identifizieren' Pp
  {Abfrage der Elementdaten}
  INQ_ELEM Pp
  IF ((INQ 403) = CIRCLE)
   LET Rad (INQ 3)
   LET Mb (INQ 101)
   LET Vk (PNT_XY (Rad + 2) 0)
   LET P1 (Mb - Vk)
   LET P2 (Mb + Vk)
   LET P3 (Mb + ROT Vk 90)
   LET P4 (Mb - ROT Vk 90)
   LINE YELLOW DOT_CENTER
    P1 P2 P3 P4
    {Einstellen der gespeicherten Systemwerte}
    RGB_COLOR Farbe LINEPATTERN Linienart
   END
  ELSE
   BEEP
   DISPLAY 'Identifiziertes Element ist kein Kreis'
  END_IF
 END_LOOP
END_DEFINE
```

Aufgabe 13_3

```
DEFINE Trimlinie
{Trimmen von Linien}
{Fischer, 09.09.91, Stand 27.08.97}

LOCAL Sys_farbe       {Systemeinstellung Geometrie-Farbe}
LOCAL Sys_linienart   {Systemeinstellung Geometrie-Linienart}
LOCAL Elem_farbe      {Trimmelement-Farbe}
```

```
LOCAL Elem_linienart  {Trimmelement-Linienart}
LOCAL Px              {Indentifizierungspunkt}
LOCAL P0              {Anfangspunkt der zu trimmenden Linie}
LOCAL P1              {Endpunkt der zu trimmenden Linie}
LOCAL Phi             {Linienwinkel}
LOCAL Pfix            {Festes Ende der zu trimmenden Linie}
LOCAL Ptrim           {Getrimmtes Ende der zu trimmenden Linie}
LOCAL Nochmal         {Steuervariable fuer innere Schleifen}

{Abfrage und Speichern der aktuellen Systemeinstellungen}
{von Geometrie-Farbe und Geometrie-Linienart}
INQ_ENV 3
LET Sys_farbe (INQ 201)
LET Sys_linienart (INQ 301)

{Schleife zur Wiederholung des Makros}
LOOP

LOOP
{Bestimmung der zu trimmenden Linie}
READ PNT 'Zu trimmende Linie?' Px
{Bestimmung der Elementdaten}
INQ_ELEM Px
IF ((INQ 403) = LINE)
 LET Nochmal FALSE
 LET P0  (INQ 101)
 LET P1  (INQ 102)
 LET Elem_farbe (INQ 201)
 LET Elem_linienart (INQ 301)
ELSE
 {Bei Fehler Meldung und rekursiver Makroaufruf}
 LET Nochmal TRUE
 BEEP
 DISPLAY 'Keine Linie gefunden'
END_IF
EXIT_IF (NOT Nochmal)
END_LOOP

{Berechnung des Linienwinkels}
LET Phi (ANG (P1 - P0))
{Bestimmung des festen Endes der zu trimmenden Linie}
IF (LEN (Px - P0) > LEN (Px - P1))
 LET Pfix P0
ELSE
 LET Pfix P1
END_IF

{Loeschen der urspruenglichen Linie}
DELETE Px

{Bestimmung des Trimmpunkts}
LOOP
 LET Nochmal FALSE
 READ PNT 'Linienende?' RUBBER_LINE_ANG Pfix Phi Ptrim
 {Wiederholung, wenn Endpunkt = Anfangspunkt}
 IF (Ptrim = Pfix)
  BEEP
  DISPLAY 'Trimmen auf Laenge = 0 nicht moeglich'
  LET Nochmal TRUE
 END_IF
 EXIT_IF (NOT Nochmal)
END_LOOP

{Zeichnen einer Linie vom festen Ende zum Trimmpunkt}
LINE PT_ANG_DIST
 RGB_COLOR Elem_farbe LINEPATTERN Elem_linienart Pfix Phi Ptrim
 {Ruecksetzen von Farbe und Linienart auf alte Systemwerte}
 RGB_COLOR Sys_farbe LINEPATTERN Sys_linienart
END

END_LOOP
END_DEFINE
```

Aufgabe 13_5

```
DEFINE Trim
{Trimmen von Linien und Kreisboegen}
{Fischer, 09.09.91, Stand 15.11.96}

LOCAL Sys_farbe       {Systemeinstellung Geometrie-Farbe}
LOCAL Sys_linienart   {Systemeinstellung Geometrie-Linienart}
LOCAL Art_a           {Elementart Trimmelements}
LOCAL Art_b           {Elementart des Zielelements}
LOCAL Elem_farbe      {Trimmelement-Farbe}
LOCAL Elem_linienart  {Trimmelement-Linienart}
LOCAL Px              {Indentifizierungspunkt}
LOCAL P0_a            {Anfangspunkt des Trimmelements}
LOCAL P1_a            {Endpunkt des Trimmelements}
LOCAL P0_b            {Anfangspunkt des Zielelements}
LOCAL P1_b            {Endpunkt des Zielelements}
LOCAL X0_a            {X-Koordinate P0_a}
LOCAL Y0_a            {Y-Koordinate P0_a}
LOCAL X1_a            {X-Koordinate P1_a}
LOCAL Y1_a            {Y-Koordinate P1_a}
LOCAL X0_b            {X-Koordinate P0_b}
LOCAL Y0_b            {Y-Koordinate P0_b}
LOCAL X1_b            {X-Koordinate P1_b}
LOCAL Y1_b            {Y-Koordinate P1_b}
LOCAL Xs              {X-Koordinate Schnittpunkt Trimmelement #
Zielelement}
LOCAL Ys              {Y-Koordinate Schnittpunkt Trimmelement #
Zielelement}
LOCAL Phi_a           {Linienwinkel des Trimmelements}
LOCAL Mp_a            {Bogenmittelpunkt des zu trimmenden Bogens}
LOCAL Rad_a           {Bogenradius des zu trimmenden Bogens}
LOCAL Pfix_a          {Festes Ende des zu trimmenden Elements}
LOCAL Ptrim_a         {Getrimmtes Ende des zu trimmenden Elements}
LOCAL Ganzweg         {Steuervariable fuer vollstaendiges Loeschen
des Trimmelements}

{Abfrage und Speichern der aktuellen Systemeinstellungen}
{von Geometrie-Farbe und Geometrie-Linienart}
INQ_ENV 3
LET Sys_farbe (INQ 201)
LET Sys_linienart (INQ 301)

{Schleife zur Wiederholung des Makros}
LOOP
LET Ganzweg FALSE

LOOP
 {Bestimmung des zu trimmenden Elements}
 READ PNT 'Zu trimmendes Element?' Px
 {Bestimmung der Elementdaten}
 INQ_ELEM Px
 LET Art_a (INQ 403)
 IF (Art_a = LINE)
  LET P0_a  (INQ 101)
  LET P1_a  (INQ 102)
  LET Phi_a (ANG (P1_a - P0_a))
 ELSE_IF (Art_a = ARC)
  LET Rad_a (INQ 3)
  LET Mp_a (INQ 101)
  LET P0_a (INQ 102)
  LET P1_a (INQ 103)
 END_IF
 EXIT_IF ((Art_a = LINE) OR (Art_a = ARC))
  BEEP DISPLAY 'Keine trimmbares Element gefunden'
END_LOOP

LET Elem_farbe (INQ 201)
LET Elem_linienart (INQ 301)

{Bestimmung des festen Endes des zu trimmenden Elements}
```

```
IF (LEN (Px - P0_a) > LEN (Px - P1_a))
 LET Pfix_a P0_a
ELSE
 LET Pfix_a P1_a
END_IF

{Loeschen des urspruenglichen Elements}
DELETE Px

{Bestimmung des Trimmpunkts fuer Trimmelement = LINE}
IF (Art_a = LINE)
 READ PNT 'Zielpunkt?' RUBBER_LINE_ANG Pfix_a Phi_a Ptrim_a
 INQ_ELEM Ptrim_a
 LET Art_b (INQ 403)

 {Berechnung eines Schnittpunkts, wenn Zielement = LINE}
 IF (Art_b = LINE)
  LET P0_b (INQ 101)
  LET P1_b (INQ 102)
  LET X0_a (X_OF P0_a)
  LET Y0_a (Y_OF P0_a)
  LET X1_a (X_OF P1_a)
  LET Y1_a (Y_OF P1_a)
  LET X0_b (X_OF P0_b)
  LET Y0_b (Y_OF P0_b)
  LET X1_b (X_OF P1_b)
  LET Y1_b (Y_OF P1_b)
  {Pruefen auf Trimmelement senkrecht und Zielelement <> senkr.}
  IF ((ABS (X1_a-X0_a) < 0.00001) AND (ABS (X1_b-X0_b) > 0.00001))
   LET M_b ((Y1_b - Y0_b) / (X1_b - X0_b))
   LET Xs X0_a
   LET Ys (Y0_b + M_b * (X0_a - X0_b))
   LET Ptrim_a (PNT_XY Xs Ys)
  {Pruefen auf Trimmelement <> senkr. und Zielelement senkr.}
  ELSE_IF
  ((ABS (X1_a-X0_a) > 0.00001) AND (ABS (X1_b-X0_b) < 0.00001))
   LET M_a ((Y1_a - Y0_a) / (X1_a - X0_a))
   LET Xs X0_b
   LET Ys (Y0_a + M_a * (Xs - X0_a))
   LET Ptrim_a (PNT_XY Xs Ys)
  {Pruefen auf Trimmelement <> senkr. und Zielelement <> senkr.}
  ELSE_IF
  ((ABS (X1_a-X0_a) > 0.00001) AND (ABS (X1_b-X0_b) > 0.00001))
   LET M_a ((Y1_a - Y0_a) / (X1_a - X0_a))
   LET M_b ((Y1_b - Y0_b) / (X1_b - X0_b))
   {Pruefen auf Trimmelement und Zielelement <> parallel}
   IF (ABS (M_a - M_b) > 0.00001)
    LET Xs ((Y0_a - M_a * X0_a - Y0_b + M_b * X0_b) / (M_b - M_a))
    LET Ys (Y0_a - M_a * X0_a + M_a * Xs)
    LET Ptrim_a (PNT_XY Xs Ys)
   END_IF
  END_IF
 END_IF

ELSE_IF (Art_a = ARC)
 {Bei Trimmen des Bogenendpunkts P1_a (->Pfix_a = P0_a) beginnt}
 {RUBBER_ARC bei P0_a, bei Trimmen des Bogenanfangspunkts P0_a}
 {endet RUBBER_ARC bei P1_a}
 IF (Pfix_a = P0_a)
  READ PNT 'Zielpunkt?' RUBBER_ARC_CEN_BEG Mp_a P0_a Ptrim_a
 ELSE
  READ PNT 'Zielpunkt?' RUBBER_ARC_CEN_END Mp_a P1_a Ptrim_a
 END_IF
END_IF
{Vollstaendiges Loeschen, wenn wenn Endpunkt = Anfangspunkt}
IF (ABS (Ptrim_a - Pfix_a) < 0.00001)
 LET Ganzweg TRUE
END_IF

{Zeichnen des getrimmten Elementes, wenn Laenge > 0}
IF (NOT Ganzweg)
{Linie vom festen Ende zum Trimmpunkt}
```

```
 IF (Art_a = LINE)
  LINE PT_ANG_DIST RGB_COLOR Elem_farbe LINEPATTERN Elem_linienart
  Pfix_a Phi_a Ptrim_a
 ELSE_IF (Art_a = ARC)
  {Bogen zwischen Fixpunkt und Trimpunkt}
  {Zeichnen nur in mathem.pos. Drehsinn moeglich!}
  IF (Pfix_a = P0_a)
   ARC CEN_RAD_ANG RGB_COLOR Elem_farbe LINEPATTERN Elem_linienart
   Mp_a Rad_a Pfix_a Ptrim_a
  ELSE
   ARC CEN_RAD_ANG RGB_COLOR Elem_farbe LINEPATTERN Elem_linienart
   Mp_a Rad_a Ptrim_a Pfix_a
  END_IF
 END_IF
END_IF

{Ruecksetzen von Farbe und Linienart auf alte Systemwerte}
LINE RGB_COLOR Sys_farbe LINEPATTERN Sys_linienart END

END_LOOP
END_DEFINE
```

Aufgabe 14_2

```
DEFINE Normzahl
{Anzeigen der Hauptwerte von Normzahlreihen}
{Fischer, 12.09.91, Stand 23.09.96}

 {Prüfen, ob Menü darstellbar ist}
 IF (I_port) Check_i_port END_IF

 IF (NOT I_port)

  CURRENT_MENU 'Normzahlmenu'
  MENU_LAYOUT
   Menu_position RIGHT
   Headline_height    '1| |               | '
   Text_slot_height   '2                    '
   Text_slot_height   '3     |       |      '
   Text_slot_height   '4     |       |      '
   Text_slot_height   '5     |       |      '
   Text_slot_height   '6     |       |      '
   Text_slot_height   '7     |       |      '
   Text_slot_height   '8     |       |      '
   Text_slot_height   '9     |       |      '
   Text_slot_height   '10    |       |      '
   Text_slot_height   '11    |       |      '
   Text_slot_height   '12    |       |      '
   Text_slot_height   '13    |       |      '
   Text_slot_height   '14    |       |      '
   Text_slot_height   '15    |       |      '
   Text_slot_height   '16    |       |      '
   Text_slot_height   '17    |       |      '
   Text_slot_height   '18    |       |      '
   Text_slot_height   '19    |       |      '
   Text_slot_height   '20    |       |      '
   Text_slot_height   '21    |       |      '
   Text_slot_height   '22    |       |      '
   Text_slot_height   '23    |       |      '
   Text_slot_height   '24    |       |      '
   Text_slot_height   '25    |       |      '
   Bottom_slot_height '26  |     |     |    '
   Bottom_slot_height '27  |     |     |    '
   Menu_home_point_top
  END

  {Beschriftung und Belegung der Menüfelder}
  Menu_control_icons {Standardbelegung für erste Zeile}
```

```
  MENU Colo0 Bcol5 CENTER 'NORMZ. DIN 323' '' 1 3
  MENU Colo0 Bcol1 CENTER 'R5' '' 3 1
  MENU Colo0 Bcol1 CENTER 'R10' '' 3 2
  MENU Colo0 Bcol1 CENTER 'R20' '' 3 3
  MENU ' 1,0 ' 'ENTER 1'    4  1
  MENU ' 1,0 ' 'ENTER 1'    4  2
  MENU ' 1,0 ' 'ENTER 1'    4  3
  MENU ' 1,12' 'ENTER 1.12' 5  3
  MENU ' 1,25' 'ENTER 1.25' 6  2
  MENU ' 1,25' 'ENTER 1.25' 6  3
  MENU ' 1,4 ' 'ENTER 1.4'  7  3
  MENU ' 1,6 ' 'ENTER 1.6'  8  1
  MENU ' 1,6 ' 'ENTER 1.6'  8  2
  MENU ' 1,6 ' 'ENTER 1.6'  8  3
  MENU ' 1,8 ' 'ENTER 1.8'  9  3
  MENU ' 2,0 ' 'ENTER 2'    10 2
  MENU ' 2,0 ' 'ENTER 2'    10 3
  MENU ' 2,24' 'ENTER 2.24' 11 3
  MENU ' 2,5 ' 'ENTER 2.5'  12 1
  MENU ' 2,5 ' 'ENTER 2.5'  12 2
  MENU ' 2,5 ' 'ENTER 2.5'  12 3
  MENU ' 2,8 ' 'ENTER 2.8'  13 3
  MENU ' 3,15' 'ENTER 3.15' 14 2
  MENU ' 3,15' 'ENTER 3.15' 14 3
  MENU ' 3,55' 'ENTER 3.55' 15 3
  MENU ' 4,0 ' 'ENTER 4'    16 1
  MENU ' 4,0 ' 'ENTER 4'    16 2
  MENU ' 4,0 ' 'ENTER 4'    16 3
  MENU ' 4,5 ' 'ENTER 4.5 ' 17 3
  MENU ' 5,0 ' 'ENTER 5.0 ' 18 2
  MENU ' 5,0 ' 'ENTER 5.0 ' 18 3
  MENU ' 5,6 ' 'ENTER 5.6 ' 19 3
  MENU ' 6,3 ' 'ENTER 6.3 ' 20 1
  MENU ' 6,3 ' 'ENTER 6.3 ' 20 2
  MENU ' 6,3 ' 'ENTER 6.3 ' 20 3
  MENU ' 7,1 ' 'ENTER 7.1 ' 21 3
  MENU ' 8,0 ' 'ENTER 8.0 ' 22 2
  MENU ' 8,0 ' 'ENTER 8.0 ' 22 3
  MENU ' 9,0 ' 'ENTER 9.0 ' 23 3
  MENU '10,0 ' 'ENTER 10.0' 24 1
  MENU '10,0 ' 'ENTER 10.0' 24 2
  MENU '10,0 ' 'ENTER 10.0' 24 3
  MENU Colo0 Bcol1 '' '' 25 1
  MENU Colo0 Bcol1 '' '' 25 2
  MENU Colo0 Bcol1 '' '' 25 3
  Eight_menu_slots_add

 END_IF

END_DEFINE
```

Aufgabe 15_1

```
DEFINE Make_din1_lt
{Logische Tabelle für Kegelstifte DIN 1}
{Fischer, 18.09.91, Stand 24.09.96}

 CREATE_LTAB 5 17 'Din1_lt'

 WRITE_LTAB 'Din1_lt' TITLE 1 'Kegelstifte nach DIN 1'
 WRITE_LTAB 'Din1_lt' TITLE 2 'ø'
 WRITE_LTAB 'Din1_lt' TITLE 3 'Längen'

 WRITE_LTAB 'Din1_lt' 1 1 4     WRITE_LTAB 'Din1_lt' 1 2 4
 WRITE_LTAB 'Din1_lt' 1 3 16    WRITE_LTAB 'Din1_lt' 1 4 18
 WRITE_LTAB 'Din1_lt' 1 5 20    WRITE_LTAB 'Din1_lt' 1 6 22
 WRITE_LTAB 'Din1_lt' 1 7 24    WRITE_LTAB 'Din1_lt' 1 8 26
 WRITE_LTAB 'Din1_lt' 1 9 28    WRITE_LTAB 'Din1_lt' 1 10 30
```

```
 WRITE_LTAB 'Din1_lt' 1 11 32   WRITE_LTAB 'Din1_lt' 1 12 36
 WRITE_LTAB 'Din1_lt' 1 13 40   WRITE_LTAB 'Din1_lt' 1 14 45
 WRITE_LTAB 'Din1_lt' 1 15 50   WRITE_LTAB 'Din1_lt' 1 16 55
 WRITE_LTAB 'Din1_lt' 1 17 60

 WRITE_LTAB 'Din1_lt' 2 1 5     WRITE_LTAB 'Din1_lt' 2 2 6
 WRITE_LTAB 'Din1_lt' 2 3 20    WRITE_LTAB 'Din1_lt' 2 4 22
 WRITE_LTAB 'Din1_lt' 2 5 24    WRITE_LTAB 'Din1_lt' 2 6 26
 WRITE_LTAB 'Din1_lt' 2 7 28    WRITE_LTAB 'Din1_lt' 2 8 30
 WRITE_LTAB 'Din1_lt' 2 9 32    WRITE_LTAB 'Din1_lt' 2 10 36
 WRITE_LTAB 'Din1_lt' 2 11 40   WRITE_LTAB 'Din1_lt' 2 12 45
 WRITE_LTAB 'Din1_lt' 2 13 50   WRITE_LTAB 'Din1_lt' 2 14 55
 WRITE_LTAB 'Din1_lt' 2 15 60   WRITE_LTAB 'Din1_lt' 2 16 70
 WRITE_LTAB 'Din1_lt' 2 17 0

 WRITE_LTAB 'Din1_lt' 3 1 6     WRITE_LTAB 'Din1_lt' 3 2 6
 WRITE_LTAB 'Din1_lt' 3 3 24    WRITE_LTAB 'Din1_lt' 3 4 26
 WRITE_LTAB 'Din1_lt' 3 5 28    WRITE_LTAB 'Din1_lt' 3 6 30
 WRITE_LTAB 'Din1_lt' 3 7 32    WRITE_LTAB 'Din1_lt' 3 8 36
 WRITE_LTAB 'Din1_lt' 3 9 40    WRITE_LTAB 'Din1_lt' 3 10 45
 WRITE_LTAB 'Din1_lt' 3 11 50   WRITE_LTAB 'Din1_lt' 3 12 55
 WRITE_LTAB 'Din1_lt' 3 13 60   WRITE_LTAB 'Din1_lt' 3 14 70
 WRITE_LTAB 'Din1_lt' 3 15 80   WRITE_LTAB 'Din1_lt' 3 16 90
 WRITE_LTAB 'Din1_lt' 3 17 100

 WRITE_LTAB 'Din1_lt' 4 1 8     WRITE_LTAB 'Din1_lt' 4 2 10
 WRITE_LTAB 'Din1_lt' 4 3 28    WRITE_LTAB 'Din1_lt' 4 4 30
 WRITE_LTAB 'Din1_lt' 4 5 32    WRITE_LTAB 'Din1_lt' 4 6 36
 WRITE_LTAB 'Din1_lt' 4 7 40    WRITE_LTAB 'Din1_lt' 4 8 45
 WRITE_LTAB 'Din1_lt' 4 9 50    WRITE_LTAB 'Din1_lt' 4 10 55
 WRITE_LTAB 'Din1_lt' 4 11 60   WRITE_LTAB 'Din1_lt' 4 12 70
 WRITE_LTAB 'Din1_lt' 4 13 80   WRITE_LTAB 'Din1_lt' 4 14 90
 WRITE_LTAB 'Din1_lt' 4 15 100  WRITE_LTAB 'Din1_lt' 4 16 110
 WRITE_LTAB 'Din1_lt' 4 17 120

 WRITE_LTAB 'Din1_lt' 5 1 10    WRITE_LTAB 'Din1_lt' 5 2 10
 WRITE_LTAB 'Din1_lt' 5 3 32    WRITE_LTAB 'Din1_lt' 5 4 36
 WRITE_LTAB 'Din1_lt' 5 5 40    WRITE_LTAB 'Din1_lt' 5 6 45
 WRITE_LTAB 'Din1_lt' 5 7 50    WRITE_LTAB 'Din1_lt' 5 8 55
 WRITE_LTAB 'Din1_lt' 5 9 60    WRITE_LTAB 'Din1_lt' 5 10 70
 WRITE_LTAB 'Din1_lt' 5 11 80   WRITE_LTAB 'Din1_lt' 5 12 90
 WRITE_LTAB 'Din1_lt' 5 13 100  WRITE_LTAB 'Din1_lt' 5 14 110
 WRITE_LTAB 'Din1_lt' 5 15 120  WRITE_LTAB 'Din1_lt' 5 16 130
 WRITE_LTAB 'Din1_lt' 5 17 140

END_DEFINE

DEFINE Dt_standards
{Standardeinstellungen für Anzeigetabellen}
{Fischer, 24.09.96, Stand 24.09.96        }

INQ_ENV 10

 IF (INQ 6) {MEPELOOK=1}
  DEFINE Dt_border          RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_bg              RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_scrollbar_fg    RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_scrollbar_bg    RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_head_fg         RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_head_bg         RGB_COLOR 0.143 0.286 0.667 END_DEFINE
  DEFINE Dt_title_fg        RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_title_bg0       RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_title_bg1       RGB_COLOR 0.429 0.429 0.667 END_DEFINE
  DEFINE Dt_data_fg         RGB_COLOR 1 1 1             END_DEFINE
  DEFINE Dt_data_bg         RGB_COLOR 0.571 0.571 0.667 END_DEFINE
  DEFINE Dt_frame_with      5       END_DEFINE
  DEFINE Dt_horizontal_col WHITE   END_DEFINE
  DEFINE Dt_horizontal_typ SOLID   END_DEFINE
  DEFINE Dt_vertical_col   WHITE   END_DEFINE
  DEFINE Dt_vertical_typ   SOLID   END_DEFINE
 ELSE {MEPELOOK=0}
```

```
  DEFINE Dt_border         WHITE  END_DEFINE
  DEFINE Dt_bg             BLACK  END_DEFINE
  DEFINE Dt_scrollbar_fg   WHITE  END_DEFINE
  DEFINE Dt_scrollbar_bg   BLUE   END_DEFINE
  DEFINE Dt_head_fg        BLACK  END_DEFINE
  DEFINE Dt_head_bg        YELLOW END_DEFINE
  DEFINE Dt_title_fg       WHITE  END_DEFINE
  DEFINE Dt_title_bg0      BLACK  END_DEFINE
  DEFINE Dt_title_bg1      BLACK  END_DEFINE
  DEFINE Dt_data_fg        WHITE  END_DEFINE
  DEFINE Dt_data_bg        BLACK  END_DEFINE
  DEFINE Dt_frame_with     1      END_DEFINE
  DEFINE Dt_horizontal_col WHITE  END_DEFINE
  DEFINE Dt_horizontal_typ SOLID  END_DEFINE
  DEFINE Dt_vertical_col   WHITE  END_DEFINE
  DEFINE Dt_vertical_typ   SOLID  END_DEFINE
END_IF

END_DEFINE



{Standardeinstellungen für Anzeigetabellen}
Dt_standards



DEFINE Make_din1_dt_1
{Anzeigetabelle Nr.1 für Kegelstifte DIN 1}
{Fischer, 18.09.91, Stand 24.09.96}

 TABLE_LAYOUT 'Din1_dt_1' 'Din1_lt'
  Dt_border Dt_bg
  ROWS 5
  FRAME_WIDTH Dt_frame_width
  HORIZONTAL Dt_horizontal_col Dt_horizontal_typ
  VERTICAL Dt_vertical_col Dt_vertical_typ
  TITLE_LAYOUT
   (Text_slot_height + 1) ' | |                          | '
   (Text_slot_height + 1) '   |                      |    '
                          1 '                           '
  END
  COLUMN_LAYOUT
   (Text_slot_height + 1) '   |    |    |    |    |    |    '
 END

 Table_control_icons 'Din1_dt_1'
 TABLE_TITLE 'Din1_dt_1'
  Dt_head_fg Dt_head_bg '  @s1' '' 1 3
  Dt_title_fg Dt_title_bg1 CENTER '@s2' '' 2 1
  Dt_title_fg Dt_title_Bg1 CENTER 'ENDE'
   'END SHOW_TABLE OFF "Din1_dt_1"' 2 3
  Dt_title_fg Dt_title_bg1 '  @s3' '' 2 2
  Dt_head_fg Dt_head_bg '' '' 3 1
 END

 TABLE_COLUMN 'Din1_dt_1'
  COLUMN 1 Dt_data_fg Dt_title_bg1 3 1  'BEEP'
  COLUMN 2 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 3  '@v1 @v2 @v3'
  COLUMN 3 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 5  '@v1 @v2 @v5'
  COLUMN 4 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 7  '@v1 @v2 @v7'
  COLUMN 5 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 10 '@v1 @v2 @v10'
  COLUMN 6 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 13 '@v1 @v2 @v13'
  COLUMN 7 Dt_data_fg Dt_data_bg RIGHT FORMAT 3 16 '@v1 @v2 @v16'
 END

 MOVE_TABLE 'Din1_dt_1' UPPER LEFT END

END_DEFINE
```

```
DEFINE Make_din1_dt_2
{Anzeigetabelle Nr. 2 für Kegelstifte DIN 1}
{Fischer, 18.09.91, Stand 23.09.96}

 TABLE_LAYOUT 'Din1_dt_2' 'Din1_lt'
  FRAME_WIDTH 2
  TITLE_LAYOUT
   (Text_slot_height + 1) ' | |                                    | '
   (Text_slot_height + 1) '                                          '
   (Text_slot_height + 1) '                                          '
   (Text_slot_height + 1) 'Seitenansicht                             '
   (Text_slot_height + 1) 'Vorderansicht (Kegel zum Betrachter)  '
   (Text_slot_height + 1) 'Rückansicht (Kegel vom Betrachter weg)'
  END
 END

 Table_control_icons 'Din1_dt_2'
 TABLE_TITLE 'Din1_dt_2'
  Dt_head_fg Dt_head_bg '@s1' '' 1 3
  Dt_title_fg Dt_title_bg0 'Seitenansicht' '"S"' 4 1
  Dt_title_fg Dt_title_bg0 'Vorderansicht (Kegel zum Betrachter)' '"V"' 5 1
  Dt_title_fg Dt_title_bg0 'Rückansicht (Kegel vom Betrachter weg)' '"R"' 6 1
 END

END_DEFINE



{Erzeugen der logischen Tabelle}
Make_din1_lt

{Erzeugen der Anzeigetabellen}
Make_din1_dt_1
Make_din1_dt_2



DEFINE Kegelstift_tab
 {Zeichnen eines Kegelstifts nach DIN1}
 {Fischer, 18.09.91, Letzte Änderung: 23.09.96}

 LOCAL Dn LOCAL Dm LOCAL Ls LOCAL Rk LOCAL Nochmal LOCAL H0 LOCAL H1
 LOCAL P0 LOCAL P1 LOCAL P2 LOCAL P3 LOCAL P4 LOCAL P5 LOCAL P6 LOCAL P7

 LOOP
 {Einblenden der 1. Anzeigetabelle}
 SHOW_TABLE ON 'Din1_dt_1'

 READ NUMBER 'Stiftlänge in Zeile mit gewünschentem Durchmesser auswählen' Dn
 READ NUMBER 'Kuppenradius?' Rk
 READ NUMBER 'Länge?' Ls

 {Belegen und Einblenden der 2. Anzeigetabelle}
 TABLE_TITLE 'Din1_dt_2'
  Dt_title_fg Dt_title_bg1 ('Gewählter Durchmesser : ' + (STR Dn))  'BEEP' 2 1
  Dt_title_fg Dt_title_bg1 ('Gewählte Länge        : ' + (STR Ls)) 'BEEP' 3 1
 END
 MOVE_TABLE 'Din1_dt_2' LOWER OF 'Din1_dt_1' 0,0 30,200 END
 SHOW_TABLE ON 'Din1_dt_2'

 {Auswahl der Ansicht}
 READ STRING 'Ansicht?' Ans

 SHOW_TABLE OFF 'Din1_dt_1'
 SHOW_TABLE OFF 'Din1_dt_2'

 {Positionierung des Kegelstifts}
 READ PNT 'Kegelposition?' P0

 {Richtung durch zweiten Punkt oder durch Winkelangabe definiert}
```

```
 READ PNT NUMBER 'Richtung? (Punkt oder Winkel)' RUBBER_LINE P0 P1
 IF (TYPE (P1) = NUMBER)
  LET P1 (P0 + (PNT_RA 1 P1))
 END_IF

 {Berechnung von Einheitsvektoren längs und quer}
 LET Evl ((P1 - P0) / LEN (P1 - P0))
 LET Evq (ROT Evl 90)

 {Berechnung des maximalen Kegeldurchmessers für Kegel 1 : 50}
 LET Dm (Dn + Ls / 50)

 {Anlegen eines Teils mit Namen = DIN-Bezeichnung}
 INIT_PART ('Kegelstift DIN 1 - ' + STR Dn + ' x ' + STR Ls)

 {Zeichnen der Seitenansicht}
 IF (Ans = 'S')
  LET H0 (Rk - 0.5 * SQRT (4 * Rk * Rk - Dm * Dm))
  LET H1 (Rk - 0.5 * SQRT (4 * Rk * Rk - Dm * Dm))
  LET P2 (P0 + Evq * Dm / 2)
  LET P3 (p0 - Evl * H0)
  LET P4 (P0 - Evq * Dm / 2)
  LET P5 (P0 + Evl * Ls - Evq * Dn / 2)
  LET P6 (P0 + Evl * (Ls + H1))
  LET P7 (P0 + Evl * Ls + Evq * Dn / 2)
  LINE POLYGON WHITE SOLID P2 P4 P5 P7 P2
  ARC THREE_PTS P2 P4 P3
                P5 P7 P6
  LINE YELLOW DOT_CENTER (P3 - Evl * 2) (P6 + Evl * 2)
  END
 END_IF

 {Zeichnen der Vorderansicht}
 IF (Ans = 'V')
  CIRCLE WHITE SOLID P0 (Dn / 2) END
  LINE YELLOW DOT_CENTER (P0 - Evl * (Dn / 2 + 2)) (P0 + Evl * (Dn / 2 + 2))
                         (P0 - Evq * (Dn / 2 + 2)) (P0 + Evq * (Dn / 2 + 2))
  END
 END_IF

 {Zeichnen der Rückansicht}
 IF (Ans = 'R')
  CIRCLE WHITE SOLID P0 (Dm / 2)
  LINE YELLOW DOT_CENTER (P0 - Evl * (Dm / 2 + 2))  (P0 + Evl * (Dm / 2 + 2))
                         (P0 - Evq * (Dm / 2 + 2))  (P0 + Evq * (Dm / 2 + 2))
  END
 END_IF
 END_PART

 END_LOOP

END_DEFINE
```

A5 Literatur

/1/ Hewlett-Packard:
Konstruieren mit ME10 (Ausgabe 5 Juni 1995)

/2/ Hewlett-Packard:
Makropragrammierung mit ME10 (Ausgabe 4 Juni 1995)

/3/ Hewlett-Packard:
Konfiguration der ME-Produktfamilie (Ausgabe 2 Juni 1995)

/4/ Fischer, D., Leidig, A.:
Skriptum zum Arbeiten mit ME10.
Fachhochschule Rosenheim, 1990.

/5/ Eigner, M., Maier, H.:
Einstieg in CAD: Lehrbuch für CAD-Anwender.
München, Wien: Carl Hanser Verlag 1985

/6/ IFAO:
CAD-Ausbildung für die Konstruktionspraxis: Teil 1 - CAD 2D.
Herausgegeben und erarbeitet vom Institut für angewandte Organisationsforschung, Karlsruhe. München, Wien: Carl Hanser Verlag 1986.

/7/ IFAO:
CAD-Ausbildung für die Konstruktionspraxis: Teil 2 - CAD 3D.
Herausgegeben und erarbeitet vom Institut für angewandte Organisationsforschung, Karlsruhe. München, Wien: Carl Hanser Verlag 1988.

/8/ Fischer, D.:
Graphische Programmierung mit PC-DRAFT.
Die Programmiersprache des CAD-Systems PC-DRAFT für Befehlserweiterungen, allgemeine Prozeduren und Varianten.
München, Wien: Carl Hanser Verlag 1989

A6 Index