Michael Gehret

Softwareentwicklung am Beispiel einer Dateiverwaltung (HP-41)

Vieweg

Michael Gehret

Softwareentwicklung am Beispiel einer Dateiverwaltung (HP-41)

Bücher zum HP-41

Anwenderhandbuch HP-41 C/CV,
von K.-H. Gosmann

Der HP-41 C in Handwerk und Industrie,
von K. Kraus

Algorithmen der Netzwerkanalyse für programmierbare Taschenrechner (HP-41 C),
von D. Lange

Statistik für programmierbare Taschenrechner (UPN),
von J. Bruhn

Matrix-Steifigkeitsmethode für den HP-41,
von A. Kammerl

Vieweg

Michael Gehret

Softwareentwicklung am Beispiel einer Dateiverwaltung (HP-41)

Friedr. Vieweg & Sohn Braunschweig/Wiesbaden

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Gehret, Michael:
Softwareentwicklung am Beispiel einer Dateiverwaltung (HP-41) / Michael Gehret. – Braunschweig; Wiesbaden: Vieweg, 1984.
ISBN 3-528-04286-9

Druck und buchbinderische Verarbeitung: W. Langelüddecke, Braunschweig
Printed in Germany

ISBN 3-528-04286-9

Vorwort

"Taschencomputer" lautet die neue Bezeichnung Hewlett-Packards für die Reihe HP-41C/CV/CX. Aus dem programmierbaren Taschenrechner wurde ein Computer mit umfangreicher Peripherie (Drucker, Massenspeicher, Monitor, Standard- und spezielle Schnittstellen, Meßgeräte u. a.). Trotz der nicht gerade computergemäßen Programmiersprache und der Speicheradressierung über Registernummern, lassen sich mit dem HP-41-System sehr komplexe Probleme bewältigen. So spricht nichts mehr dagegen, auch bei der Entwicklung von HP-41-Software nach den von größeren Computern bekannten Methoden vorzugehen. Zwei Prinzipien sollen hier betont werden:

- **Modularität:** Ein Modul ist ein Stück Programm (meist ein Unterprogramm), dessen Funktion sowie dessen Eingangs- und Ausgangswerte (die Schnittstelle) exakt definiert sind. Ist ein Modul programmiert und getestet, interessiert die Realisation der Funktion (der Programmtext) nicht mehr, nur noch Funktion und Schnittstelle sind relevant.

- **Schrittweise Verfeinerung (stepwise refinement):** Ein komplexes, abstrakt formuliertes Problem wird hierarchisch in immer kleinere und konkretere Teilaufgaben zerlegt. Diese werden natürlich als Module realisiert.

Das zweite Prinzip spiegelt sich in der Einteilung des Buches wider. **Teil 1** führt in die Problemstellung ein und gibt einen Überblick über seine Lösung. **Teil 2** behandelt die dadurch bedingte Datenstruktur. **Teil 3** beschreibt in Katalogform die einzelnen Module. Die schrittweise Verfeinerung erstreckt sich hier über zwei Stufen: die dem Anwender zugänglichen Funktionen einerseits und die internen Module andererseits. **Teil 4**

und **Teil 5** schließlich behandeln wesentliche Aspekte der Programmoptimierung und der synthetischen Programmierung, soweit sie für ernsthafte Anwendungen relevant sind.

Das Buch erfüllt somit mehrere Funktionen:

- Es enthält ein bewährtes Programmpaket zur komfortablen Verwaltung von Adressenmaterial.

- Der Programmierer kann aus den Modulen der Adressenverwaltung nach einfachen Modifikationen ein System zur Verwaltung eines beliebigen strukturierten Datenbestandes zusammenstellen und optimieren.

- Das Buch zeigt den Einsatz moderner Methoden der Softwareentwicklung mit dem HP-41.

- Schließlich findet der interessierte Leser zwei einfache Verfahren zur Programmoptimierung, zahlreiche Tricks zur Verringerung der Rechenzeit und des Speicherbedarfes sowie interessante Details wie etwa die Speicherformate des Rechners und des IL-Massenspeichers. Außerdem wird ein kurzer, praxisbezogener Zugang zur synthetischen Programmierung vermittelt.

Mein Dank gilt Herrn Gerhard Kruse, Aachen, für die beiden exzellenten Programme Key Assignment und Load Bytes sowie der EDV-COMPAS GmbH, Memmingen, für die Stellung des Textverarbeitungssystems. Schließlich möchte ich den zuständigen Mitarbeitern des Verlages für ihre Offenheit gegenüber der besonderen Form des Buches und die gute Zusammenarbeit danken.

Grönenbach, im Januar 1984 Michael Gehret

Inhaltsverzeichnis

Anmerkung: Zeichen mit ASCII-Codes größer als 127 (dezimal), also Zeichen der unteren Hälfte der Hexcode-Tabelle, sind unterstrichen dargestellt. So ist ASCII (A) = 65 und ASCII (A) = 193.

1 Einführung

1.1 Dateiverwaltung?

Die Verwaltung eines strukturierten Datenbestandes ist eine in der EDV allgegenwärtige Aufgabenstellung. Ob Adressenverwaltung oder Telefonverzeichnis, ob Bibliographie oder Personaldatei - immer sind bestimmte Funktionen auf einer Menge von Datenobjekten auszuführen:

- **Neuaufnahme**: Daten werden eingegeben und in die Datei aufgenommen.

- **Löschen**: bestimmte Datenobjekte werden gelöscht.

- **Ändern**: Daten werden abgeändert.

- **Drucken**: bestimmte Daten werden auf dem Drucker ausgegeben.

- **Auswahl**: die zu löschenden, zu ändernden oder auszugebenden Datenobjekte müssen erst ausgewählt (selektiert) werden.

Da mehrere Eigenschaften oder Merkmale vieler Datenobjekte verwaltet werden sollen, muß die Datei strukturiert sein. Die traditionelle Einteilung entspricht der einer Kartei:

EDV	Inhalt	manuelle DV
Datei	Gesamtheit aller Objekte	Kartei
Datensatz	Beschreibung eines Objekts	Karteikarte
Datensatzfeld	ein Merkmal des Objekts	Feld

1.2 Konkretisierung der Aufgabenstellung

1.2.1 Hardware

Da der HP-41CV bzw. das HP-82170A QUAD MEMORY stark verbreitet sind, kann von einem Rechner mit maximalem Arbeitsspeicher (320 Register einschließlich permanentem .END.) ausgegangen werden. Die zahlreichen Funktionen des HP-82180A X FUNCTIONS Modules bzw. die entsprechenden Anweisungen des HP-41CX ermöglichen erst eine effiziente Programmierung des Rechners. Das Dateisystem benötigt insbesondere die ALPHA-Funktionen sowie die ASCII-File-Datenstruktur. So bildet der HP-41CX die beste Grundlage für die hier vorgestellte Software, jedoch eignen sich auch der HP-41CV und - eventuell mit Port-Extender - der HP-41C. Je nach der Anzahl der Datensätze pro Cassette sind noch ein oder zwei HP-82181A X MEMORY Module notwendig.

Zur Speicherung der Datei dient das HP-82161A DIGITAL CASSETTE DRIVE. Das System soll neben IL-Druckern (HP-82162A) auch den "alten" Einsteckdrucker HP-82143A unterstützen.

1.2.2 Mengengerüst

Die Datei soll unbegrenzt viele Datensätze aufnehmen. Sie kann sich also über mehrere Cassetten erstrecken, die automatisch verwaltet werden. Das ADRESS-System speichert pro Cassette maximal 413 Datensätze mit je bis zu 206 Zeichen (vgl. 2.3). Ein Datensatz des ADRESS-Systems enthält 17 Felder:

WAHL	Zwischenspeicher für eine Auswahl
ANREDE1, TITEL1, VORNAME1, NACHNAME1	erste Person
ANREDE2, TITEL2, VORNAME2, NACHNAME2	zweite Person
ZUSATZ	zusätzliches Feld
STRASSE,_NR., PLZ, ORT	Adresse
FELD1, FELD2, FELD3, FELD4	zusätzliche Felder

Jedes Datensatzfeld faßt maximal 24 Zeichen. Dies erlaubt eine effiziente Verarbeitung direkt im ALPHA-Register, das auch bei der Eingabe verwendet wird.

1.2.3 Zeichendarstellung

Der Rechner kann im Display nur die Kleinbuchstaben a bis e, Großbuchstaben und Sonderzeichen anzeigen. Da Großbuchstaben relativ selten vorkommen, werden Kleinbuchstaben als Großbuchstaben dargestellt und eingegeben. Großbuchstaben werden durch ein vorangestelltes : gekennzeichnet (dieses Zeichen benötigt in der Anzeige keinen zusätzlichen Platz). Mit den Sonderzeichen für ß und die Umlaute gelten folgende Ersatzdarstellungen bei der Eingabe und der Anzeige:

Kleinbuchstaben	A bis Z	
Großbuchstaben	:A bis :Z	
ä, Ä	a	("SHIFT" "A"), :a
ö, Ö	↓	("SHIFT" "O"), :↓
ü, Ü	≠	("SHIFT" "H"), :≠
ß	$	("SHIFT" "P"),
	wird bei Großschrift in SS umgewandelt.	

1.2.4 Funktionen

- **Neuaufnahme** eines Datensatzes;

- **Auswahl** eines oder mehrerer Datensätze nach definierbaren Feldinhalten oder -teilinhalten;

- **Ändern** ausgewählter Datensätze: entweder manuell einzeln (ähnlich Neuaufnahme) oder automatisch einheitlich (bei Zahlen auch relativ);

- **Löschen** ausgewählter Datensätze: entweder im Dialog einzeln oder automatisch; versehentlich gelöschte Datensätze können unter Umständen "repariert" werden;

- **Drucken** ausgewählter Datensätze: das Format (Position der einzelnen Felder, Breit- oder Normalschrift) ist definierbar, die Reihenfolge der Felder jedoch fest; Groß- und Kleinschrift oder nur Großschrift; Umlaute und ß ; zusätzliche Schnelldruck-Routine, die die Feldinhalte ausdruckt, wie sie in der Anzeige dargestellt werden;

- **Zählen** der ausgewählten und aller Datensätze; Ausgabe über Anzeige oder Drucker;

- Anzeige oder Ausdruck eines Datei-**Inhaltsverzeichnisses** (je Cassette);

- **Cassettenwechsel** mit Benutzerführung;

- Zusätzliche vom Anwender **programmierbare Funktion**;

- Kontrolliertes **Ende** der Dateibearbeitung.

Alle Funktionen können beliebig kombiniert werden. Die selektierten Datensätze bleiben ausgewählt. So können Und- oder Oder-Verknüpfungen realisiert werden.

Bei der Neuaufnahme, der Auswahl und der manuellen Änderung können :HERRN durch H und :FRAU durch F abgekürzt werden (Felder ANREDE1 und ANREDE2); ein = im Feld NACHNAME2 kopiert den Inhalt des Feldes NACHNAME1 (**programmierte Abkürzungen** des ADRESS-Systems). Das System überprüft, ob die Felder NACHNAME1, ORT und FELD1 nicht leer sind (**Plausibilitätskontrolle**).

1.2.5 Bedienung

Auch Benutzer, die mit dem HP-41 nicht vertraut sind, sollen nach kurzer Zeit - ohne Nachschlagen in der Bedienungsanleitung - mit dem System umgehen können. Deshalb verfügen die Programme über nur zwei einheitliche Benutzerschnittstellen:

- **Fragen des Systems** werden mit JA oder NEIN beantwortet (NEIN kann entfallen oder durch beliebigen Text außer JA ersetzt werden). Das **Menu** dient zur Auswahl der einzelnen Funktionen. Es besteht aus einer Reihe von Fragen. Nach der Ausführung einer Funktion wird wieder in das Menu verzweigt.

- Die **Satzeingabe** dient zur Eingabe aller satz- oder feldbezogenen Daten. Neben Datensatzinhalten (Neuaufnahme, manuelles Ändern) sind dies die Suchbegriffe bei der Auswahl, die automatischen Änderungen und das Druckformat. Die Satzeingabe kann beliebig oft wiederholt werden.

Ein Datensatzfeld kann natürlich auch leer sein. Es enthält dann an erster Stelle entweder

- ein ↑ : das Feld ist zwar leer, der Inhalt wird aber noch ermittelt und später eingegeben. Durch die Auswahl nach ↑ erhält man alle unvollständigen Datensätze (Datenpflege);

- ein (Space, Blank): das Feld ist - und bleibt voraussichtlich - leer.

1.2.6 Modularisierung, residente und transiente Files

Die gesamte Software wird modular aufgebaut (vgl. 6.1). Hierzu wird die komplexe Aufgabe Dateibearbeitung hierarchisch in leichter überschaubare und konkretere Einzelprobleme zerlegt (**stepwise refinement**).

Als erste Stufe der Verfeinerung bietet sich eine Aufteilung nach den Funktionen an, die dem Benutzer direkt zugänglich sind (**Anwendermodule**). Dies sind die Hilfsprogramme (Module **BE**, **IN**, **UP**) sowie die Bearbeitungsfunktionen des Systems (mit Kleinbuchstaben als Modulnamen).

Die Anwendermodule rufen ihrerseits eine große Anzahl von internen Funktionen auf, die der Benutzer normalerweise nicht erreichen kann (**interne Module**, benannt mit Großbuchstaben). Die internen Module rufen sich auch gegenseitig auf.

Der große Umfang der Software erfordert eine Aufteilung auf mindestens drei Programmfiles, die auch auf den Datencassetten gespeichert werden (Modul **IN**). Dies ermöglicht das Laden von Programmteilen ohne Cassettenwechsel. Da die internen Module von mehreren Anwendermodulen und anderen internen Modulen aufgerufen werden, stehen sie - mit einer Ausnahme (Modul **A**) - nach dem Start der Dateibearbeitung dauernd im Programmspeicher des Rechners (in einem **residenten File**, ADRESS-System: **A**). Die Anwendermodule sind in **transienten Files** (**AA**, **AAA**) gespeichert. Wird ein solches Programmfile von der Cassette eingelesen, ersetzt es das zuvor geladene transiente File. Die Filenamen werden aus Zeichen mit ASCII-Codes größer als 127 (dezimal) gebildet, um Kollisionen mit den Namen der Datensätze zu vermeiden.

2 Datenstruktur

Bild 2-1 zeigt die Datenstruktur des Systems. Die Namen der Funktionen sind durch die entsprechenden Modulnamen abgekürzt.

2.1 Datei

Zum Speichern einzelner Datensätze im Rechner eignen sich sehr gut die ASCII-Files des Extended Memories. Sie verfügen über eine der Datensatzstruktur entsprechende Einteilung in Records. Diese ist zudem dynamisch: Speicherplatz, der für ein Feld nicht benötigt wird, kann für die anderen Felder verwendet werden. Ein weiterer Vorteil ist die einfache Editierbarkeit dieser Struktur auf Zeichen- und Feldebene. Die entsprechenden Funktionen sind allerdings wegen des hohen internen Verwaltungsaufwandes relativ langsam.

ASCII-Files können direkt mit IL-Massenspeichern ausgetauscht werden. Die AS-Files des Cassettenlaufwerks müssen jedoch einzeln formatiert (CREATE) werden und besitzen vor der ersten Speicherung eines ASCII-Files aus dem Extended Memory den Typ DA-File. Ein GETAS eines solchen leeren DA-Files führt zur Fehlermeldung FL_TYPE_ERROR.

Zur Bearbeitung im Rechner werden Datensätze in die ASCII-Files **A** oder **V** geladen. Felder oder Zeichen dieser Files (sie werden fast immer durch einen einzigen Zeiger adressiert) heißen im folgenden **AFELD**, **AZEICHEN**, **VFELD** bzw. **VZEICHEN**.

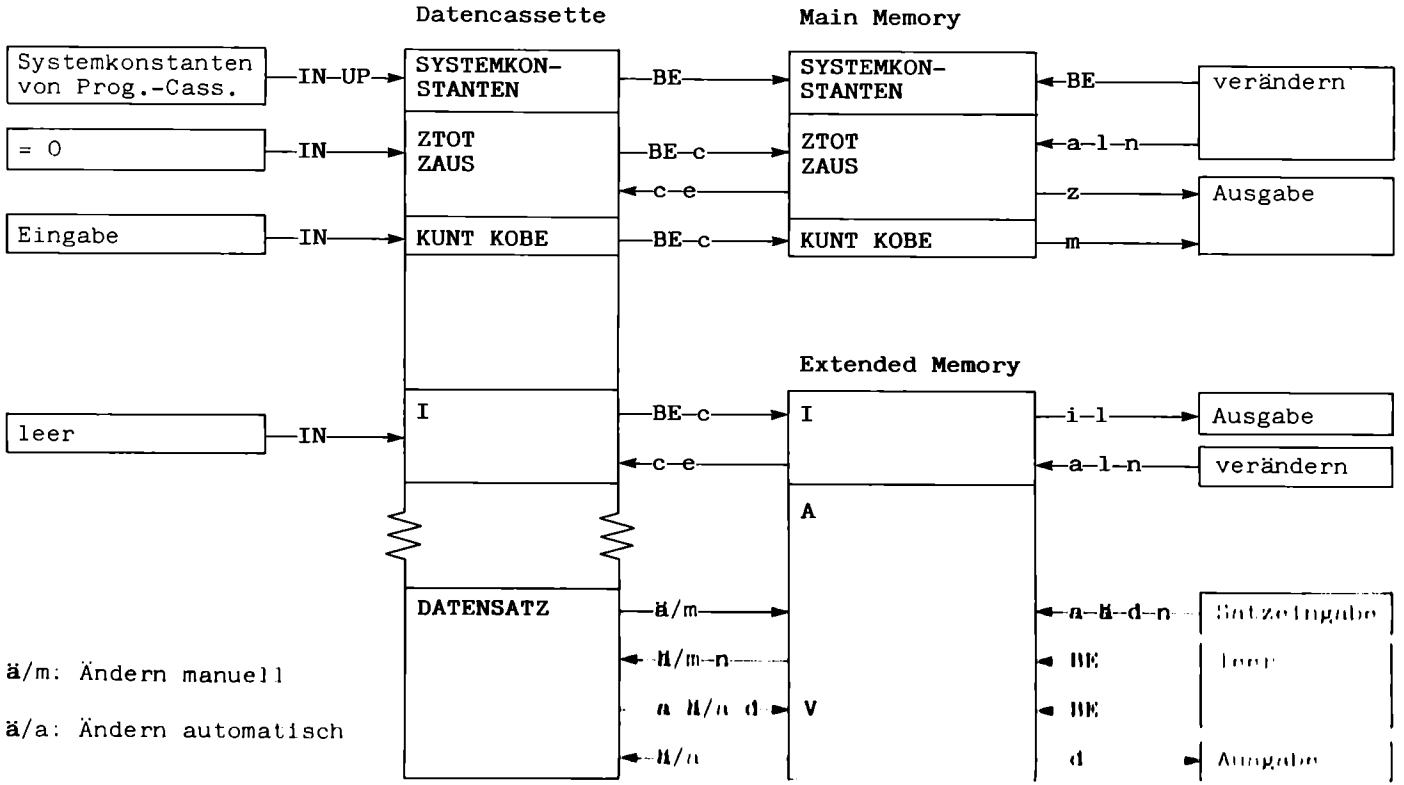


Bild 2-1 Datenstruktur des DATEI-Systems

2.2 Daten zur Verwaltung der Datei

2.2.1 Inhaltsverzeichnis

Die zentralen Mittel zur Verwaltung der Datei sind die Inhaltsverzeichnisse. Sie werden neben den Cassetten-Directories geführt und enthalten die Namen aller Datensätze einer Cassette. Diese Datensatznamen (**DSN**) werden jeweils aus maximal sechs Zeichen (der Rest wird mit Blanks aufgefüllt) eines Datensatzfeldes (beim ADRESS-System die ersten sechs Zeichen des Feldes NACHNAME1) und einer binär codierten Zahl zwischen 45 und 255 gebildet (**CODE**). 44 ist der ASCII-Code des Kommas, das bei File-Operationen (X FUNCTIONS oder IL) als Argumenttrenner verwendet wird und deshalb nicht in Filenamen auftauchen darf.

Datensätze können **markiert** werden. Dazu stehen an der achten Position eines Datensatznamens alternativ folgende Zeichen:

- **&** - **demarkiert:** keine Markierung;

- **#** - **ausgewählt:** der entsprechende Datensatz ist ausgewählt (Auswahl, Neuaufnahme oder Löschen);

- **!** - **vorausgewählt:** der Datensatzname wurde vorselektiert; diese Markierung wird noch innerhalb der Auswahl-Funktion durch & oder # ersetzt;

- **µ** - **gelöscht:** der entsprechende Datensatz ist - logisch - gelöscht und kann bei der Neuaufnahme überschrieben und damit auch physisch gelöscht werden.

Die Inhaltsverzeichnisse werden in den ASCII-Files **I** sowohl im Extended Memory als auch auf den Datencassetten gespeichert.

Die Datensätze sind auf den Cassetten unter ihren Datensatznamen gespeichert.

2.2.2 Cassettenkennung

Die Datei kann sich über mehrere Cassetten erstrecken. Da jeweils nur auf eine Cassette zugegriffen werden kann, muß es eine eindeutige Zuordnung zwischen den Datensatznamen und den Cassetten geben. Dies wird durch die Cassettenkennung erreicht.

Die Kennung besteht aus zwei Zeichen (beim ADRESS-System aus Buchstaben - Umlaute werden in die vokalen Äquivalente umgewandelt), die die lexikografischen Grenzen der Datensatznamen einer Cassette angeben. So können auf einer Cassette mit der Kennung G und K nur Datensätze gespeichert werden, deren Datensatznamen mit den Buchstaben G , H , I , J oder K beginnen. Diese Zuordnung wird vom System laufend überprüft. Paßt die Kennung nicht, wird der Benutzer zum Wechsel der Cassette aufgefordert, wobei das erste Zeichen des entsprechenden Datensatznamens angezeigt wird.

Diese Sicherung funktioniert nur korrekt, wenn die Menge der Datensatznamen lückenlos in sich nicht überschneidende Teilbereiche eingeteilt worden ist. Der Benutzer nimmt diese Einteilung bei der Initialisierung der Datencassetten (Modul **IN** bzw. **ADRIN**-Programm des ADRESS-Systems) vor.

Die ASCII-Werte der Kennung (**KUNT** und **KOBE**, untere und obere Kennung) werden auf der Cassette im Datenregisterfile (beim ADRESS-System: a) gespeichert. Die Kennung der eingelegten Cassette wird im Menu angezeigt.

2.2.3 Datensatzanzahlen

Damit nicht bei jedem Aufruf der Zählen-Funktion die Datensatznamen im Inhaltsverzeichnis gezählt werden müssen - eine recht langwierige Operation -, werden vom System laufend zwei weitere Variablen unterhalten:

- **ZTOT**: die Anzahl aller Datensätze der eingelegten Cassette

- **ZAUS**: die Anzahl der ausgewählten Datensätze der Cassette

Diese Werte werden im Datenregisterfile auf der Cassette gespeichert.

2.2.4 Systemkonstanten

Die Konstanten werden nur einmal zu Beginn der Dateibearbeitung (Modul **BE** bzw. **ADRBE**-Programm) aus dem Datenregisterfile in die Register des Rechners geladen.

2.3 Mengengerüst

2.3.1 Speicherung im Rechner

Daten und Programme im Main Memory

Der Rechner zeigt bei leerem Programmspeicher und SIZE_000 319 freie Register an. Drei Bytes eines Registers werden jedoch vom permanenten .END. belegt, das den Programmspeicherbereich - auch den leeren - abschließt. Da dieses .END. die END-Anweisung des jeweils letzten Programmes im Speicher ersetzen kann, stehen für Daten und Programme sowie für Tastenzuordnungen, Alarme etc. insgesamt 320 Register zur Verfügung.

ASCII-Files im Extended Memory

Je nach der Zahl der eingesetzten X MEMORY Module beträgt die Kapazität des Extended Memories 126, 364 oder 602 Register mit je 7 Bytes. Jedes File des Extended Memories benötigt zwei zu-

sätzliche Register für interne Verwaltungsinformationen. EMDIR wie FLSIZE zeigen die Filelängen ohne diese Header-Register an. Die von EMDIR gelieferte Zahl berücksichtigt jedoch die beiden Register für das "nächste" File, sodaß die Anweisungsfolge EMDIR CRFLAS das Extended Memory ganz belegt.

Den Records eines ASCII-Files im Extended Memory - bei IL-Massenspeichern verhält es sich anders - ist jeweils ein Byte vorangestellt, das die Recordlänge in Bytes (Zeichen) angibt. Ein Byte nach dem letzten Record kennzeichnet mit dem Wert 255 (hexadezimal FF) das Ende des Files. Deshalb ist die Recordlänge bei ASCII-Files auf 254 Zeichen beschränkt.

Somit benötigt ein ASCII-File des Extended Memories mit einer Kapazität von z Zeichen in f Feldern (Records)

2 + c = 2 + aufgerundet((z + f + 1) / 7) Register.

c ist die Registeranzahl, die bei CRFLAS anzugeben ist.

Programme im Main Memory

Beim Laden von Programmen von IL-Massenspeichern ist die Programmlänge in Registern (nicht in Bytes) maßgebend. Für Programme der Länge b_n gilt also bei d Datenregistern:

aufgerundet(b_1 / 7) + aufgerundet(b_2 / 7) + ... ≤ 320 - d

Programme im Extended Memory

Auch ein PR-File des Extended Memories benötigt zwei zusätzliche Register zur Verwaltung. Die Programmbytes werden um ein Byte Prüfsumme ergänzt. Ein Programm mit b Bytes benötigt also

2 + c = 2 + aufgerundet((b + 1) / 7) Register.

2.3.2 Speicherung auf Cassette

IL-Massenspeichermedien sind in Records - nicht zu verwechseln mit den Records der ASCII-Files - mit je 256 Bytes eingeteilt. Die Digitalcassette hat eine Kapazität von 512 Records, die wie folgt belegt werden:

2 Records	interne Verwaltungsdaten (Header)
n Records	(n = 1 bis 56) Directory
512 - n - 2 Records	Daten und Programme

Ein Record des Directories nimmt acht Einträge auf, der letzte nur sieben. Der Rest eines "angebrochenen" Records kann nicht für ein anderes File verwendet werden. Daten und Programme werden auf der Cassette nicht in dem Format des Main oder Extended Memories des Rechners gespeichert.

DA- und AS-Files sind in Registern zu je 8 Bytes organisiert. Ein mit 32 CREATE erzeugtes DA-File ist also nicht 224 sondern 256 Bytes lang und belegt exakt einen Record.

Datenregister auf Cassette

Ein Record eines DA-Files faßt - wie eine Magnetkarte - 32 Datenregister des Rechners. Die 7-Byte-Register-Inhalte werden bei der Speicherung in 8-Byte-Werte umgewandelt.

ASCII-Files auf Cassette

Um Datenkompatibilität mit anderen HP-IL-Computern zu erreichen, werden bei den AS-Files der IL-Massenspeicher jeweils zwei Bytes für die Speicherung der Recordlänge verwandt. Das Endezeichen hat den Wert 65535 (hexadezimal FFFF).

Den 7-Byte-Registern des Extended Memories entsprechen jedoch Register mit 8 Bytes beim IL-Massenspeicher. So läßt sich in

ASCII-Files einheitlicher Länge (in Registern gemessen: CRFLAS bzw. CREATE) im Extended Memory und auf Cassette dieselbe Informationsmenge speichern, sofern die Feldanzahl kleiner als die Registerzahl ist. Die folgenden Beispiele verdeutlichen diesen Zusammenhang:

30 Felder mit je 6 Bytes und ein Feld mit 12 Bytes

Extended Memory:	30 * 6 + 12	= 192 Datenbytes
	31 + 1	= 32 Formatbytes
32 Register	32 * 7	= 224 Bytes
Cassette:	30 * 6 + 12	= 192 Datenbytes
	31 * 2 + 2	= 64 Formatbytes
32 Register	32 * 8	= 256 Bytes

31 Felder mit je 6 Bytes und ein Feld mit 5 Bytes

Extended Memory:	31 * 6 + 5	= 191 Datenbytes
	32 + 1	= 33 Formatbytes
32 Register	32 * 7	= 224 Bytes
Cassette:	31 * 6 + 5	= 191 Datenbytes
	32 * 2 + 2	= 68 Formatbytes
33 Register	33 * 8	> 259 Bytes

Programme auf Cassette

Programme werden direkt transferiert, jedoch im IL-Massenspeicher um ein Byte Prüfsumme ergänzt. So faßt der letzte Record eines PR-Files auf der Cassette nur 255, alle anderen 256 Programmbytes.

2.3.3 Folgerungen für das Mengengerüst - FORMAT

Aus den dargestellten Zusammenhängen lassen sich zwei Restriktionen für die Kapazität eines Datensatzes und für die Anzahl der Datensätze pro Cassette ableiten.

Vorausgesetzt wird, daß die Anzahl der Felder eines Datensatzes kleiner ist als seine Länge in Registern. Dies ist ab einer durchschnittlichen Feldlänge von 7 Zeichen sicher der Fall. Da Datensatznamen mit Markierung acht Bytes belegen, ist die entsprechende Voraussetzung für das Inhaltsverzeichnis I erfüllt.

Die gegebenen Ausgangsgrößen sind:

ZA Anzahl der zusätzlichen Files auf der Datencassette (Programme, Datenregisterfile etc., ohne I)
ZL Länge aller dieser Files in Records
FA Anzahl der Felder pro Datensatz
XA Anzahl der eingesetzten X MEMORY Module

Die Größen

SA Anzahl der Datensätze pro Cassette
SK Kapazität eines Datensatzes in Zeichen

sind zu maximieren. Es treten folgende Zwischengrößen auf:

XK Nettokapazität des Extended Memories (ohne Headers) in Registern
CK Nettokapazität einer Cassette (ohne Directory und Header) in Records
SX Länge eines Datensatzes im Extended Memory in Registern
SC Länge eines Datensatzes auf der Cassette in Records
IX Länge des Inhaltsverzeichnisses I im Extended Memory in Registern
IC Länge des Inhaltsverzeichnisses I auf der Cassette in Records

Der Rechengang für das Extended Memory:

```
XK =      122     +     240 * XA     -     3 * 2
     (X FUNCTIONS)    (X MEMORY)    (Header A, V, I)
SX = aufgerundet((SK + FA + 1) / 7)
```

```
IX = aufgerundet((8 * SA + SA + 1) / 7)
2 * SX + IX ≤ XK (in Registern)
(A, V)  (I)

... und für die Cassette:

CK = 512 - 2 - aufgerundet((ZA + SA + 1 + 1) / 8)
       (Header)                      (I)
SC = aufgerundet(SX / 32)
IC = aufgerundet(IX / 32)
SA * SC + IC ≤ CK (in Records)
```

Bild 2-2 zeigt das Programm **FORMAT** (Barcode s. 6.2), das beliebige Kombinationen der Ausgangsgrößen und der zu optimierenden Werte testet und die Güte der Optimierung anzeigt. Die Bedienung des Programmes (XXXX bezeichnet eine Ausgabe des Rechners, £ steht für vom Programmlauf abhängige Zeichen):

1. CF_21 oder SF_21 und Printer auf NORM stellen (zur Protokollierung)
2. XEQ "FORMAT"
3. ZUS.FILES ANZAHL? £
4. ZA eingeben oder ändern (ohne R/S)
5. ZUS.FILES LAENGE REC.S? £
6. ZL eingeben oder ändern
7. SATZLAENGE ZEICH.? £
8. SK eingeben oder ändern
9. FELDANZAHL? £
10. FA eingeben oder ändern
11. SATZANZAHL? £
12. SA eingeben oder ändern
13. XMEM. ANZAHL? £
14. XA eingeben oder ändern
15. XMEM. £ REG.S
16. falls das Extended Memory nicht ausreicht (£ REG.S < 0), weiter bei 7.
17. CASS. £ REC.S
18. weiter bei 7.

An den Zahlen der nicht belegten (£ positiv) oder der überzähligen (£ negativ) Register bzw. Records erkennt man die Güte der Annäherung an das Optimum. Mit einem Druck auf die USER-Taste kann die Eingabe vorzeitig verlassen werden (weiter bei 15.).

```
01♦LBL "FORMAT"
   02 8
   03 PSIZE
   04 CLRG
   05 CF 27
   06 CF 29
   07 FIX 0
08 "ZUS.FILES "
   09 XEQ 03
10 "ZUS.FILES LAENG"
11 "ÞE REC.S"
   12 XEQ 04

   13♦LBL 15
   14 CF 22
   15 2
   16 STO 07
17 "SATZLAENGE ZEIC"
   18 "ÞH."
   19 XEQ 04
   20 "FELD"
   21 XEQ 03
   22 "SATZ"
   23 XEQ 03
   24 "XMEM. "
   25 XEQ 03

   26♦LBL 16
   27 RCL 02
   28 RCL 03
   29 +
   30 XEQ 01
   31 STO 06
   32 ST+ X
   33 9
   34 RCL 04
   35 *
   36 XEQ 01
   37 STO 07
   38 +
   39 RCL 05
   40 240
   41 *
   42 -
   43 116
44 " REG.SXMEM. "
   45 XEQ 00
   46 RCL 06
   47 32
   48 XEQ 02
   49 RCL 04
   50 *
   51 RCL 01
   52 +
   53 RCL 07
   54 32
   55 XEQ 02
   56 +
   57 RCL 00
   58 RCL 04
   59 +
   60 2
   61 +
   62 8
   63 XEQ 02
   64 +
   65 510
66 " REC.SCASS. "
   67 XEQ 00
   68 BEEP
   69 GTO 15

   70♦LBL 00
   71 -
   72 CHS
   73 X>0?
   74 "Þ+"
   75 ARCL X
   76 6
   77 AROT
   78 X<>Y
   79 X<0?
   80 TONE 0
   81 AVIEW
   82 X<0?
   83 GTO 15
   84 RTN

   85♦LBL 01
   86 1
   87 +
   88 7

   89♦LBL 02
   90 /
   91 ENTER↑
   92 FRC
   93 X≠0?
   94 SIGN
   95 +
   96 INT
   97 RTN

   98♦LBL 03
   99 "ÞANZAHL"

  100♦LBL 04
  101 "Þ? "
102 ARCL IND 07
  103 TONE 9
  104 AVIEW
  105 PSE
  106 FC? 22
  107 PSE
  108 CLD
  109 FS?C 22
110 STO IND 07
  111 FS?C 27
  112 GTO 16
  113 ISG 07
  114 .END.
```

Bild 2-2 Listing **FORMAT**

2.3.4 Folgerungen für die Speicherung der Software

Für die Aufteilung der Module auf das residente und die transienten Programmfiles sind vor allem folgende Aspekte relevant:

- Programmfilelängen von knapp unter n * 256 Bytes (n natürliche Zahl) sichern eine gute Ausnutzung des Speichermediums.

- Für das residente Programmfile, das längste transiente File und die Datenregister stehen insgesamt 320 Register des Rechners zur Verfügung.

- Der Programmspeicher des Rechners wird effizient genutzt, wenn die transienten Files ungefähr gleich lang sind.

3 Software

Hier sind alle Module des ADRESS-Systems in Katalogform aufgeführt (zur Modularisierung vgl. 1.2.6 und 6.1). Die einheitliche Struktur der Modulbeschreibungen erleichtert das Auffinden bestimmter Einzelheiten:

- Kurze Beschreibung der **Aufgabe** des Moduls innerhalb des gesamten Systems; die Funktionsbeschreibungen der Anwendermodule ergeben zusammengenommen die Bedienungsanleitung der Adressenverwaltung.

- Übersichtliche Darstellung wichtiger **Eigenschaften** des Moduls: Namen der aufrufenden und aufgerufenen Module, genaue Definition der Schnittstelle (Register und Flags als Eingangs- oder Ausgangsparameter), Name des nach Verlassen des Moduls definierten working files, Inhalte der verwendeten Datenregister, Bedeutung der benutzten Flags (im gesetzten Zustand). Die wichtigsten dieser Daten finden sich in zusammengefaßter Form in 6.1 (Cross Reference).

- Eingehende Beschreibung algorithmischer oder programmiertechnischer **Besonderheiten**, soweit sie nicht als Maßnahmen der Programmoptimierung in Teil 5 aufgeführt sind.

- Hinweise auf die bei der Programmierung einer eigenen Dateiverwaltung möglichen oder notwendigen **Modifikationen**. In einigen Fällen werden auch Module anderer Dateiverwaltungen gezeigt.

- Der Text wird jeweils durch einen **Programmablaufplan** sowie ein **Programmlisting** ergänzt.

Falls nicht anders vermerkt, beziehen sich alle Daten (Zeilennummern, Labelargumente etc.) auf die Module des ADRESS-Systems.

3.1 Anwendermodule

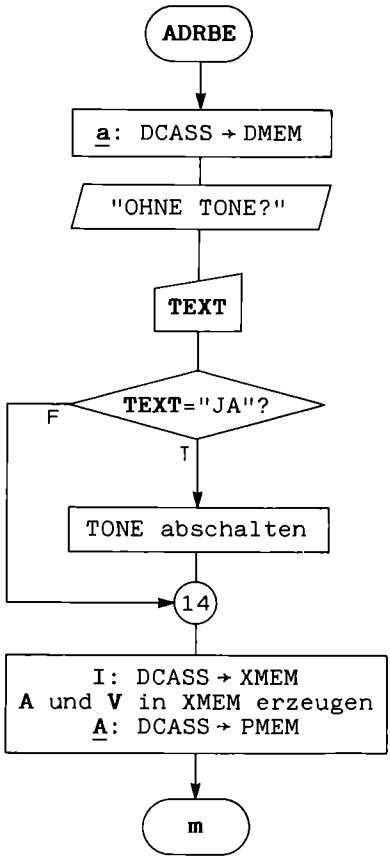
3.1.1 BE - Bootstrap der Dateibearbeitung

Aufgabe

Das kurze Programm dient als Vorlauf für das eigentliche Dateiverwaltungsprogramm, das mit dem Menu beginnt.

Alle Programme des Dateisystems setzen ein MEMORY_LOST voraus. So kann der Benutzer mit der einfachen master clear-Prozedur einheitliche Startbedingungen schaffen und mit einer über die Tastatur ausgeführten READP-Anweisung das Programm laden und starten. Wegen der durch master clear bewirkten Speicherverteilung von minimal (HP-41C/CV) nur 47 Registern Programmspeicher darf dieses erste Programm nur 329 Bytes lang sein. Es kann aber seinerseits den Speicher mit PSIZE neu einteilen und ein längeres Programm laden. **BE** führt noch Aktionen aus, die nur einmal pro Lauf der Dateiverwaltung notwendig sind:

- Einlesen des Datenregisterfiles **a**
- Frage, ob TONE und BEEP abgestellt werden sollen
- Initialisierung der Flags
- Erzeugung der ASCII-Files **A**, **V** und **I** im Extended Memory
- Neuverteilung des Speicherbereiches
- Laden des residenten Programmfiles **A**; das **ADRBE**-Programm wird überschrieben und der Programmlauf mit der ersten Anweisung von **A** fortgesetzt.



DCASS: Datencassette
DMEM : Datenspeicher
PMEM : Programmspeicher
XMEM : Extended Memory

Bild 3-1 Ablaufplan ADRBE

```
01♦LBL "ADRBE"
   02 ""
   03 READR
   04 RCL 31
   05 STOFLAG
   06 AON
07 "OHNE TONE?"
   08 TONE 7
   09 AVIEW
   10 PSE
   11 FC? 23
   12 PSE
   13 FC?C 23
   14 PSE
   15 ASTO X
   16 "JA"
   17 ASTO Y
   18 X≠Y?
   19 GTO 14
   20 CF 26
   21 RCLFLAG
   22 STO 31

   23♦LBL 14
   24 EMDIR
   25 CLD
   26 68
   27 -
   28 "I"
   29 CRFLAS
   30 SF 25
   31 GETAS
   32 CF 25
   33 RCL 11
   34 PSIZE
   35 "A"
   36 CRFLAS
   37 "V"
   38 CRFLAS
   39 ""
   40 READP
   41 .END.
```

Bild 3-2 Listing ADRBE

Moduleigenschaften

aufgerufenes Modul: **m**
working file: **V**

Besonderheiten

Die Zeilen 008 bis 017 entsprechen fast genau einem Aufruf des Modules **J**, welches jedoch zum Zeitpunkt des **BE**-Laufes nicht geladen ist.

Die Größe des ASCII-Files **I** wird nach dem zur Verfügung stehenden Extended Memory bestimmt (vgl. 2.3).

Die Anweisungen 030 SF_25 und 032 CF_25 vermeiden die Fehlermeldung FL_TYPE_ERR beim GETAS eines leeren Inhaltsverzeichnisses **I**, das ja in diesem Fall noch den Typ DA-File hat.

Modifikation

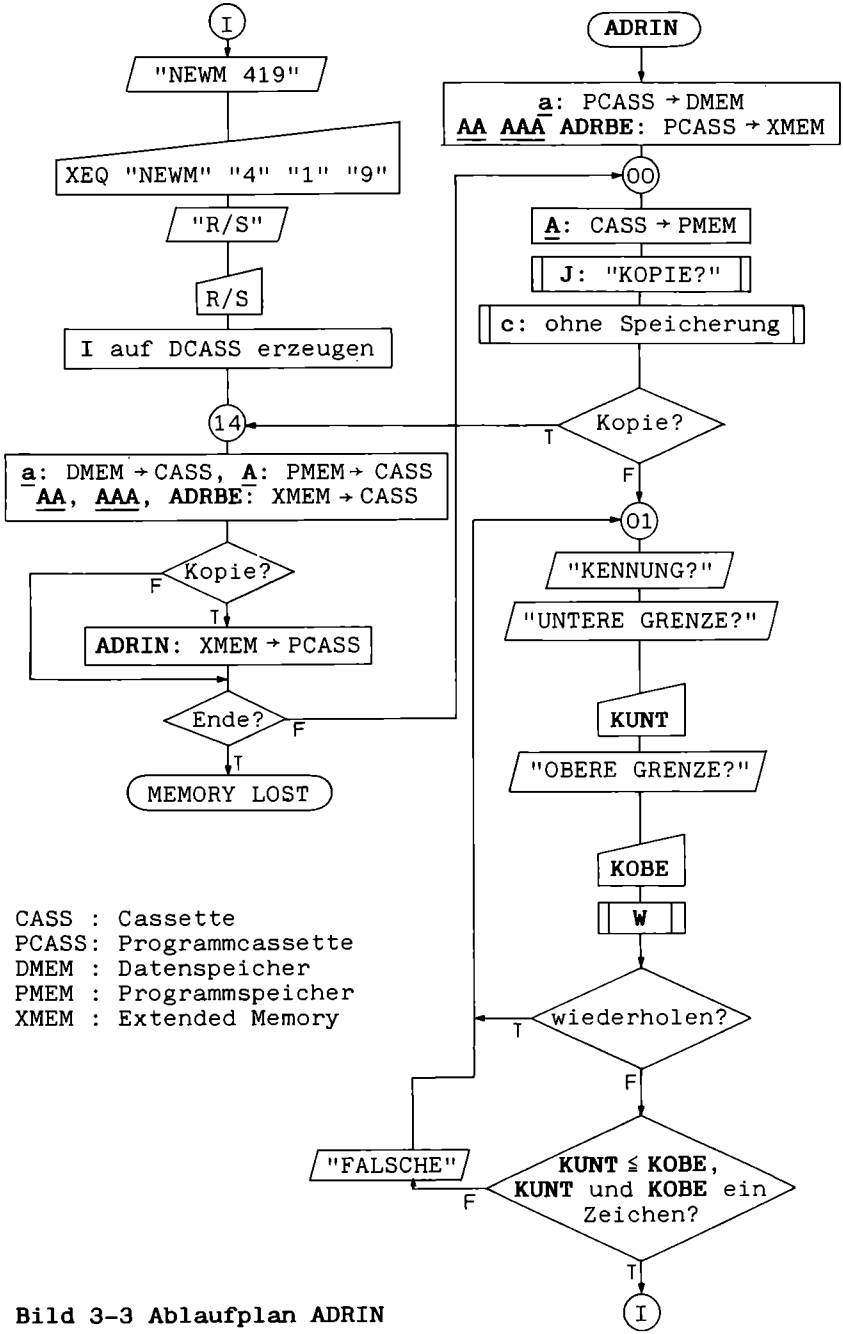
001 LBL_"ADRBE" - Name des Programmes mit dem Modul **BE**
002 "a"- Name des Datenregisterfiles
026 68 - Größe 2 * SX + 4 (vgl. 2.3.3)
039 "A" - Name des residenten Programmfiles

3.1.2 IN - Initialisierung der Datencassetten oder Kopie

Aufgabe

Das Modul **IN** vereinigt zwei Funktionen:

- Vorbereitung leerer oder zu löschender Cassetten zur Speicherung der Datensätze



CASS : Cassette
PCASS: Programmcassette
DMEM : Datenspeicher
PMEM : Programmspeicher
XMEM : Extended Memory

Bild 3-3 Ablaufplan ADRIN

```
01♦LBL "ADRIN"
  02 ""
  03 READR
  04 RCL 11
  05 PSIZE
  06 RCL 31
  07 STOFLAG
  08 AON
  09 ""
  10 2
  11 RCL b
  12 "├"
  13 READSUB
  14 SAVEP
  15 PCLPS
  16 DSE Y
  17 STO b
  18 "ADRBE"
  19 READP
  20 SAVEP
  21 PCLPS

  22♦LBL 00
  23 ""
  24 READP
  25 "KOPIE"
  26 XEQ "J"
  27 X=Y?
  28 SF 06
  29 SF 07
  30 XEQ "c"
  31 "ENDE"
  32 XEQ "J"
  33 X=Y?
  34 SF 05
  35 FS? 06
  36 GTO 14
  37 CLA
  38 ASTO 04
  39 ASTO 05

  40♦LBL 01
41 "├KENNUNG"
  42 TONE 0
  43 PSE
  44 "UNT"
  45 XEQ 13
  46 "OB"
  47 XEQ 13
  48 XEQ "W"
  49 CLA
  50 X=Y?
  51 GTO 01
  52 ARCL 04
  53 ARCL 05
  54 ATOX
  55 STO 02
  56 ATOX
  57 STO 03
58 "FALSCHE "
  59 X>Y?
  60 GTO 01
  61 *
  62 X=0?
  63 GTO 01
  64 "R/S"
  65 ASTO X
  66 AOFF
67 "NEWM 419"
  68 BEEP
  69 PROMPT
  70 AON
  71 532
  72 "I"
  73 CREATE

  74♦LBL 14
  75 RCL 11
  76 ""
  77 CREATE
  78 WRTR
  79 FS? 06
  80 SEC
  81 ""
  82 WRTP
  83 SEC
  84 2
  85 RCL b
  86 "├"
  87 GETP
  88 WRTP
  89 SEC
  90 DSE Y
  91 STO b
  92 "ADRBE"
  93 GETP
  94 FC? 06
  95 SF 11
  96 WRTP
  97 SEC
  98 PCLPS
  99 "ADRIN"
 100 SF 11
 101 FS? 06
 102 WRTP
 103 FS?C 06
 104 SEC
 105 CF 11
 106 FC?C 05
 107 GTO 00
 108 STO c
 109 STOP

 110♦LBL 13
 111 ALENG
 112 2
 113 +
114 "├ERE GRENZE? "
 115 ARCL IND X
 116 TONE 9
 117 AVIEW
 118 PSE
 119 FC? 23
 120 PSE
 121 FC? 23
 122 PSE
 123 CLD
 124 FC?C 23
 125 RTN
 126 SIGN
 127 ALENG
 128 X>Y?
 129 CLA
130 ASTO IND L
 131 .END.
```

Bild 3-4 Listing ADRIN

- Anfertigen einer Sicherungskopie des gesamten Systems (KOPIE).

IN lädt und schreibt die Systemteile **ADRBE**, **ADRIN** (nur bei KOPIE), **A**, **AA** und **AAA** sowie das Datenregisterfile **a**. Auf Datencassetten (nicht KOPIE) wird außerdem das ASCII-File I erzeugt und die Cassettenkennung (**KUNT**, **KOBE**) gespeichert.

Moduleigenschaften

aufgerufene Module: **c J W**

Datenregister:	04	**KOBE** als Zeichen
	05	**KUNT** als Zeichen
Flags:	05	ENDE gewählt
	06	KOPIE gewählt

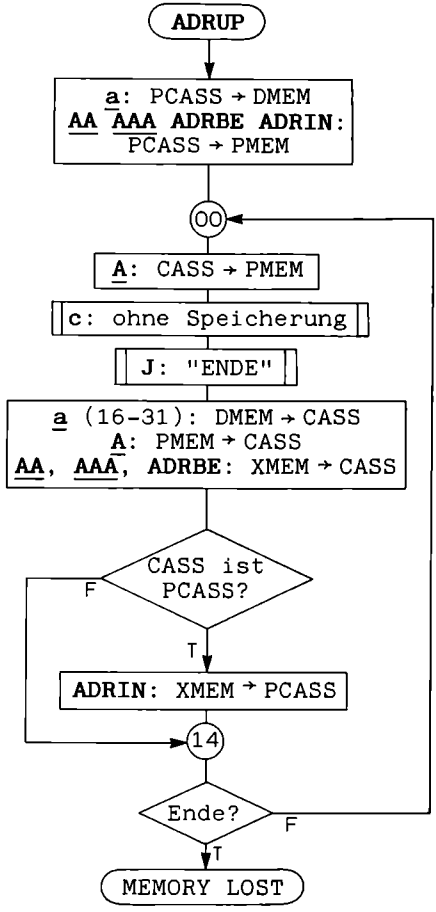
Besonderheiten

Die Programme und das Datenregisterfile - letzteres nur auf der Sicherungscassette - werden mit SEC vor versehentlichem Überschreiben geschützt. **ADRBE** wird auf Datencassetten wie **ADRIN** auf Sicherungscassetten mit SF_11 als automatisch startendes File gespeichert.

Modifikation

001	LBL_"ADRIN" - Name des Programmes mit dem Modul **IN**
099	"ADRIN"
018, 092	"ADRBE" - Name des Programmes mit dem Modul **BE**
009, 023, 081	"A" - Name des residenten Programmfiles
012, 086	"-A"
002, 076	"a" - Name des Datenregisterfiles
067	"NEWM_419" - der Wert SA + ZA + 1 (vgl. 2.3.3)
071	532 - der Wert IX für zwei X MEMORY Module (vgl. 2.3.3)

3.1.3 UP - Update des Systems



CASS : Cassette
PCASS: Programmcassette
DMEM : Datenspeicher
PMEM : Programmspeicher
XMEM : Extended Memory

Bild 3-5 Ablaufplan ADRUP

Das Modul **UP** dient zur Aktualisierung der Programme und Konstanten auf Programm- und Datencassetten. Dies ist nach Programmänderungen notwendig. Das Modul funktioniert ähnlich wie der KOPIE-Teil des Moduls **IN**.

Besonderheiten

Die Programme und das Datenregisterfile werden - wenn nötig - mit UNSEC entsichert und mit SEC vor versehentlichem Überschreiben geschützt. **ADRBE** (nur auf Datencassetten) und **ADRIN** (nur auf Programmcassetten) werden mit SF_11 als automatisch startende Programmfiles gespeichert.

```
01♦LBL "ADRUP"
  02 ""
  03 READR
  04 RCL 11
  05 PSIZE
  06 RCL 31
  07 STOFLAG
  08 AON
  09 ""
  10 2
  11 RCL b
  12 "├"
  13 READSUB
  14 SAVEP
  15 PCLPS
  16 DSE Y
  17 STO b
  18 "ADRBE"
  19 READP
  20 SAVEP
  21 PCLPS
  22 "ADRIN"
  23 READP
  24 SAVEP
  25 PCLPS

26♦LBL 00
  27 ""
  28 READP
  29 SF 07
  30 XEQ "c"
  31 "ENDE"
  32 XEQ "J"
  33 X=Y?
  34 SF 05
  35 "ADRIN"
  36 SF 25
  37 UNSEC
  38 FS?C 25
  39 SF 06
  40 ""
  41 FS? 06
  42 UNSEC
  43 6.031
  44 SEEKR
  45 WRTRX
  46 FS? 06
  47 SEC
  48 ""
  49 UNSEC
  50 WRTP
  51 SEC
  52 2
  53 RCL b
  54 "├"
  55 GETP
  56 UNSEC
  57 WRTP
  58 SEC
  59 DSE Y
  60 STO b
  61 "ADRBE"
  62 GETP
  63 UNSEC
  64 FC? 06
  65 SF 11
  66 WRTP
  67 SEC
  68 PCLPS
  69 FC?C 06
  70 GTO 14
  71 "ADRIN"
  72 GETP
  73 SF 11
  74 WRTP
  75 SEC

76♦LBL 14
  77 CF 11
  78 FC?C 05
  79 GTO 00
  80 STO c
  81 .END.
```

Bild 3-6 Listing ADRUP

Moduleigenschaften

aufgerufene Module: c J

Flags:

05	ENDE gewählt
06	eingelegte Cassette ist Programmcassette

Modifikation

001	LBL "ADRUP" - Name des Programmes mit dem Modul **UP**
018, 061	"ADRBE" - Name des Programmes mit dem Modul **BE**
022, 071	"ADRIN" - Name des Programmes mit dem Modul **IN**
009, 027, 048	"A" - Name des residenten Programmfiles
012, 054	"├A"
002, 040	"a" - Name des Datenregisterfiles

3.1.4 a - Auswahl

Aufgabe

Mit dem Modul **a** werden Datensätze nach den in der Satzeingabe (Modul **E**) definierten Suchkriterien ausgewählt. Dabei können ein ganzes Datensatzfeld oder nur die für die Auswahl nicht relevanten Zeichenpositionen mit ? "ausgeblendet" werden (wild card). Ein ? am Ende eines Suchkriteriums steht für beliebige folgende Zeichen. Vor der Eingabe der Suchbegriffe ist der gesamte Datensatz mit ? initialisiert. Die Plausibilität wird nicht überprüft, die programmierten Abkürzungen können nicht verwendet werden.

Es ist auch möglich, alle Sätze zu selektieren (ALLE SAETZE) oder die Rollen der ausgewählten und der noch nicht selektierten Datensätze nach der Auswahl zu vertauschen (AUSTAUSCH).

Die Datensatznamen der selektierten Sätze werden im Inhaltsverzeichnis **I** ausgewählt markiert. Diese Markierung kann nur durch INITIALISIEREN bei der Neuaufnahme (vgl. 3.1.12), bei der Auswahl oder beim Löschen (vgl. 3.1.10, REPARATUR) sowie durch AUSTAUSCH (s. o.) entfernt werden.

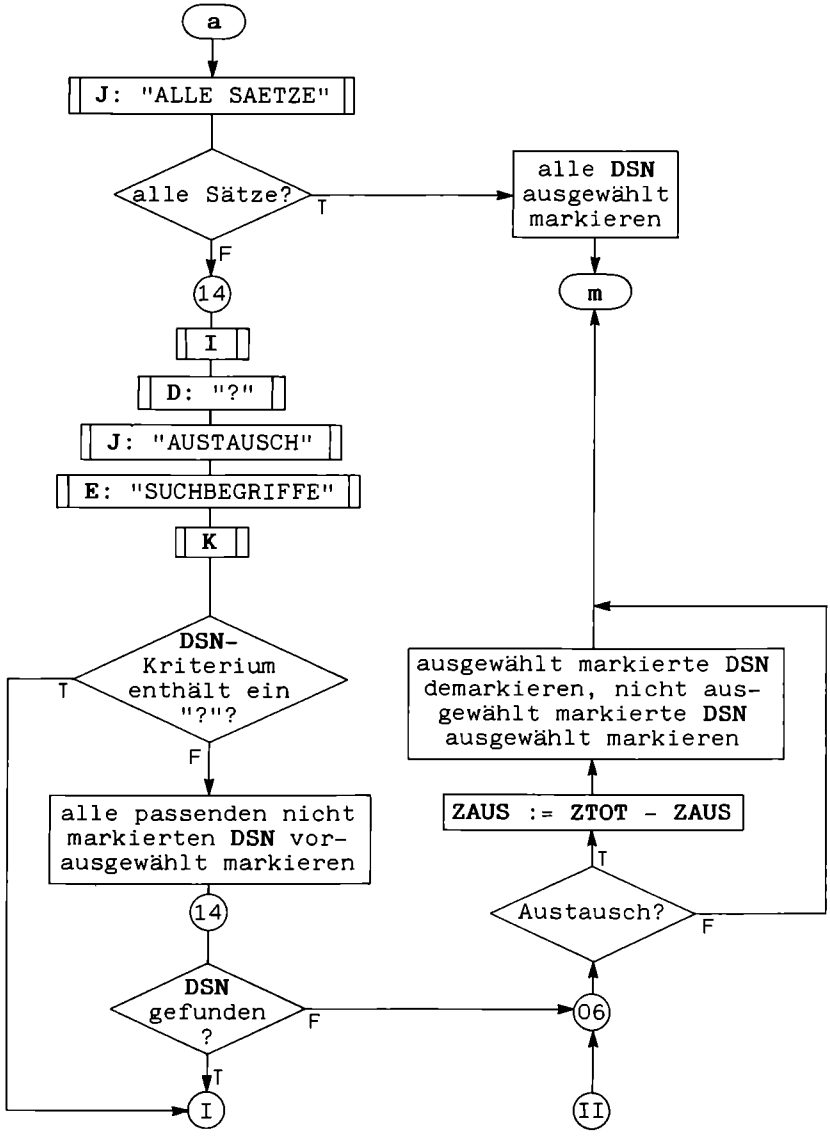


Bild 3-7/1 Ablaufplan a

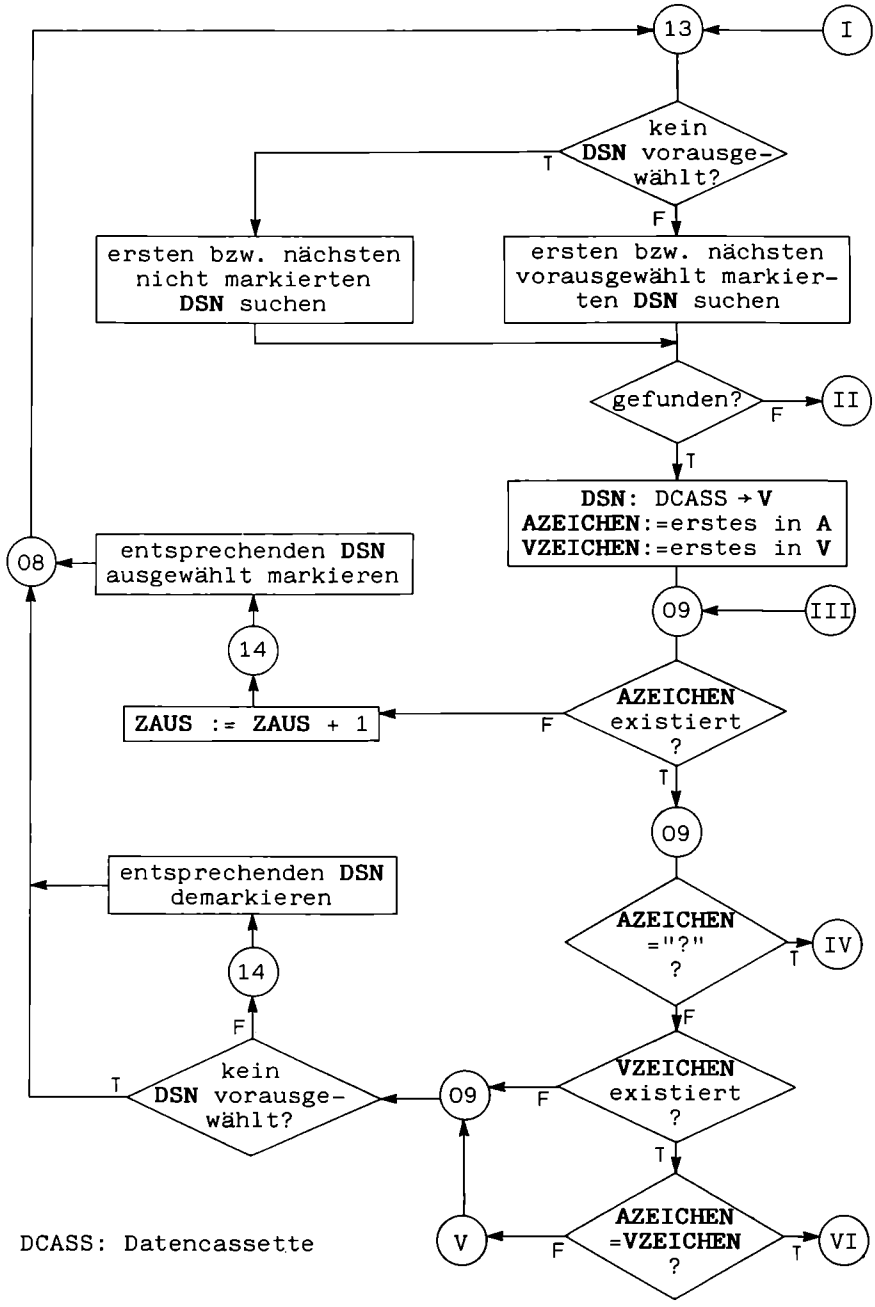


Bild 3-7/2 Ablaufplan a

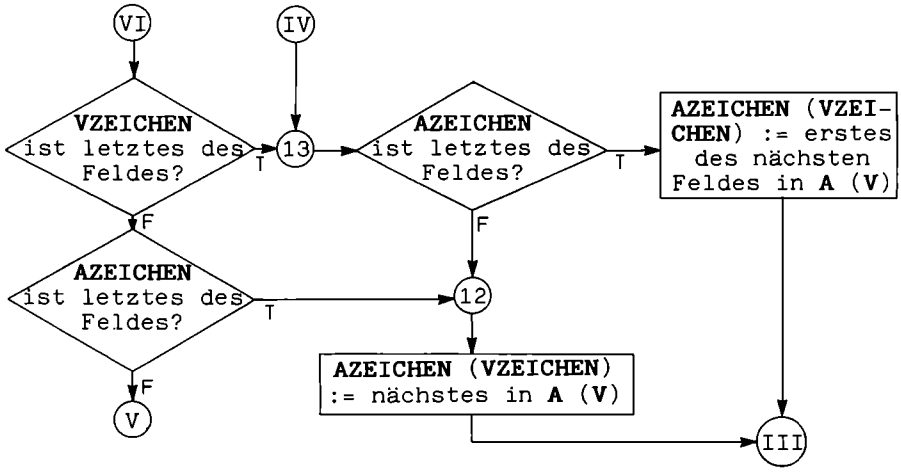


Bild 3-7/3 Ablaufplan a

Moduleigenschaften

aufrufendes Modul:	**m**	
aufgerufene Module:	**m D E I J K**	
working file:	I	
Datenregister:	04	Zeiger auf **AZEICHEN** und **VZEICHEN**
Flags:	07	AUSTAUSCH gewählt
	09	**AZEICHEN** ist letztes des Feldes
	10	mindestens ein Datensatz ist vorausgewählt

```
    236♦LBL "a"
237 "ALLE SAETZE"
    238 XEQ "J"
    239 X≠Y?
    240 GTO 14
    241 "I"
    242 CLX
    243 SEEKPTA
    244 RCL 00
    245 STO 01
    246 RCL b
    247 "&"
    248 POSFL
    249 X<0?
     250 GTO ""
    251 SIGN
    252 DELCHR
    253 "▮"
    254 INSCHR
    255 X<>Y
    256 STO b

    257♦LBL 14
    258 XEQ "I"
    259 RCL 14
    260 XEQ "D"
 261 "AUSTAUSCH"
    262 XEQ "J"
    263 X=Y?
    264 SF 07
265 "SUCHBEGRIFFE"
    266 XEQ "E"
    267 XEQ "K"
    268 "I"
    269 CLX
    270 SEEKPTA
    271 CLA
    272 ARCL 05
    273 RCL 14
    274 POSA
    275 X≠0?
    276 X>0?
    277 GTO 13

    278♦LBL 05
    279 POSFL
    280 X<0?
    281 GTO 14
    282 SF 10
    283 GETREC
    284 SIGN
    285 CHS
    286 AROT
    287 ATOX
    288 RCL 19
    289 X≠Y?
    290 GTO 05
    291 "├!"
    292 DELREC
    293 INSREC
    294 CLA
    295 ARCL 05
    296 GTO 05

    297♦LBL 06
    298 FC?C 07
     299 GTO ""
    300 RCL 19
    301 12
    302 RCL 00
    303 RCL 01
    304 -
    305 STO 01
    306 "I"
    307 CLX
    308 SEEKPTA

    309♦LBL 07
    310 SF 25
    311 GETREC
    312 FC?C 25
     313 GTO ""
    314 SIGN
    315 CHS
    316 AROT
    317 RDN
    318 ATOX
    319 X=Y?
    320 GTO 07
    321 RCL Z
    322 X=Y?
    323 "├▮"
    324 X≠Y?
    325 "├&"
    326 DELREC
    327 INSREC
    328 RDN
    329 GTO 07

    330♦LBL 14
    331 FC? 10
    332 GTO 06
    333 "I"
    334 CLX
    335 SEEKPTA
    336 GTO 13

    337♦LBL 08
    338 CF 09
    339 "I"
    340 FLSIZE

    341♦LBL 13
    342 "&"
    343 FS? 10
    344 "!"
    345 POSFL
    346 X<0?
    347 GTO 06
    348 INT
    349 SEEKPT
    350 GETREC
    351 ARCL 30
    352 GETAS
    353 CLX
    354 STO 04

    355♦LBL 09
    356 "A"
    357 SF 25
    358 SEEKPTA
    359 FS?C 25
    360 GTO 09
    361 SIGN
    362 ST+ 01
    363 35
    364 GTO 14

    365♦LBL 09
    366 GETREC
    367 SIGN
    368 ALENG
    369 X=Y?
    370 SF 09
    371 RCL 14
    372 ATOX
    373 X=Y?
    374 GTO 13
    375 RCL 04
    376 "V"
    377 SF 25
    378 SEEKPTA
    379 GETREC
    380 FC?C 25
    381 GTO 09
    382 CLX
    383 ATOX
    384 X≠Y?
    385 GTO 09
    386 ALENG
    387 X=0?
    388 GTO 13
    389 FC?C 09
    390 GTO 12

    391♦LBL 09
    392 FC? 10
    393 GTO 08
    394 RCL 19

    395♦LBL 14
    396 "I"
    397 RCLPTA
    398 RCL 06
    399 -
    400 SEEKPT
    401 SIGN
    402 DELCHR
    403 X<>Y
    404 CLA
    405 XTOA
    406 INSCHR
    407 GTO 08

    408♦LBL 13
    409 FC?C 09
    410 GTO 12
    411 SIGN
    412 RCL 04
    413 INT
    414 +
    415 STO 04
    416 GTO 09

    417♦LBL 12
    418 RCL 06
    419 ST+ 04
    420 RCL 04
    421 GTO 09
```

Bild 3-8 Listing a

Besonderheiten

Da die Sätze auf der Datencassette unter ihren Datensatznamen gespeichert sind, ist die Selektion über das entsprechende Feld (NACHNAME1) besonders effizient. Deshalb werden in den Zeilen 272 bis 296 erst die Datensätze vorausgewählt, deren Datensatznamen passen. Bei der Abklärung der restlichen Feldinhalte werden nur noch diese vorselektierten Sätze berücksichtigt. Wurde das Suchkriterium für den Datensatznamen nicht explizit angegeben, also entweder gar nicht oder mit einem ? unter den für den Datensatznamen relevanten Zeichen eingegeben, muß jeder Datensatz der Cassette mit den Suchbegriffen verglichen werden.

Modifikation

250, 299, 313 | GTO "A" - Name des residenten Programmfiles

3.1.5 ä - Ändern

Aufgabe

Mit dem Modul **ä** werden die Inhalte der ausgewählten Datensätze abgeändert. Dies kann auf zwei Arten geschehen:

- AUTOMATISCH: die Änderungen - sie können bei Zahlen auch relativ angegeben werden - werden einmal abgefragt (Modul **E**) und bei allen ausgewählten Datensätzen einheitlich durchgeführt. Nicht zu ändernde Datensatzfelder oder einzelne Zeichenpositionen werden - ähnlich wie bei der Auswahl (vgl. 3.1.4) - mit ? von der Änderung ausgenommen. Vor der Definition der Änderungen sind alle Datensatzfelder mit ? initialisiert. Es werden weder die programmierten Abkürzungen ersetzt noch die Plausibilität überprüft.

- nicht AUTOMATISCH: die ausgewählten Datensätze werden sukzessive wie bei der Neuaufnahme zur interaktiven Änderung angeboten (Modul **E**). Das Modul **P** (Plausiblitätskontrolle und programmierte Abkürzungen) wird aufgerufen.

In beiden Fällen ist die Änderung des Datensatznamens (NACHNAME1) nicht möglich. Hierzu ist der betreffende Datensatz zu löschen und mit neuem Datensatznamen einzugeben.

Moduleigenschaften

aufrufendes Modul:	**m**	
aufgerufene Module:	**m D E F J P**	
working file:	I	
Datenregister:	04	Zeiger auf **AZEICHEN** und **VZEICHEN**
Flags:	07	RELATIV gewählt
	09	**AZEICHEN** ist letztes des Feldes

Besonderheiten

Bei automatischen relativen Änderungen werden die Zahlenwerte mit ANUM und INT - bei FIX_0, CF_28 und CF_29 - ermittelt. Solange immer ganze Zahlen verwendet werden, treten keine Konvertierungsprobleme auf. Enthält ein zu änderndes Feld keine Zahl, bleibt es unverändert. Es wird jeweils nur der erste numerische Wert eines Feldes geändert und der restliche Feldinhalt gelöscht.

Bei der manuellen Änderung wird ein Datensatz nur dann auf die Cassette zurückgeschrieben, wenn er tatsächlich verändert wurde.

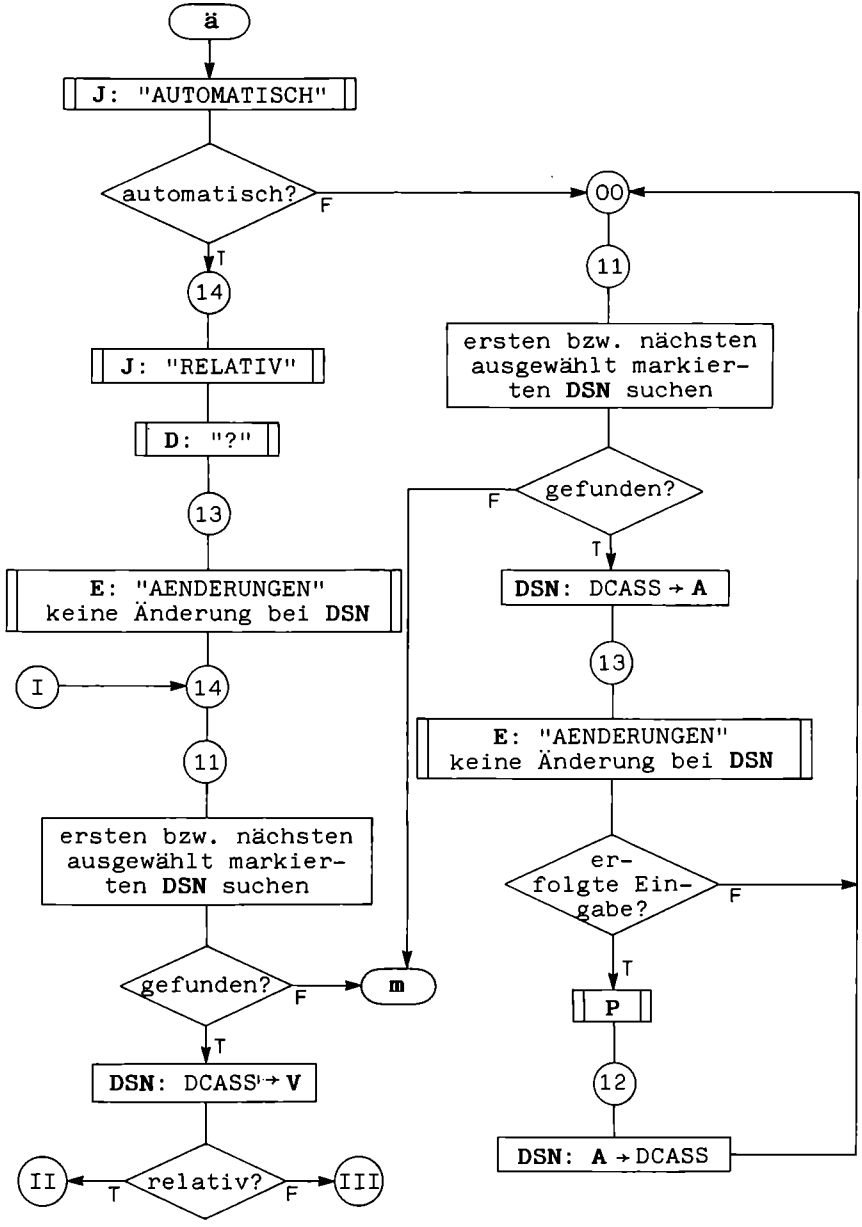


Bild 3-9/1 Ablaufplan ä DCASS: Datencassette

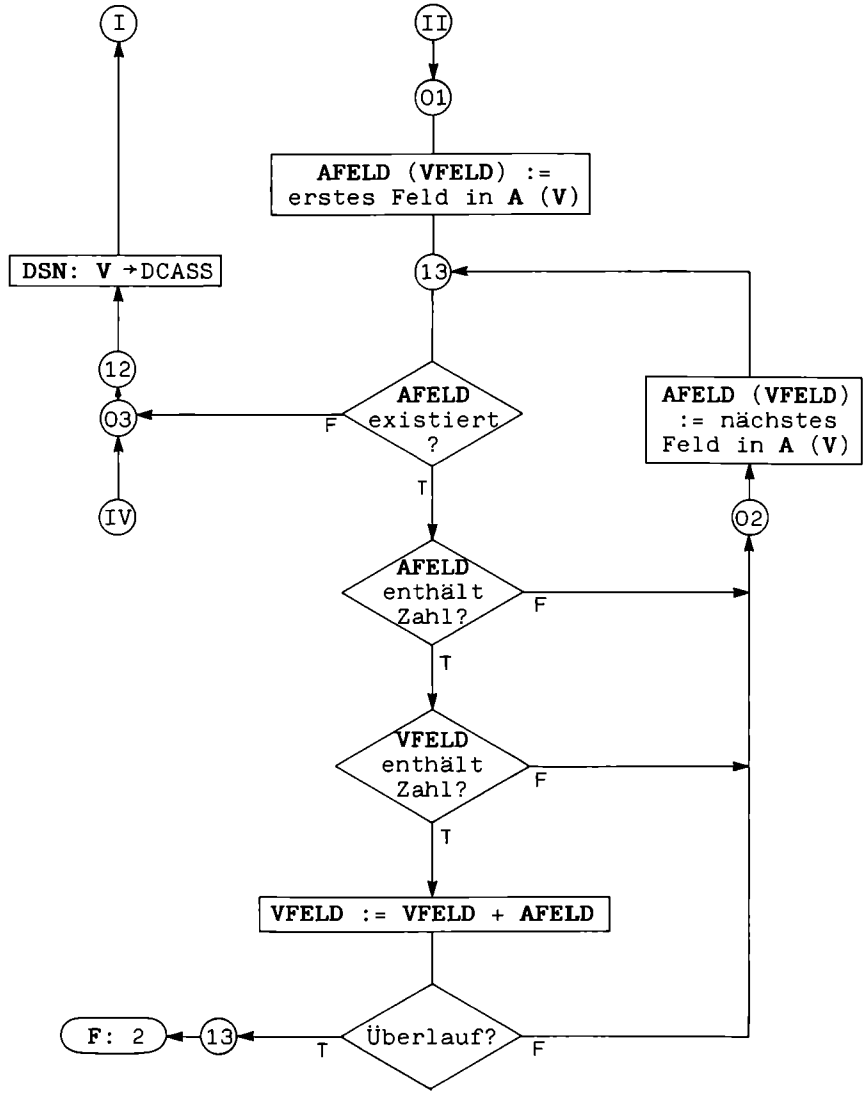


Bild 3-9/2 Ablaufplan ä

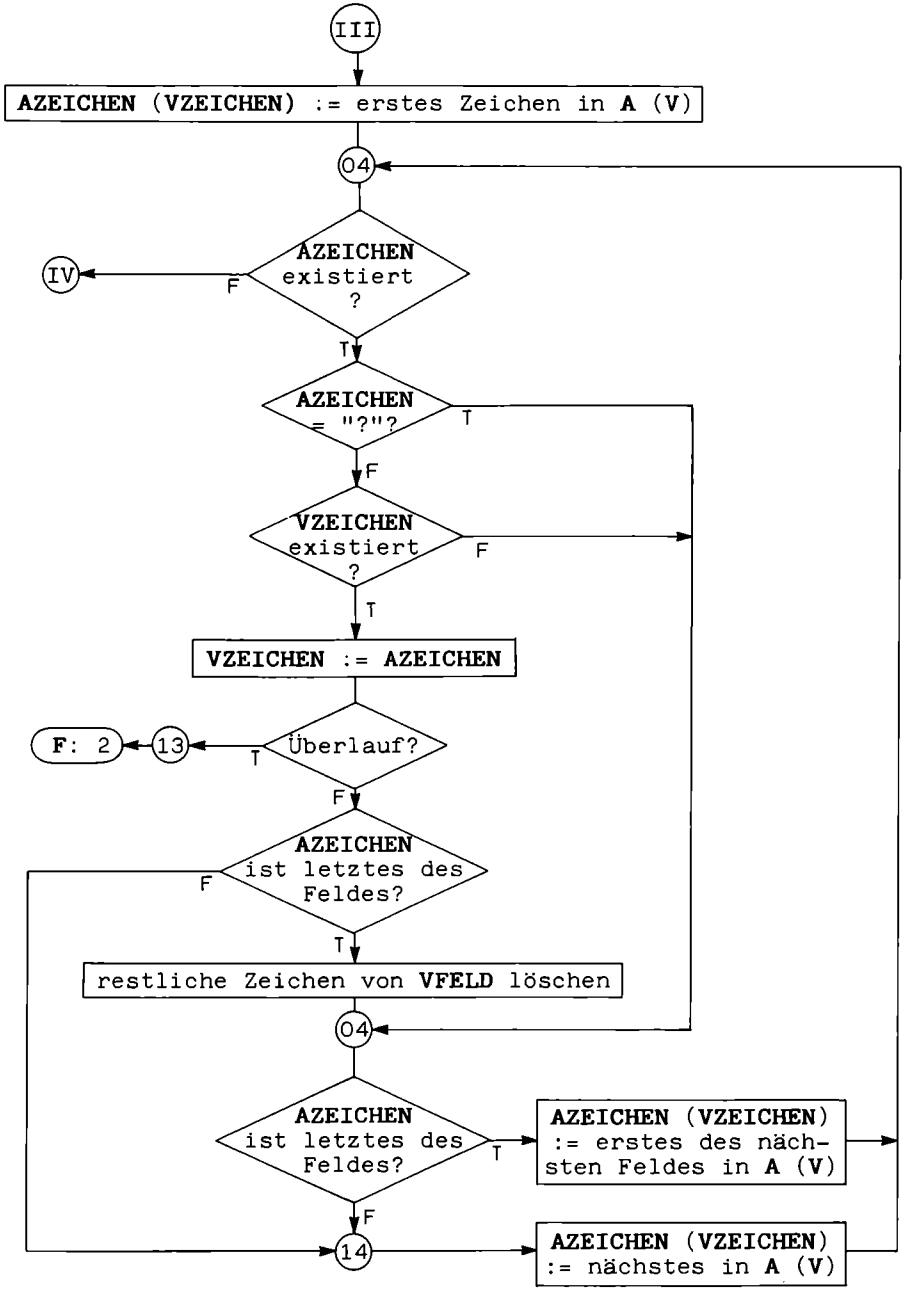


Bild 3-9/3 Ablaufplan ä

```
 75♦LBL "ä"
 76 "I"
 77 CLX
 78 SEEKPTA
79 "AUTOMATISCH"
 80 XEQ "J"
 81 X=Y?
 82 GTO 14

 83♦LBL 00
 84 XEQ 11
 85 ARCL 28
 86 GETAS
 87 XEQ 13
 88 FC?C 05
 89 GTO 00
 90 XEQ "P"
 91 RCL 09
 92 XEQ 12
 93 GTO 00

 94♦LBL 14
 95 "RELATIV"
 96 XEQ "J"
 97 X=Y?
 98 SF 07
 99 RCL 14
100 XEQ "D"
101 XEQ 13
102 GTO 14

103♦LBL 13
104 "AENDERUNGEN"
105 SF 06
106 XEQ "E"
107 CF 06
108 RTN

109♦LBL 01
110 "A"
111 SEEKPTA
112 GTO 13

113♦LBL 02
114 "A"
115 RCLPTA

116♦LBL 13
117 SF 25
118 GETREC
119 FC?C 25
120 GTO 03
121 ANUM
122 FC?C 22
123 GTO 02
124 INT
125 RCLPT
126 INT
127 "Y"
128 SEEKPTA
129 GETREC
130 X<>Y
131 ANUM
132 FC?C 22
133 GTO 02
134 INT
135 +
136 CLA
137 ARCL X
138 RDN
139 SEEKPT
140 DELREC
141 SF 25
142 INSREC
143 FC?C 25
144 GTO 13
145 GTO 02

146♦LBL 03
147 RCL 10
148 XEQ 12

149♦LBL 14
150 XEQ 11
151 ARCL 30
152 GETAS
153 CLX

154 FS? 07
155 GTO 01
156 STO 04

157♦LBL 04
158 "A"
159 SF 25
160 SEEKPTA
161 FC?C 25
162 GTO 03
163 GETREC
164 SIGN
165 ALENG
166 X=Y?
167 SF 09
168 RCL 14
169 ATOX
170 X=Y?
171 GTO 04
172 RCL 04
173 "Y"
174 SF 25
175 SEEKPTA
176 FC?C 25
177 GTO 04
178 X<>Y
179 CLA
180 XTOA
181 SF 25
182 INSCHR
183 FC?C 25
184 GTO 13
185 SIGN
186 DELCHR
187 FC? 09
188 GTO 14
189 RCLPT
190 SF 25
191 SEEKPT
192 FC?C 25
193 GTO 04
194 GETREC
195 SEEKPT
196 ALENG
197 DELCHR

198♦LBL 04
199 FC?C 09
200 GTO 14
201 SIGN
202 RCL 04
203 INT
204 +
205 STO 04
206 GTO 04

207♦LBL 13
208 2
209 GTO "F"

210♦LBL 14
211 RCL 06
212 ST+ 04
213 RCL 04
214 GTO 04

215♦LBL 12
216 "I"
217 RCLPTA
218 INT
219 SEEKPT
220 CLA
221 ARCL IND Y
222 ARCLREC
223 SAVEAS
224 RTN

225♦LBL 11
226 "I"
227 FLSIZE
228 "#"
229 POSFL
230 X<0?
 231 GTO ..
232 INT
233 SEEKPT
234 GETREC
235 RTN
```

Bild 3-10 Listing ä

Modifikation

231 |GTO_"A" - Name des residenten Programmfiles

3.1.6 c - Cassettenwechsel

Aufgabe

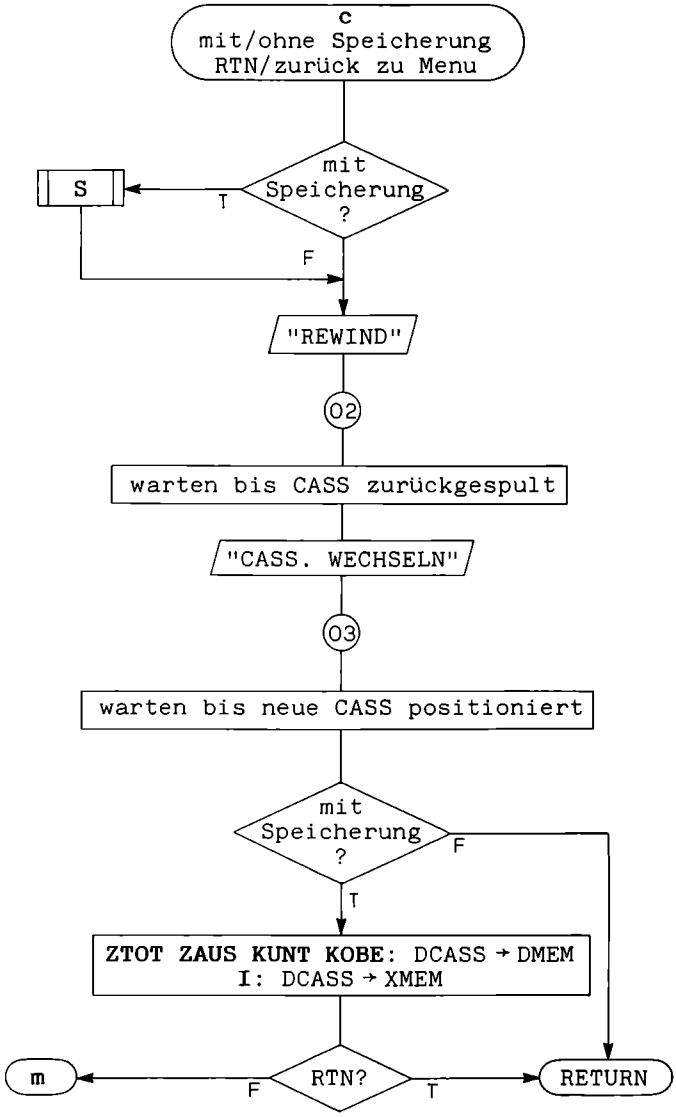
Das Modul **c** dient zum programmgesteuerten Wechsel der eingelegten Cassette. Mit REWIND und CASS._WECHSELN wird der Benutzer zu den entsprechenden Aktionen aufgefordert.

Moduleigenschaften

aufrufende Module:	**m IN UP K**	
aufgerufene Module:	**m S**	
Parameter IN:	Flag 07	ohne Schreib-/Leseoperationen
	Flag 10	Rücksprung mit RTN
working file:	I (nur wenn Flag 07 gelöscht ist)	

Besonderheiten

Das Modul hat einen ambivalenten Charakter: es kann einerseits als Anwendermodul vom Menu aus aufgerufen werden, wird jedoch andererseits auch als internes Modul (von **IN**, **UP** und **K**) verwendet. Je nach Status des Flags 10 erfolgt ein Rücksprung zum Menu (GTO_"A") oder zum aufrufenden Modul (RTN). Ist Flag 07 gelöscht, werden das Inhaltsverzeichnis **I** sowie die Zahlen **ZTOT** und **ZAUS** auf die "alte" Cassette geschrieben und - neben **KUNT** und **KOBE** - von der "neuen" Cassette geladen. Im anderen Falle (Aufruf durch die Module **IN** und **UP**) ist nur ein Rücksprung mit RTN möglich.



(D)CASS: (Daten-)Cassette
DMEM : Datenspeicher
XMEM : Extended Memory

Bild 3-11 Ablaufplan c

```
 79*LBL "c"
 80 FC? 07
 81 XEQ "S"
 82 "REWIND"
 83 AVIEW
 84 BEEP
 85 PSE
 86 PSE
 87 CLX
 88 X<>F
 89 SIGN

 90*LBL 02
 91 INSTAT
 92 X≠0?
 93 GTO 02
94 "CASS. WECHSELN"
 95 AVIEW
 96 BEEP

 97*LBL 03
 98 INSTAT
 99 23
100 X≠Y?
101 GTO 03
102 CLD
103 LASTX
104 X<>F
105 FS?C 07
106 RTN
 107 ""
108 3 E-3
109 SEEKR
110 READRX
111 "I"
112 SF 25
113 GETAS
114 CF 25
115 FC?C 10
 116 GTO ""
117 RTN
```

Bild 3-12 Listing c

Die Tätigkeiten des Benutzers am Cassettenlaufwerk werden mittels INSTAT kontrolliert. Das Laufwerk muß deshalb die erste Einheit der IL-Schleife sein.

Die Anweisungen 112 SF_25 und 114 CF_25 vermeiden die Fehlermeldung FL_TYPE_ERR beim GETAS eines noch leeren Inhaltsverzeichnisses **I**, das ja in diesem Fall noch den Typ DA-File hat.

Modifikation

107	"a" - Name des Datenregisterfiles
116	GTO_"A" - Name des residenten Programmfiles

3.1.7 d - Drucken

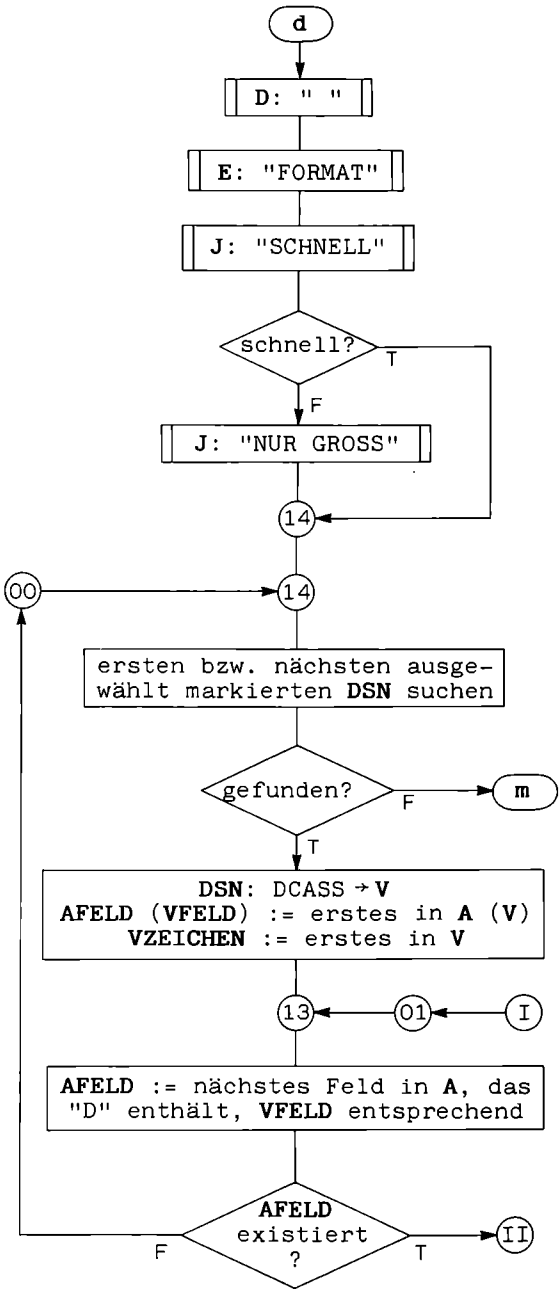
Aufgabe

Das Modul **d** gibt die ausgewählten Datensätze auf dem Drucker aus. Das Format wird mit der Satzeingabe (Modul **E**) definiert. Soll ein Feld gedruckt werden, ist an der ersten Stelle des entsprechenden Formatfeldes ein D einzugeben. Folgende Formatanweisungen können zusätzlich angegeben werden:

N	das Feld wird in eine neue Zeile gedruckt
1, 2, ...	das Feld wird nach 1, 2, ... Zeilenvorschüben gedruckt.

An zweiter oder dritter Stelle können stehen:

I	die Formatanweisung wird auch bei leerem Feld ausgeführt
d	der Feldinhalt wird in Breitschrift (SF_12) ausgegeben.



Die Reihenfolge der Datensatzfelder kann nicht verändert werden.

Die Ersatzdarstellungen für ß und die Umlaute werden umgewandelt.

Mit SCHNELL werden die Datensatzinhalte so ausgedruckt, wie sie in der Anzeige dargestellt werden. Da hierbei die Ersatzdarstellungen und die Präfixe für Großbuchstaben nicht berücksichtigt werden, erfolgt die Ausgabe wesentlich schneller.

Mit dem IL-Drucker HP-82162A wird im PARSE-Mode gedruckt. Zu Beginn der Dateibearbeitung (Modul **m**) wird die entsprechende Escape-Sequenz an den Drucker geschickt.

DCASS: Datencassette

Bild 3-13/1 Ablaufplan d

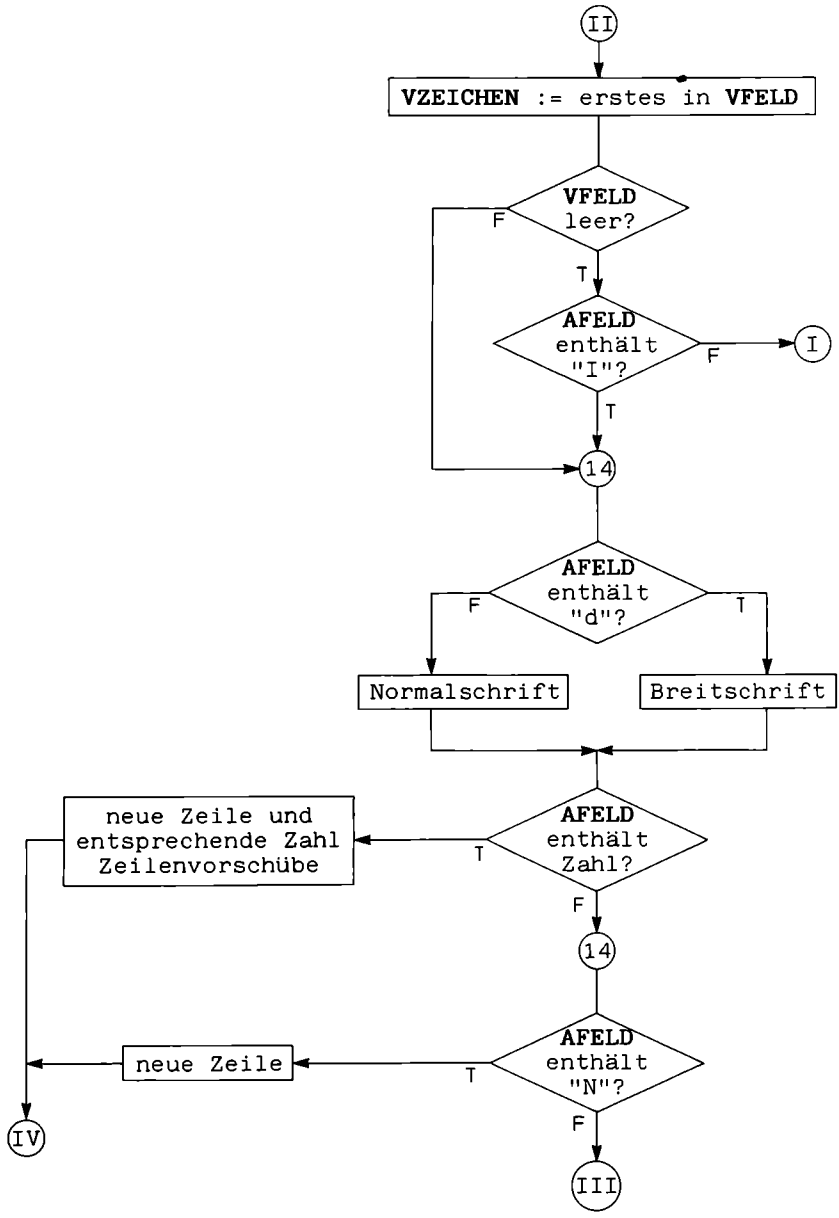


Bild 3-13/2 Ablaufplan d

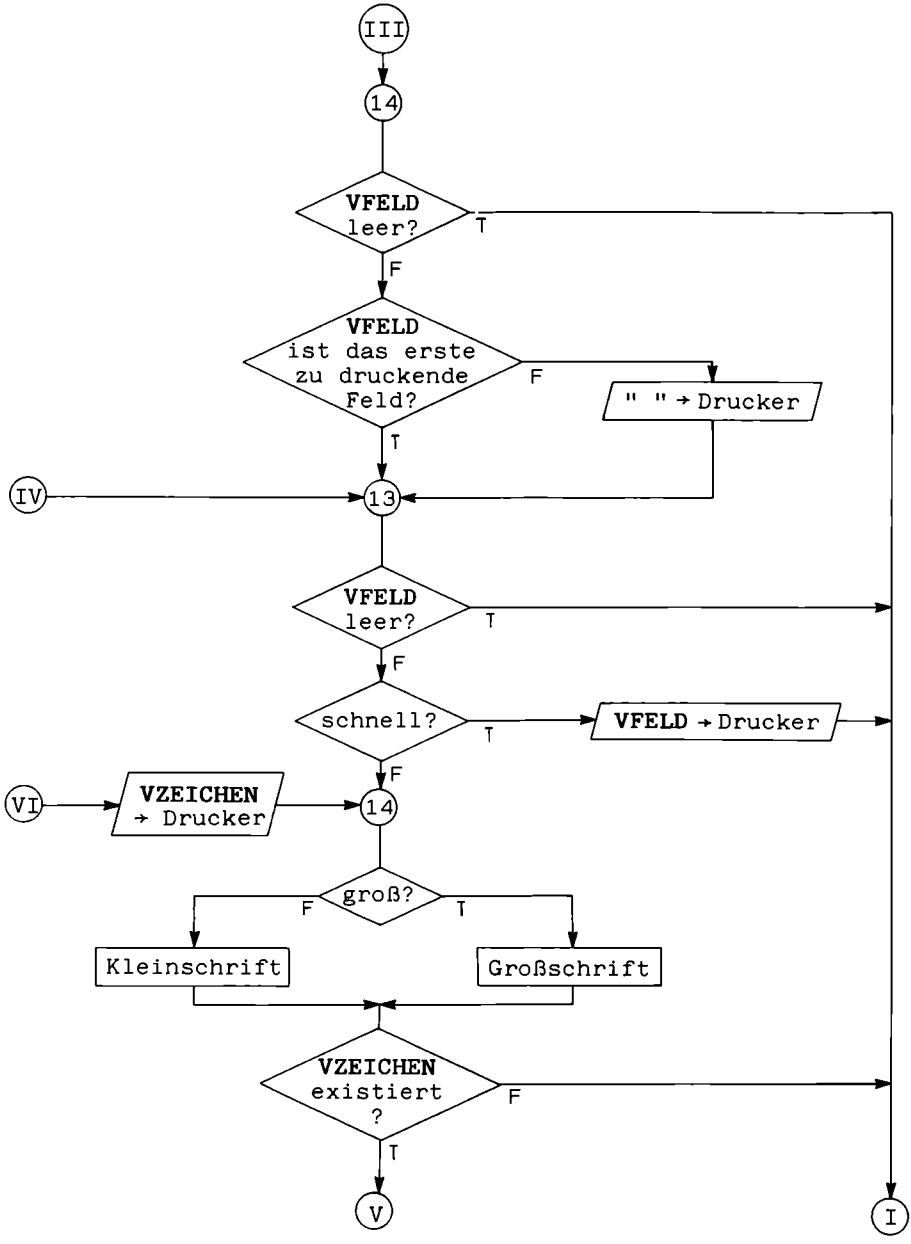


Bild 3-13/3 Ablaufplan d

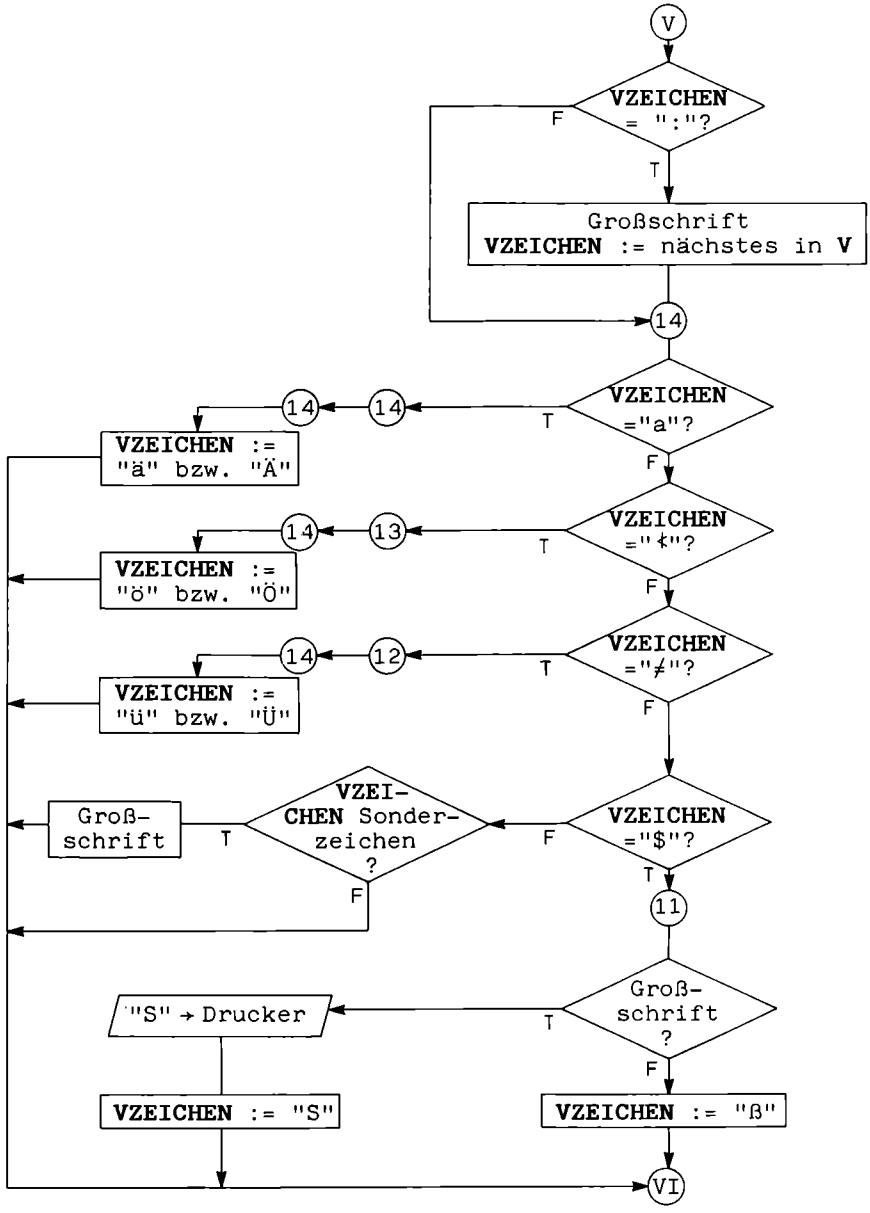


Bild 3-13/4 Ablaufplan d

```
 02♦LBL "d"
 03 RCL 11
 04 XEQ "D"
 05 "FORMAT"
 06 XEQ "E"
 07 "SCHNELL"
 08 XEQ "J"
 09 X=Y?
 10 SF 10
 11 X=Y?
 12 GTO 14
13 "NUR GROSS"
 14 XEQ "J"
 15 X=Y?
 16 SF 07

 17♦LBL 14
 18 SF 21
 19 "I"
 20 CLX
 21 SEEKPTA
 22 PRBUF
 23 SF 09
 24 GTO 14

 25♦LBL 00
 26 "I"
 27 RCLPTA

 28♦LBL 14
 29 "B"
 30 POSFL
 31 X<0?
 32 GTO 14
 33 INT
 34 SEEKPT
 35 GETREC
 36 ARCL 30
 37 GETAS
 38 CLX
 39 STO 05
 40 GTO 13

 41♦LBL 14
 42 PRBUF
 43 PRBUF
 44 PRBUF
 45 PRBUF
 46 PRBUF
 47 GTO ""
 48♦LBL 01
 49 CLX
 50 SIGN
 51 ST+ 05
 52 RCL 05

 53♦LBL 13
 54 CF 08
 55 "A"
 56 SF 25
 57 SEEKPTA
 58 "D"
 59 POSFL
 60 FS?C 25
 61 X<0?
 62 GTO 00
 63 STO 05
 64 GETREC
 65 ASTO 04
 66 "V"
 67 SEEKPTA
 68 GETREC
 69 RCL 11
 70 ATOX
 71 X=Y?
 72 SF 08
 73 RCL 15
 74 X=Y?
 75 SF 08
 76 CLA
 77 ARCL 04
 78 FC? 08
 79 GTO 14
 80 RCL 24
 81 POSA
 82 X<0?
 83 GTO 01

 84♦LBL 14
 85 RCL 27
 86 POSA
 87 SF 12
 88 X<0?
 89 CF 12
 90 ANUM
 91 FC?C 22
 92 GTO 14
 93 PRBUF
 94 RCL b
 95 PRBUF
 96 DSE Y
 97 STO b
 98 GTO 13

 99♦LBL 14
100 RCL 25
101 POSA
102 X<0?
103 GTO 14
104 PRBUF
105 GTO 13

106♦LBL 14
107 FS? 08
108 GTO 01
109 RCL 11
110 FC?C 09
111 ACCHR

112♦LBL 13
113 FS? 08
114 GTO 01
115 "V"
116 RCL 05
117 SEEKPTA
118 GETREC
119 FC? 10
120 GTO 14
121 ACA
122 GTO 01

123♦LBL 02
124 ACCHR

125♦LBL 14
126 CF 13
127 FC? 07
128 SF 13
129 RCL 20
130 ATOX
131 X=0?
132 GTO 01
133 X≠Y?
134 GTO 14
135 CF 13
136 ATOX

137♦LBL 14
138 RCL 26
139 X=Y?
140 GTO 14
141 CLX
142 RCL 09
143 X=Y?
144 GTO 13
145 CLX
146 RCL 10
147 X=Y?
148 GTO 12
149 CLX
150 RCL 18
151 X=Y?
152 GTO 11
153 CLX
154 RCL 15
155 X<>Y
156 X≠Y?
157 X>Y?
158 CF 13
159 RCL 21
160 X<>Y
161 X<Y?
162 CF 13
163 GTO 02

164♦LBL 14
165 21
166 GTO 14

167♦LBL 13
168 23
169 GTO 14

170♦LBL 12
171 25

172♦LBL 14
173 SIGN
174 FS? 13
175 ST+ L
176 LASTX
177 GTO 02

178♦LBL 11
179 5
180 FS? 13
181 GTO 02
182 83
183 ACCHR
184 GTO 02
```

Bild 3-14 Listing d

Moduleigenschaften

aufrufendes Modul:	**m**	
aufgerufene Module:	**m D E J**	
working file:	I	
Datenregister:	04	**AFELD**
	05	Zeiger auf **AFELD** und **VFELD**
Flags:	07	NUR GROSS gewählt
	08	**VFELD** ist leer
	09	erste Zeile des Datensatzes
	10	SCHNELL gewählt

Besonderheiten

Die Zeilen 043 bis 046 dürfen nicht durch ADV ersetzt werden, da sonst die letzten Zeilen der Ausgabe rechtsbündig gedruckt werden. Dieser Effekt tritt beim HP-82162A wegen seines relativ großen Buffers auf.

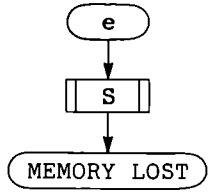
Modifikation

047 GTO_"A" - Name des residenten Programmfiles

3.1.8 e - Ende der Dateibearbeitung

Aufgabe

Das Modul **e** ermöglicht eine kontrollierte Beendigung der Dateibearbeitung. Das Inhaltsverzeichnis **I** sowie die Zahlen **ZTOT** und **ZAUS** werden auf die Cassette geschrieben. Der Programmlauf endet mit MEMORY_LOST.




```
75*LBL "e"
76 XEQ "S"
77 STO c
78 STOP
```

Bild 3-15 Ablaufplan e

Bild 3-16 Listing e

Moduleigenschaften

aufrufendes Modul: **m**
aufgerufenes Modul: S

3.1.9 i - Inhaltsverzeichnis

Aufgabe

Das Modul **i** gibt die nicht gelöschten Datensatznamen der eingelegten Cassette mit ihren Markierungen aus. Dieses Inhaltsverzeichnis kann auch ausgedruckt werden (AUSDRUCK).

Moduleigenschaften

aufrufendes Modul: **m A**
working file: I

Besonderheiten

Der **CODE** wird in eine Zahl zwischen 45 und 255 umgewandelt, die ausgewählt-Markierung mit AUS ausgegeben.

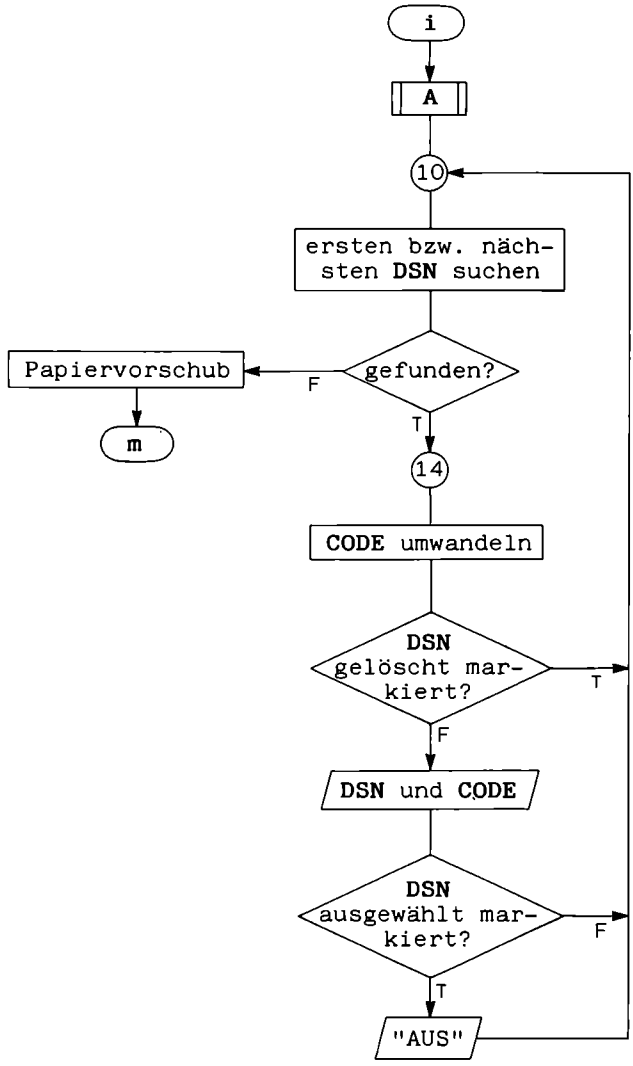


Bild 3-17 Ablaufplan i

```
448*LBL "i"
449 XEQ "A"
450 "I"
451 CLX
452 SEEKPTA

453*LBL 10
454 SF 25
455 GETREC
456 CLD
457 FS?C 25
458 GTO 14
459 ADV
460 ADV
461 ADV
462 ADV
 463 GTO ..

464*LBL 14
465 6
466 AROT
467 44
468 ATOX
469 -
470 ARCL X
471 ATOX
472 12
473 X=Y?
474 GTO 10
475 CLX
476 35
477 X=Y?
478 "⊢ AUS"
479 FS? 21
480 PRA
481 FS? 21
482 GTO 10
483 TONE 0
484 AVIEW
485 PSE
486 GTO 10
```

Bild 3-18 Listing i

Modifikation

```
463       |GTO_"A" - Name des residenten Programmfiles
```

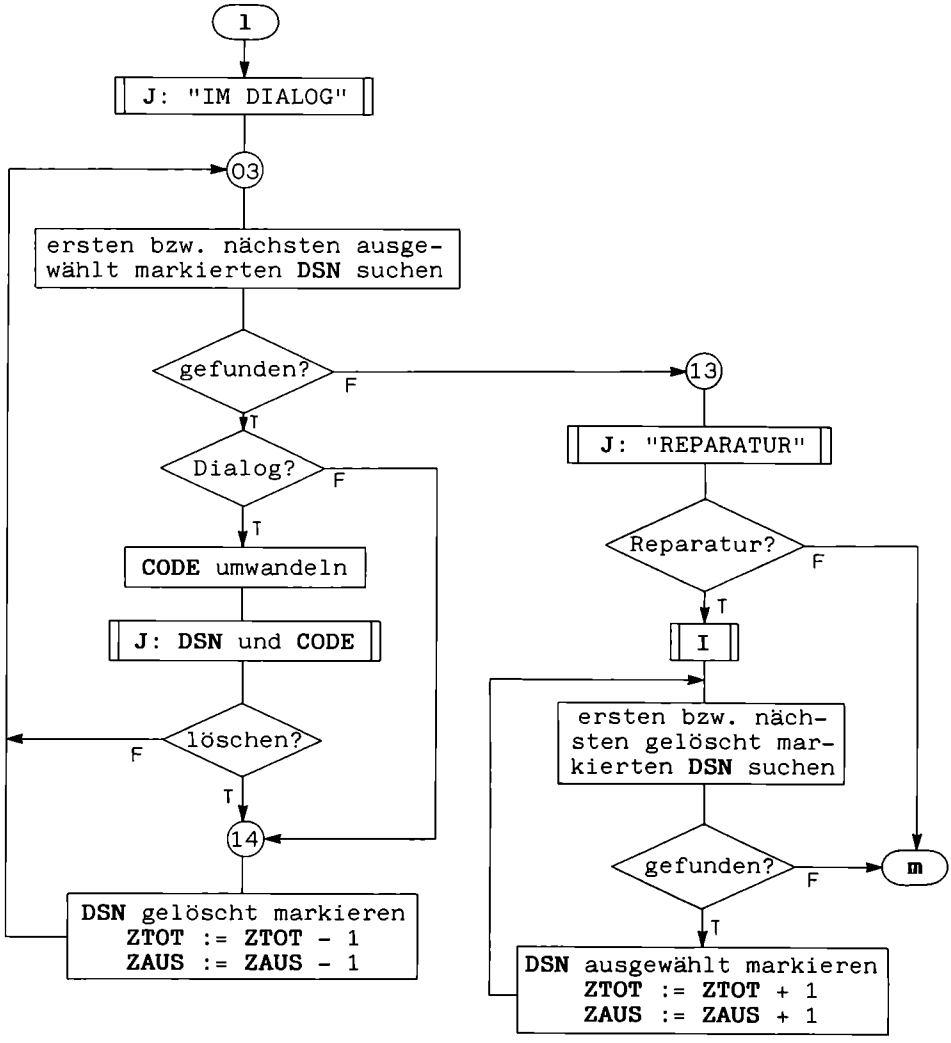


Bild 3-19 Ablaufplan 1

3.1.10 l - Löschen

Aufgabe

Mit **l** können ausgewählte Datensätze gelöscht werden - entweder alle automatisch oder einzelne interaktiv (IM DIALOG). Die Datensätze werden nur logisch gelöscht, d. h. der Datensatzname wird im Inhaltsverzeichnis I gelöscht markiert. Solange der Datensatz auf der Cassette nicht durch einen neuaufgenommenen überschrieben wurde, bleiben die gelöschten Daten - physisch - erhalten. Die REPARATUR-Routine nimmt **alle** seit der Initialisierung der Cassette gelöschten und noch nicht überschriebenen Datensätze wieder auf. Sie sind dann ausgewählt und können wieder gelöscht werden.

```
   185*LBL "l"
186 "IM DIALOG"
   187 XEQ "J"
   188 X=Y?
   189 SF 09
   190 "I"
   191 CLX
   192 SEEKPTA

   193*LBL 03
   194 "▮"
   195 POSFL
   196 FC?C 27
   197 X<0?
   198 GTO 13
   199 FC? 09
   200 GTO 14
   201 STO 04
   202 INT
   203 SEEKPT
   204 GETREC
   205 6
   206 AROT
   207 44
   208 ATOX
   209 -
   210 ARCL X
   211 ATOX
   212 XEQ "J"
   213 X≠Y?
   214 GTO 03
   215 RCL 04
   216 SEEKPT

   217*LBL 14
   218 SIGN
   219 DELCHR
   220 "µ"
   221 INSCHR
   222 ST- 00
   223 ST- 01
   224 GTO 03

   225*LBL 13
226 "REPARATUR"
   227 XEQ "J"
   228 X≠Y?
    229 GTO ""
   230 XEQ "I"
   231 CLX
   232 SEEKPT
   233 RCL b
   234 "µ"
   235 POSFL
   236 X<0?
    237 GTO ""
   238 SIGN
   239 DELCHR
   240 "▮"
   241 INSCHR
   242 ST+ 00
   243 ST+ 01
   244 X<>Y
   245 STO b
```

Bild 3-20 Listing l

Moduleigenschaften

aufrufendes Modul: **m**
aufgerufene Module: **m I J**
working file: I

Datenregister: Zeiger auf **DSN**

Flag: 09 IM DIALOG gewählt

Modifikation

229, 237 |GTO_"A" - Name des residenten Programmfiles

3.1.11 m - Menu

Aufgabe

Das Modul **m** zeigt die Kennung der eingelegten Cassette sowie die Funktionen des Systems an. Der Benutzer wählt eine aus, indem er JA eingibt. Alle Funktionen - außer ENDE, das den Programmlauf mit MEMORY_LOST beendet - kehren zum Menu zurück.

Läuft das Menu zweimal ohne Auswahl einer Funktion ab, werden der Rechner und die IL-Einheiten im STANDBY-Modus abgeschaltet. Ein Druck auf die ON-Taste reaktiviert das System. Mit der USER-Taste erreicht man den Anfang des Menus.

Moduleigenschaften

Modulname für Aufruf: Name des residenten Programmfiles
aufrufende Module: **BE a ä c d i l n p z F**
aufgerufene Module: **a ä c d e i l n p z J**
Flag: 08 erster Durchgang des Menus

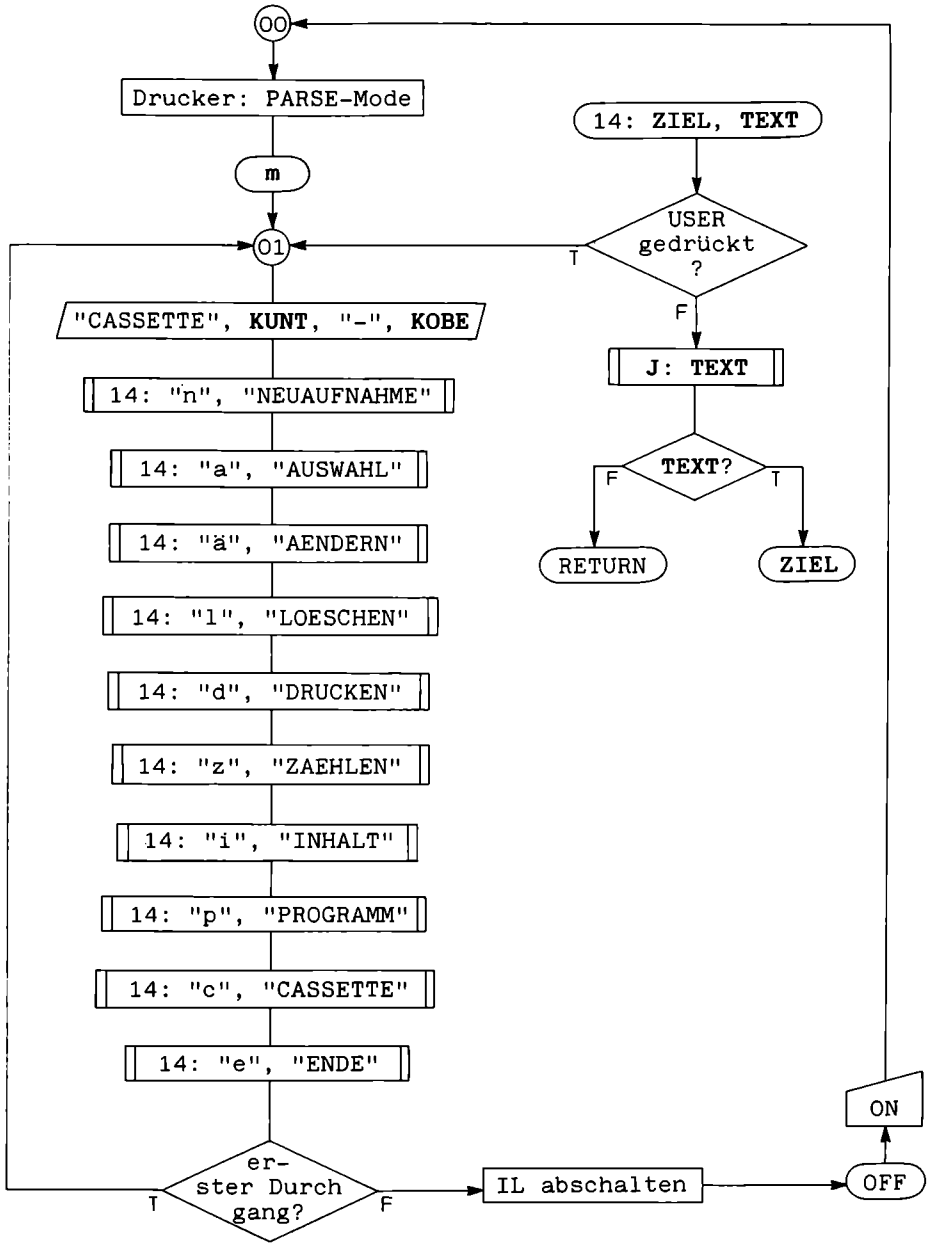


Bild 3-21 Ablaufplan m

```
 01 ""
02 READSUB

03♦LBL 00
04 2
05 SELECT
06 "E&k1H"
07 OUTA
08 SIGN
09 SELECT

 10♦LBL ""
11 RCL 31
12 STOFLAG
13 AON

14♦LBL 01
15 "CASSETTE "
16 RCL 03
17 XTOA
18 "⊦-"
19 RCL 02
20 XTOA
21 TONE 0
22 PSE
23 "nNEUAUFNAHME"
24 XEQ 14
25 "aAUSWAHL"
26 XEQ 14
27 "äAENDERN"
28 XEQ 14
29 "lLOESCHEN"
30 XEQ 14
31 "dDRUCKEN"
32 XEQ 14
33 "zZAEHLEN"
34 XEQ 14
35 "iINHALT"
36 XEQ 14
37 "pPROGRAMM"
38 XEQ 14
39 "cCASSETTE"
40 XEQ 14
41 "eENDE"
42 XEQ 14
43 FS?C 08
44 GTO 01
45 SF 11
46 PWRDN
47 OFF
48 GTO 00

49♦LBL 14
50 FS?C 27
51 GTO 01
52 ATOX
53 SIGN
54 XEQ "J"
55 X≠Y?
56 RTN
57 CF 27
58 CLA
59 LASTX
60 XTOA
61 ASTO X
 62 ""
63 GTO IND X

386♦LBL "l"
387♦LBL "d"
388♦LBL "p"
 389 "⊦"

390♦LBL "n"
391♦LBL "a"
392♦LBL "ä"
393♦LBL "z"
394♦LBL "i"
 395 "⊦"
 396 READP
397 GTO IND X
```

Bild 3-22 Listing m

Besonderheiten

Zu Beginn der Dateibearbeitung wird das transiente Programmfile **AA** mit 002 READSUB gelesen, ohne das residente File **A** zu überschreiben. Die transienten Programmfiles **AA** und **AAA** werden bei Bedarf mit 396 READP hinter das Programm **A** geladen und ersetzen dort das jeweils andere transiente File. Da globale Labels vom permanenten .END. aus in Richtung fallender Zeilennummern gesucht werden, werden die Labels der Zeilen 386 bis 394 nur erreicht, wenn das entsprechende Programmfile noch nicht geladen ist. Die folgende Sprunganweisung 397 GTO_IND_X verzweigt dann zum Beginn der gewählten Funktion.

Mit den Befehlen der Zeilen 004 bis 009 wird der PARSE-Mode eines eventuell angeschlossenen IL-Druckers eingeschaltet. Die Anweisungsfolge ist ohne Wirkung, wenn die IL-Schleife nur ein Gerät (nämlich das Cassettenlaufwerk) enthält.

Das Modul wird nicht mit **m**, sondern mit dem Namen des residenten Programmfiles (A) aufgerufen, an dessen Anfang es ja steht.

Modifikation

062	"A"
389,395	"├A"
001	"AA" - Name eines transienten Programmfiles
023-042, 386-394	Es können noch weitere Funktionsnamen bzw. Labels eingefügt werden

```
02♦LBL "n"
03 RCL 15
04 XEQ "D"
05 XEQ "I"
06 "DATEN"
07 XEQ "E"
08 XEQ "P"
09 XEQ "K"
10 CLX
11 "I"
12 SEEKPTA
13 RCL 12
14 STO 04
15 RCL b
16 CLA
17 ARCL 05
18 RCL 04
19 XTOA
20 POSFL
21 X<0?
22 GTO 14
23 CLX
24 SEEKPT
25 RCL Z
26 ISG 04
27 STO b
28  E
29 GTO "F"

30♦LBL 14
31 ST- 00
32 ST- 01
33 "µ"
34 POSFL
35 CLA
36 ARCL 05
37 RCL 04
38 XTOA
39 "├"
40 X<>Y
41 X<0?
42 GTO 14
43 INT
44 SEEKPT
45 ARCLREC
46 "├  ,"
47 RCL 17
48 AROT
49 RENAME
50 ASHF
51 ASHF
52 DELREC
53 INSREC
54 GTO 13

55♦LBL 14
56 SF 25
57 APPREC
58 FC? 25
59 GTO 12
60 RCL 11
61 CREATE
62 FC?C 25
63 GTO 14

64♦LBL 13
65 ARCL 13
66 RCL 17
67 AROT
68 SAVEAS
 69 GTO ""

70♦LBL 14
71 DELREC

72♦LBL 12
73 3
74 GTO "F"
```

Bild 3-23 Listing n

3.1.12 n - Neuaufnahme

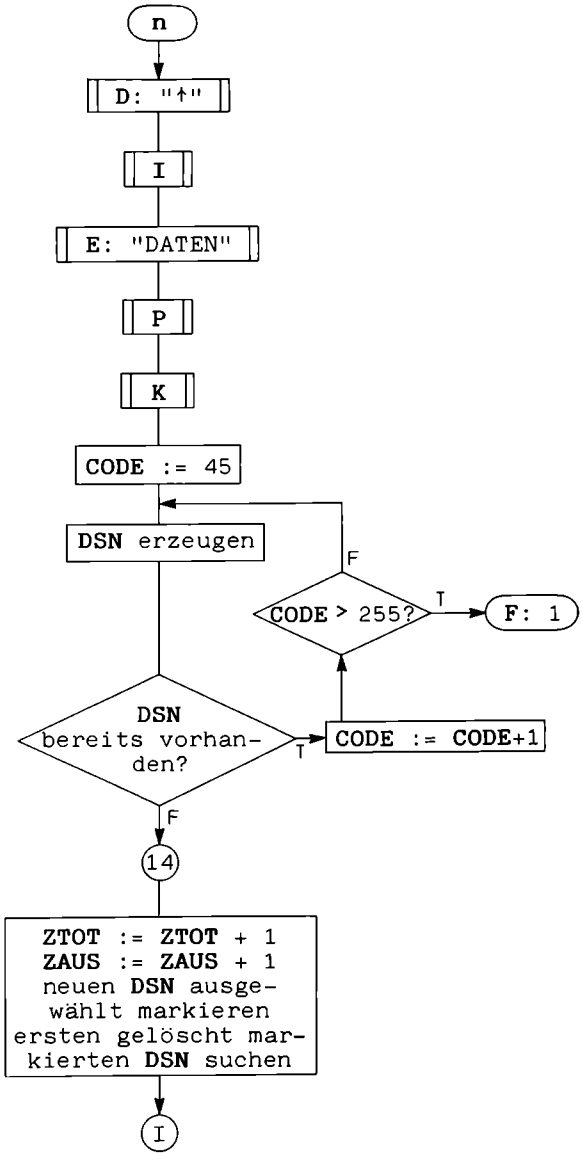


Bild 3-24/1 Ablaufplan n

Aufgabe

Mit dem Modul **n** werden Datensätze in die Datei aufgenommen. Dabei werden soweit möglich erst als gelöscht markierte Datensätze überschrieben. Neue Datensätze sind automatisch ausgewählt. Soll nach der Neuaufnahme nur mehr auf die neuen Datensätze zugegriffen werden, sind die Datensatznamen zu INITIALISIEREN. Plausibilitätskontrollen werden durchgeführt, programmierte Abkürzungen ersetzt (Modul **P**).

Die Felder des Datensatzes sind vor der Eingabe mit ↑ (Zeichen für noch zu ermittelnden Inhalt) initialisiert (Modul **D**).

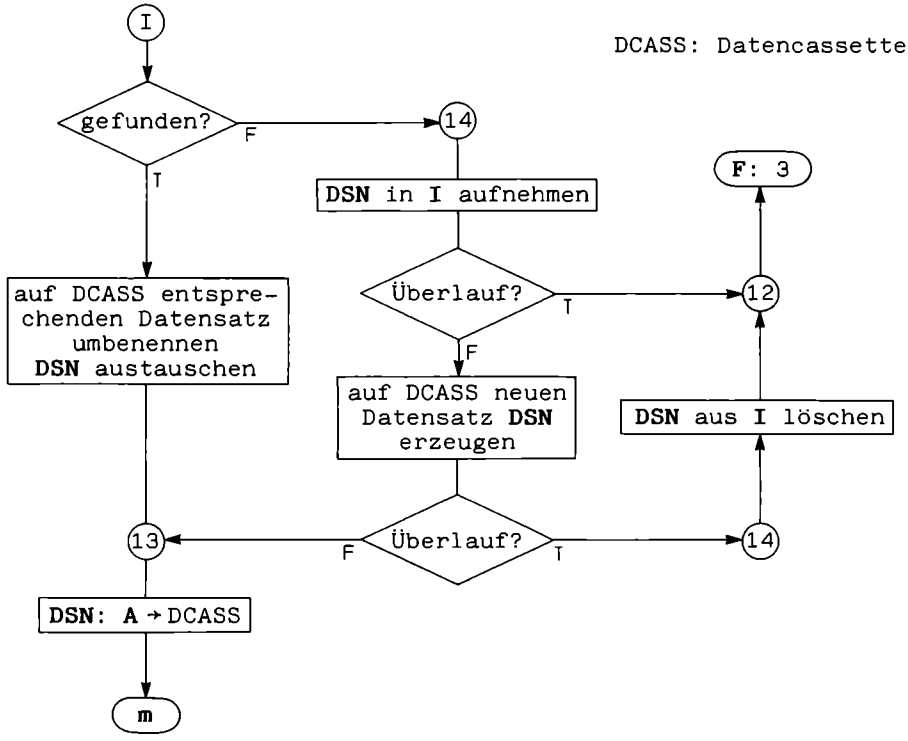


Bild 3-24/2 Ablaufplan n

Moduleigenschaften

aufrufendes Modul:	**m**
aufgerufene Module:	**m D E F I K P**
working file:	**A**
Datenregister:	04 **CODE**

Modifikation

069	GTO "A" - Name des residenten Programmfiles
060	RCL 11 - Größe SX (Länge eines Datensatzes im Extended Memory in Registern, vgl. 2.3.3)

3.1.13 p - Programmierbare Funktion

Aufgabe

Das Modul **p** kann vom Anwender programmiert werden.

Moduleigenschaften

aufrufendes Modul: m
aufgerufenes Modul: m

```
246↑LBL "p"
 247 GTO ..
248 ENTER↑
249 ENTER↑
250 ENTER↑

533 ENTER↑
534 ENTER↑
535 ENTER↑
536 ENTER↑
537 .END.
```

Bild 3-25 Listing p

Modifikation

Das Modul **p** kann auch nachträglich geändert und mit dem Modul **UP** auf Programm- und Datencassetten kopiert werden. Die Pufferbytes (ENTER↑) sichern eine konstante Länge (in Records) des entsprechenden transienten Files auf der Cassette.

3.1.14 z - Zählen

Aufgabe

Das Modul **z** gibt die Zahlen **ZTOT** (Anzahl der Datensätze der eingelegten Cassette) und **ZAUS** (Anzahl der ausgewählten Datensätze) über die Anzeige oder den Drucker (AUSDRUCK) aus.

Moduleigenschaften

aufrufendes Modul: m
aufgerufene Module: m A

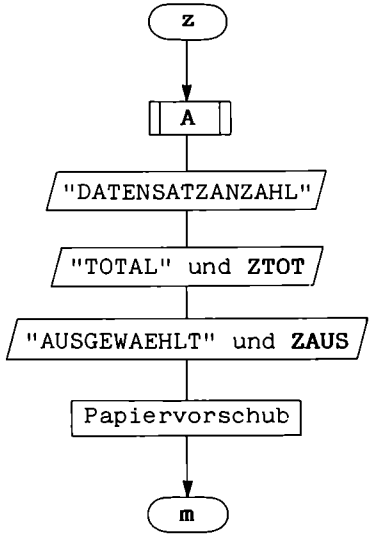


Bild 3-26 Ablaufplan z

```
422*LBL "z"
423 XEQ "A"
424 "DATENSATZANZAHL"
425 XEQ 14
426 "TOTAL: "
427 ARCL 00
428 XEQ 14
429 "AUSGEWAEHLT: "
430 ARCL 01
431 XEQ 14
432 CLD
433 ADV
434 ADV
435 ADV
436 ADV
437 GTO ""

438*LBL 14
439 FS? 21
440 PRA
441 FS? 21
442 RTN
443 TONE 0
444 AVIEW
445 PSE
446 PSE
447 RTN
```

Bild 3-27 Listing z

Modifikation

437 |GTO_"A" - Name des residenten Programmfiles

3.2 Interne Module

3.2.1 A - Ausdruck-Prompt

Aufgabe

Das Modul **A** fragt den Benutzer, ob die folgende Ausgabe (der Funktionen Inhaltsverzeichnis oder Zählen) gedruckt oder angezeigt werden soll und setzt das Flag 21 entsprechend.

Moduleigenschaften

aufrufende Module: **i z**
aufgerufenes Modul: **J**

3.2.2 c - Cassettenwechsel

Siehe 3.1.6.

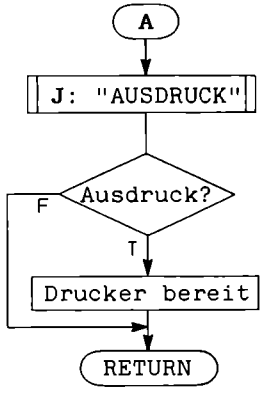


Bild 3-28 Ablaufplan A

```
487♦LBL "A"
488 "AUSDRUCK"
489 XEQ "J"
490 X=Y?
491 SF 21
492 .END.
```

Bild 3-29 Listing A

3.2.3 D - Datensatzinitialisierung

Aufgabe

D initialisiert alle Felder des ASCII-Files **A** im Extended Memory mit einem bestimmten Zeichen.

Moduleigenschaften

aufrufende Module:	**a ä d n**	
Parameter IN:	Reg. X	ASCII-Wert des Initialisierungszeichens (**TEXT**)
working file:	**A**	

Modifikation

360 |17 - Größe FA (Zahl der Datensatzfelder, vgl. 2.3.3)

3.2.4 E - Datensatzeingabe

Aufgabe

Das Modul **E** gibt erst den übergebenen Text aus. Dann werden die Feldinhalte des ASCII-Files **A** mit den entsprechenden Feld-

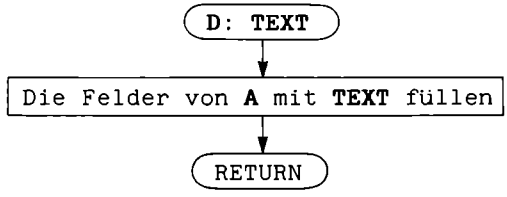


Bild 3-30 Ablaufplan D

```
355*LBL "D"
356 "A"
357 CLFL
358 CLA
359 XTOA
360 17
361 RCL b
362 APPREC
363 DSE Y
364 STO b
365 RTN
```

Bild 3-31 Listing D

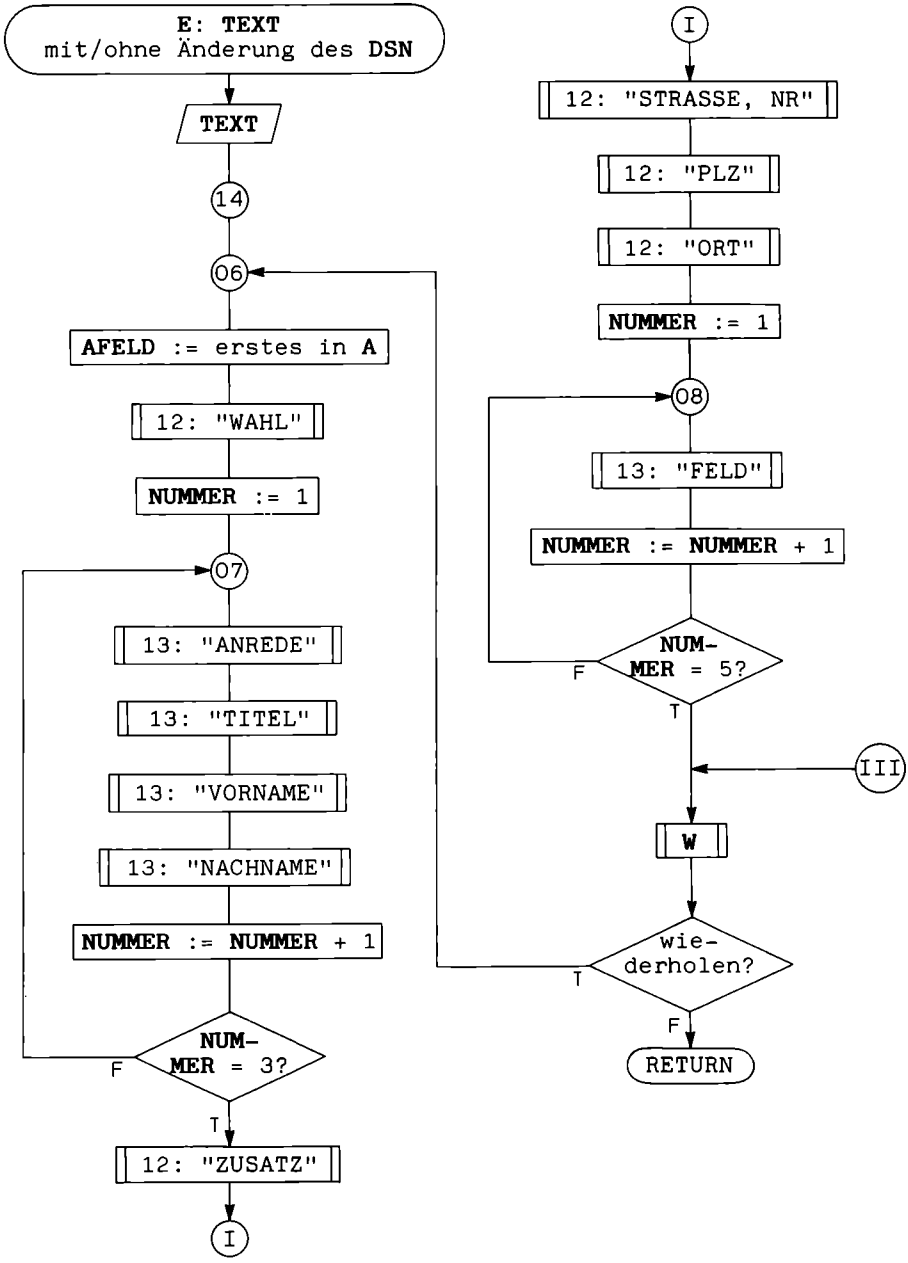


Bild 3-32/1 Ablaufplan E

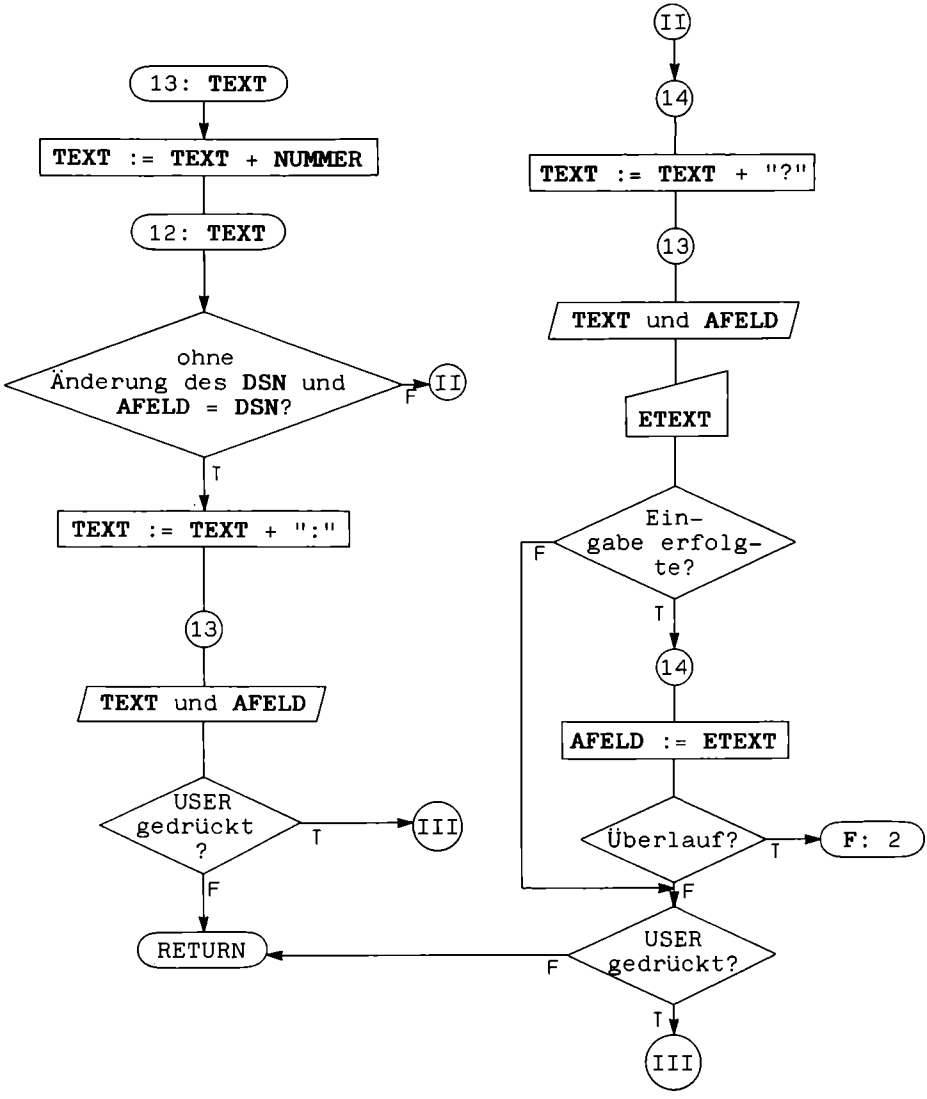


Bild 3-32/2 Ablaufplan E

bezeichnungen ausgegeben. Während ein Feld angezeigt wird, kann es durch eine Eingabe des Benutzers überschrieben werden. Ein leer eingegebenes Feld wird durch (Blank) ersetzt.

```
233♦LBL "E"
234 CF 05
235 TONE 0
236 PSE
237 "A"
238 FLSIZE
239 SF 09
240 GTO 14

241♦LBL 06
242 R↑
243 RCL 07
244 STO 05
245 SIGN
246 CHS
247 STO 04
248 X<>Y
249 "WAHL"
250 XEQ 12

251♦LBL 07
252 "ANREDE"
253 XEQ 13
254 "TITEL"
255 XEQ 13
256 "VORNAME"
257 XEQ 13
258 "NACHNAME"
259 XEQ 13
260 ISG 05
261 GTO 07
262 "ZUSATZ"
263 XEQ 12
264 "STRASSE, NR."
265 XEQ 12
266 "PLZ"
267 XEQ 12
268 "ORT"
269 XEQ 12
270 RCL 08
271 STO 05
272 X<>Y

273♦LBL 08
274 "FELD"
275 XEQ 13
276 ISG 05
277 GTO 08

278♦LBL 14
279 RCL b
280 RDN
281 FS?C 09
282 GTO 06
283 XEQ "W"
284 X=Y?
285 GTO 06
286 RTN

287♦LBL 13
288 ARCL 05
289♦LBL 12
290 CF 23
291  E
292 ST+ 04
293 RCL 16
294 RCL 04
295 SEEKPT
296 FS? 06
297 X≠Y?
298 GTO 14
299 "├: "
300 XEQ 13
301 TONE 0
302 AVIEW
303 PSE
304 CLD
305 GTO 12

306♦LBL 14
307 "├? "
308 XEQ 13
309 TONE 9
310 AVIEW
311 PSE
312 FC? 23
313 PSE
314 FC? 23
315 PSE
316 CLD
317 FS?C 23
318 XEQ 14
319♦LBL 12
320 X<>Y
321 FS?C 27
322 STO b
323 RTN

324♦LBL 14
325 SF 05
326 ALENG
327 X=0?
328 " "
329 RDN

330♦LBL 11
331 DELREC
332 SF 25
333 INSREC
334 FS?C 25
335 RTN
336 2
337 GTO "F"

338♦LBL 13
339 R↑
340 RCL b
341 ARCLREC
342 FC?C 17
343 RTN
344 ATOX
345 RDN
346 STO b
```

Bild 3-33 Listing E

Ein Druck auf die USER-Taste führt an das Ende der Satzeingabe, wo gefragt wird, ob sie wiederholt werden soll. Ein Flag bestimmt, ob der Datensatzname (NACHNAME1) abgeändert werden kann oder nur angezeigt werden soll. Ein weiteres Flag meldet dem aufrufenden Modul, ob eine Eingabe erfolgte.

Moduleigenschaften

aufrufende Module: **a ä d n P**
aufgerufene Module: **F W**

Parameter	IN:	ALPHA	**TEXT**
		Flag 06	keine Änderung des **DSN**
	OUT:	Flag 05	es erfolgte eine Eingabe
working file:		**A**	
Datenregister:		04	Zeiger auf **AFELD**
		05	**NUMMER**
Flag:		09	Rücksprungadresse unbestimmt

Besonderheiten

Die Bedienungserleichterung durch die USER-Taste erfordert unstrukturierte Rücksprünge aus einem Unterprogramm. Da die Rücksprungadressen vorheriger Unterprogrammaufrufe (u. a. des Modulaufrufs) nicht verloren gehen dürfen, muß der Programmzeiger im Statusregister b direkt manipuliert werden. Dies darf höchstens in der zweiten Unterprogrammebene stattfinden, da das Register b neben dem Programmzeiger nur zwei Rücksprungadressen vollständig speichert. Das Statusregister a enthält ebenfalls drei komplette Adressen, eine Rücksprungadresse ist auf beide Register aufgeteilt.

Das Sprungziel ist die Zeile 280 RDN. Beim Aufruf des Modules (Flag 09 gesetzt) wird mit 279 RCL_b das Statusregister b in den Stack geladen und im folgenden meist im T-Register aufbewahrt. Die Zwischenspeicherung in einem normalen Datenregister hätte die Verfälschung der Adressen durch Normalisierung zur Folge. Mit den Anweisungen 321 FS?C_27 322 STO_b werden die Adressen wieder in das Register b geladen, sofern die USER-Taste gedrückt wurde. Hinsichtlich Programmzeiger und Rücksprungadressen der beiden untersten Ebenen ist damit der Status der Zeile 279 STO_b restauriert. Der Programmlauf wird mit der Anweisung 280 RDN fortgesetzt (Flag 09 ist nun gelöscht).

In den Zeilen 338 bis 346 werden von der Feldbezeichnung vorne solange Zeichen abgeschnitten, bis der Rest mit dem entsprechenden Feldinhalt in das ALPHA-Register paßt (24 Zeichen). Die Zeilen 330 bis 337 gehören auch zu dem Modul **P**.

```
251♦LBL "E"
252 CF 05
253 TONE 0
254 PSE
255 "A"
256 FLSIZE
257 SF 09
258 GTO 14

259♦LBL 06
260 R↑
261 RCL 07
262 STO 05
263 SIGN
264 CHS
265 STO 04
266 X<>Y
267 "BOOTSNAME"
268 XEQ 12
269 "ANREDE"
270 XEQ 12
271 "VORNAME"
272 XEQ 12
273 "NACHNAME"
274 XEQ 12

275♦LBL 07
276 "P"
277 XEQ 13
278 ISG 05
279 GTO 07
280 "STRASSE, NR."
281 XEQ 12
282 "PLZ"
283 XEQ 12
284 "ORT"
285 XEQ 12
286 "BOOTSTYP"
287 XEQ 12
288 "LAENGE UEA"
289 XEQ 12
290 "BREITE UEA"
291 XEQ 12
292 "LAENGE WL"
293 XEQ 12
294 "TIEFGANG"
295 XEQ 12
296 "TAKELUNG"
297 XEQ 12
298 "SEGELFL."
299 XEQ 12
300 "LIEGEPL."
301 XEQ 12
302 "UNT."
303 XEQ 11
304 "UKW-"
305 XEQ 11
306 "IOR"
307 XEQ 12
308 RCL 08
309 STO 05
310 X<>Y

311♦LBL 08
312 "B"
313 XEQ 13
314 ISG 05
315 GTO 08
316♦LBL 14
317 RCL b
318 RDN
319 FS?C 09
320 GTO 06
321 XEQ "W"
322 X=Y?
323 GTO 06
324 RTN

325♦LBL 11
326 "ŀZEICHEN"
327 GTO 12

328♦LBL 13
329 "ŀ.FELD"
330 ARCL 05

331♦LBL 12
332 CF 23
333  E
334 ST+ 04
335 RCL 16
```

Bild 3-34 Listing E (YACHT-System)

Modifikation

249-277, 287-288	Hier können beliebige Feldbezeichnungen und Algorithmen zur platzsparenden Programmierung derselben eingesetzt werden, solange die Routine 289 LBL_12 höchstens die zweite Unterprogrammebene belegt und das zwischengespeicherte Statusregister im T-Register steht. **Bild 3-34** zeigt einen Ausschnitt aus dem komplizierteren **E**-Modul des **YACHT**-Systems (zur Verwaltung der Daten eines Yachtclubs).
312-315	die Anzahl der Anweisungsfolgen FC?_23 PSE entspricht dem Zeitraum, in dem ein Feld angezeigt wird und überschrieben werden kann.

3.2.5 F - Fehlermeldung

Aufgabe

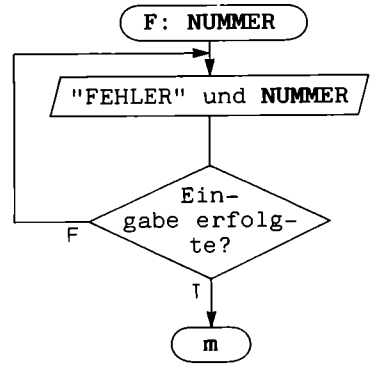


Bild 3-35 Ablaufplan F

Tritt während des Programmlaufes aufgrund der Daten ein Fehler auf, wird die Fehlernummer mit einem Alarmsignal ausgegeben. Nach der Eingabe eines beliebigen Zeichens wird das Programm im Menu fortgesetzt. Systemfehler (OUT_OF_PAPER, TRANSMIT_ERROR, BAT) werden nicht abgefangen. Die Fehlernummern haben folgende Bedeutungen:

- 1: zu viele gleiche Datensatznamen
  Es wurde versucht, mehr als 211 Datensätze mit gleichen Datensatznamen zu speichern.
  Abhilfe: Löschen eines Datensatzes dieser Gruppe oder Wahl eines anderen Datensatznamens.
- 2: Überschreitung der Datensatzkapazität
  Es wurde versucht, einen Datensatz mit mehr als SK Zeichen (Datensatzkapazität, vgl. 2.3.3) zu speichern.
  Abhilfe: Datenfeldinhalte abkürzen.

```
64*LBL "F"
65 SF 26
66 "FEHLER"
67 ARCL X
68 RCL b
69 TONE 7
70 TONE 2
71 PSE
72 FS?C 23
 73 GTO ..
74 STO b
```

Bild 3-36 Listing F

- 3: Überschreitung der Cassetten- bzw. Extended Memory-Kapazität
  Abhilfe: zweites X MEMORY Modul einsetzen oder Datensätze löschen.

Moduleigenschaften

aufrufende Module: **ä n E P**
aufgerufenes Modul: **m**
Parameter IN: Reg. X Fehlernummer (**NUMMER**)

Besonderheiten

Um bei der Fehlermeldung die Aufmerksamkeit des Benutzers zu sichern, werden zwei synthetische TONE-Anweisungen verwendet. Da sich manche Töne je nach Betriebssystem-Version unterscheiden, kann es sein, daß die hier programmierten TONEs nicht die gewünschte Wirkung zeigen - oder hören lassen.

Modifikation

073 |GTO_"A" - Name des residenten Programmfiles

3.2.6 I - Initialisierung des Inhaltsverzeichnisses

Aufgabe

Wird die Frage INITIALISIEREN? vom Benutzer positiv beantwortet, werden alle ausgewählt markierten Datensatznamen demarkiert und **ZAUS** auf 0 gesetzt.

Moduleigenschaften

aufrufende Module: **a l n**
aufgerufenes Modul: J
working file: I

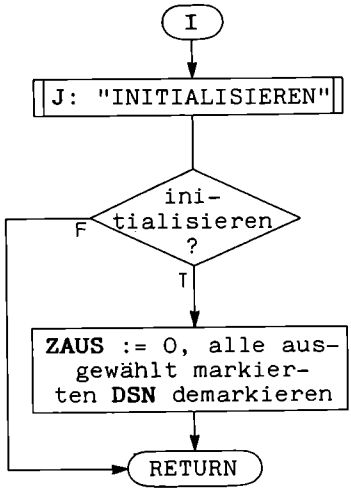


Bild 3-37 Ablaufplan I

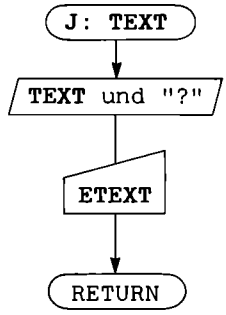
```
366*LBL "I"
367 "INITIALISIEREN"
368 XEQ "J"
369 X≠Y?
370 RTN
371 "I"
372 CLX
373 SEEKPTA
374 STO 01
375 RCL b
376 "&"
377 POSFL
378 X<0?
379 RTN
380 SIGN
381 DELCHR
382 "&"
383 INSCHR
384 X<>Y
385 STO b
```

Bild 3-38 Listing I

3.2.7 J - Ja/Nein-Prompt

Aufgabe

Das Modul J gibt den im ALPHA-Register übergebenen Text mit einem ? aus und erwartet die Eingabe des Benutzers. Die ersten sechs Zeichen des eingegebenen Textes werden in das X-Register, JA in das Y-Register gespeichert. Das aufrufende Modul kann die Wahl des Benutzers durch die Vergleiche X=Y? oder X≠Y? ermitteln.




```
400◆LBL "J"
401 "├?"
402 TONE 7
403 AVIEW
404 CLA
405 PSE
406 FC? 23
407 PSE
408 FC?C 23
409 PSE
410 CLD
411 ASTO X
412 "JA"
413 ASTO Y
414 .END.
```

Bild 3-39 Ablaufplan J

Bild 3-40 Listing J

Moduleigenschaften

aufrufende Module:		**IN UP a ä d l A I W**	
Parameter	IN:	ALPHA	**TEXT**
	OUT:	Reg. Y	"JA"
		Reg. X	**ETEXT**

Modifikation

406-407 | Die Zahl der Anweisungsfolgen FC?_23 PSE entspricht dem Zeitraum, in dem eine Antwort des Benutzers angenommen wird.

3.2.8 K - Kennungstest

Aufgabe

Das Modul **K** erzeugt aus den Daten im ASCII-File **A** einen Datensatznamen (ohne **CODE**).

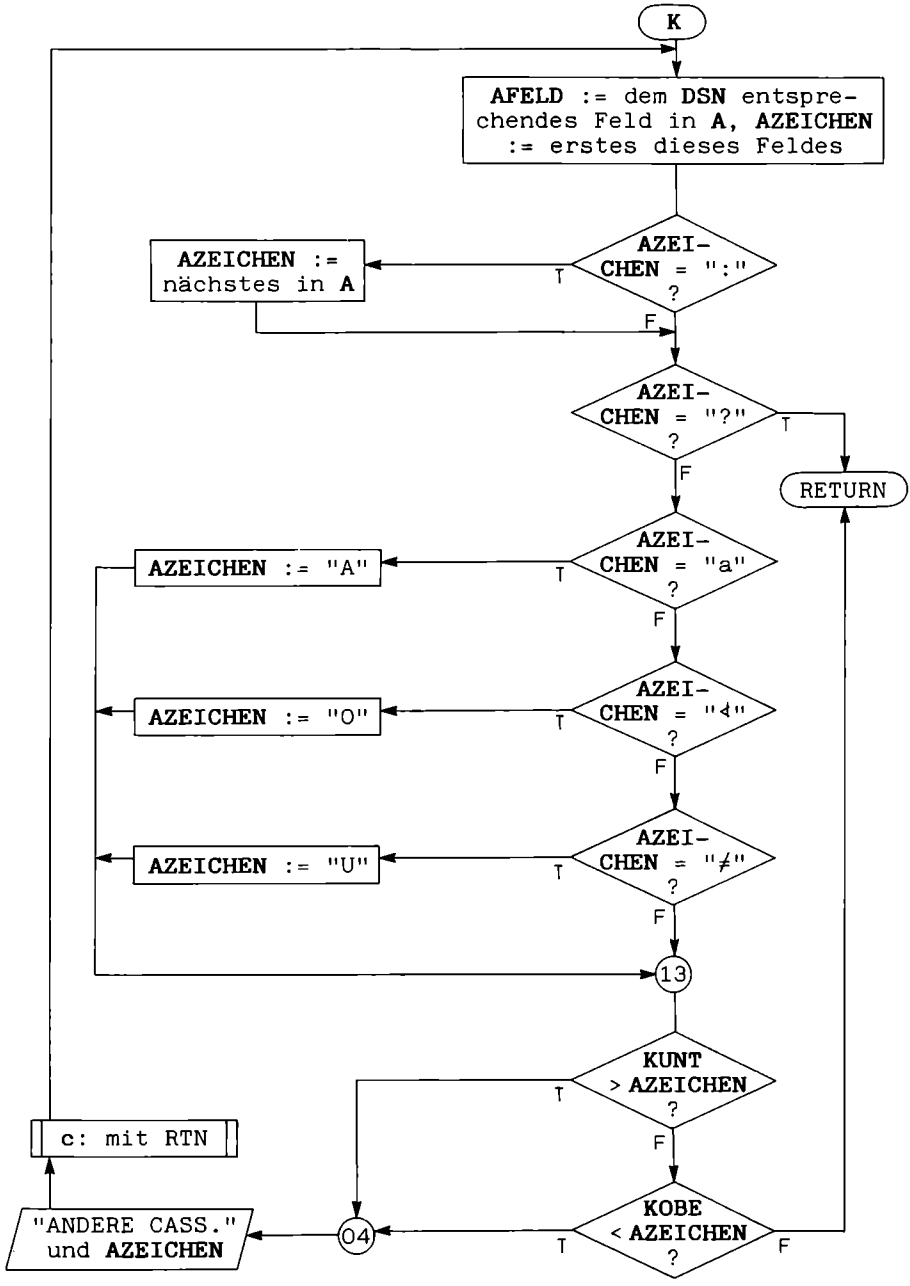


Bild 3-41 Ablaufplan K

```
      118*LBL 04
119 "ANDERE CASS.: "
      120 X<>Y
      121 XTOA
      122 TONE 0
      123 AVIEW
      124 SF 10
      125 XEQ "c"

      126*LBL "K"
      127 "A"
      128 RCL 16
      129 SEEKPTA
      130 GETREC
      131 SEEKPT
      132 RCL 20
      133 ATOX
      134 X≠Y?
      135 GETREC
      136 ASTO X
      137 CLA
      138 ARCL X
      139 "+      "
      140 ASTO 05
      141 RCL 14
      142 ATOX
      143 X=Y?
      144 RTN
      145 RCL 26
      146 X≠Y?
      147 GTO 14
      148 RCL 21
      149 GTO 13

      150*LBL 14
      151 CLX
      152 RCL 09
      153 X≠Y?
      154 GTO 14
      155 79
      156 GTO 13

      157*LBL 14
      158 CLX
      159 RCL 10
      160 X<>Y
      161 X=Y?
      162 85

      163*LBL 13
      164 RCL 03
      165 X>Y?
      166 GTO 04
      167 CLX
      168 RCL 02
      169 X<Y?
      170 GTO 04
      171 RTN
```

Bild 3-42 Listing K

Außerdem überprüft es, ob dieser Datensatzname zur Kennung der eingelegten Datencassette paßt. Ist dies nicht der Fall, wird der Benutzer zum Wechsel der Cassette (Modul **c**) aufgefordert.

Moduleigenschaften

aufrufende Module:	**a n**
aufgerufenes Modul:	**c**
Parameter OUT:	Reg. 05 **DSN**
working file:	**A**

```
      116*LBL 04
117 "ANDERE CASS.: "
      118 X<>Y
      119 XTOA
      120 TONE 0
      121 AVIEW
      122 SF 10
      123 XEQ "c"

      124*LBL "K"
      125 "A"
      126 CLX
      127 SEEKPTA
      128 GETREC
      129 RCL 14
      130 ATOX
      131 X=Y?
      132 GTO 14
      133 RCL 12
      134 X>Y?
      135 GTO 14
      136 X<>Y
      137 RCL 22
      138 X<>Y
      139 X≠Y?
      140 X>Y?
      141 GTO 14
      142 CLA
      143 XTOA
      144 RCL 07
      145 SF 25
      146 SEEKPT
      147 FC?C 25
      148 GTO 14
      149 ARCLREC
      150 ASTO X
      151 CLA
      152 ARCL X
      153 "├    "
      154 ASTO 05
      155 X<>Y
      156 RCL 03
      157 X>Y?
      158 GTO 04
      159 X<>Y
      160 RCL 02
      161 X<Y?
      162 GTO 04
      163 RTN

      164*LBL 14
      165 "?"
      166 ASTO 05
      167 RTN
```

Bild 3-43 Listing K (ORNI-System)

Besonderheiten

Bei der "Berechnung" des Datensatznamens wird ein führender Präfix eines Großbuchstabens (:) nicht berücksichtigt (Zeilen 131 bis 135). Der Datensatzname wird rechts mit Blanks aufgefüllt. Ist das erste Zeichen des Datensatznamens ein Umlaut, wird er - nur für die Überprüfung der Kennung - in den entsprechenden Vokal umgewandelt (Zeilen 145 bis 162).

Modifikation

128	RCL_16 - Position des dem Datensatznamen entsprechenden Feldes
130-139	Dies ist die einzige Stelle im gesamten System, an der Datensatznamen erzeugt werden. Eine Änderung der "Berechnungsmethode" ist somit recht einfach. **Bild 3-43** zeigt das Modul **K** des **ORNI**-Systems (zur Verwaltung von Vogeldaten), bei dem die Datensatznamen aus den Stellen 1 und 8 bis 12 der 13-stelligen Ringnummer gebildet werden (Datenregisterinhalte: 07: 0,007, 12: 48, 14: 63, 22: 58)

3.2.9 P - Plausibilitätskontrolle und programmierte Abkürzungen

Aufgabe

Das Modul **P** überprüft die Plausibilität des im ASCII-File **A** gespeicherten Datensatzes und ersetzt - soweit erforderlich - die programmierten Abkürzungen durch Klartext. Die hierzu notwendigen Aktionen unterscheiden sich je nach Verwendungszweck des Systems.

Als Mindestanforderung ist zu überprüfen, daß das dem Datensatznamen entsprechende Feld nicht leer ist. Beim ADRESS-System wird außerdem getestet, ob die Felder ORT und FELD1 nicht leer sind. Erfüllen die Daten nicht die Plausibilitätsbedingungen, wird der Benutzer mit der Meldung DATEN_FALSCH zur Änderung aufgefordert (Modul **E**).

An programmierten Abkürzungen stehen hier F für :FRAU und H für :HERRN (Felder ANREDE1 und ANREDE2) zur Verfügung. Ein = im Feld NACHNAME2 kopiert den Inhalt des Feldes NACHNAME1.

Moduleigenschaften

aufrufende Module:	**ä n**
aufgerufenes Modul:	**E F**
working file:	**A**

Datenregister: 04 erstes Zeichen von **AFELD**

A muß bereits beim Aufruf von **P** working file sein.

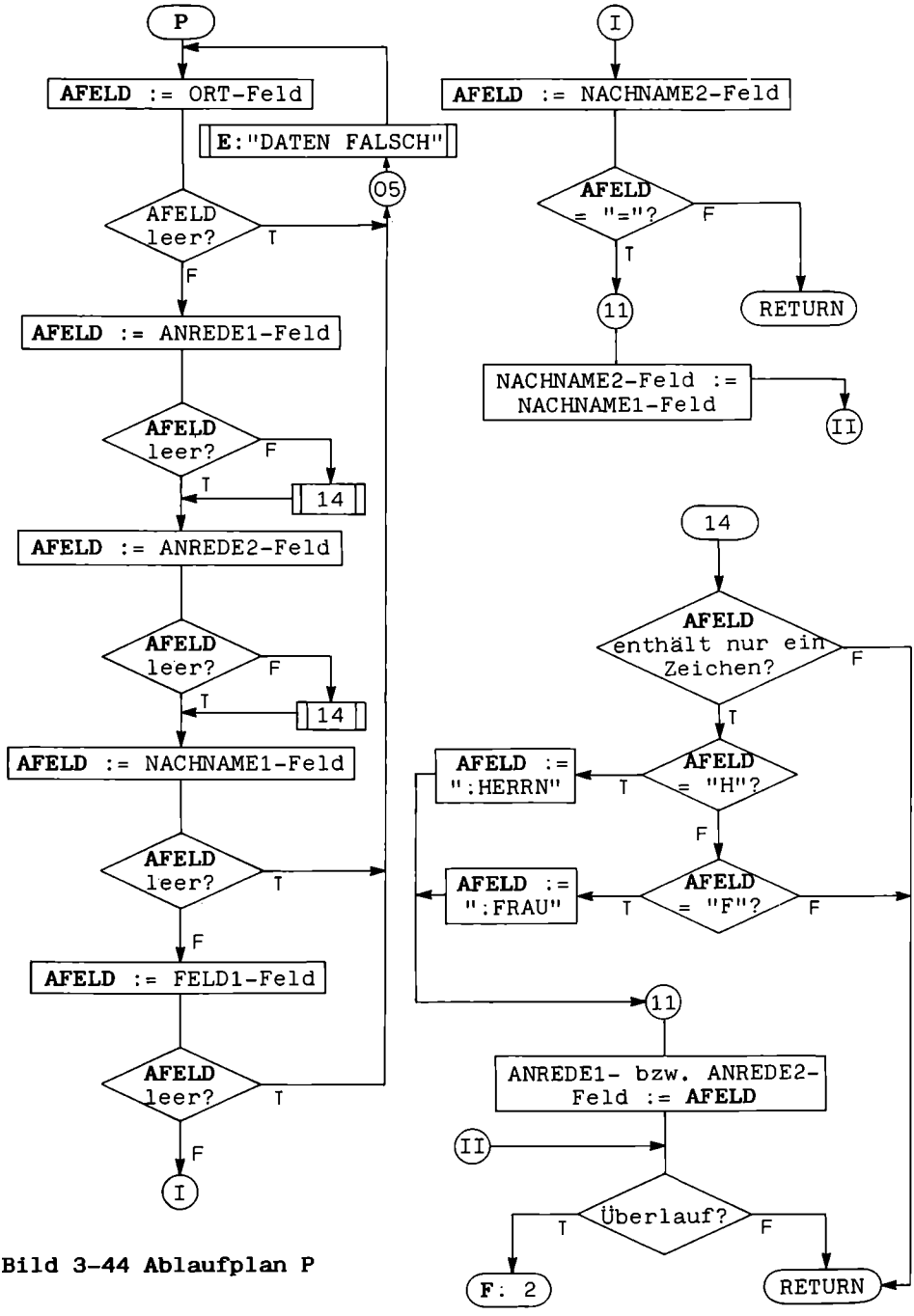


Bild 3-44 Ablaufplan P

```
    172*LBL 05
173 "DATEN FALSCH"
    174 XEQ "E"

    175*LBL "P"
    176 12
    177 XEQ 13
    178 X=Y?
    179 GTO 05
    180 SIGN
    181 XEQ 13
    182 X≠Y?
    183 XEQ 14
    184 5
    185 XEQ 13
    186 X≠Y?
    187 XEQ 14
    188 RCL 16
    189 XEQ 13
    190 X=Y?
    191 GTO 05
    192 RCL 09
    193 XEQ 13
    194 X=Y?
    195 GTO 05
    196 RCL 17
    197 SEEKPT
    198 GETREC
    199 ATOX
    200 61
    201 X≠Y?
    202 RTN
    203 RCL 16
    204 SEEKPT
    205 GETREC
    206 RT
    207 SEEKPT
    208 GTO 11

    209*LBL 14
    210 ALENG
    211 X≠0?
    212 RTN
    213 RCL 23
    214 RCL 04
    215 X=Y?
    216 ":HERRN"
    217 X=Y?
    218 GTO 11
    219 RCL 22
    220 X≠Y?
    221 RTN
    222 ":FRAU"
    223 GTO 11
    224*LBL 13
    225 SEEKPT
    226 GETREC
    227 RCL 15
    228 ATOX
    229 STO 04
    230 X≠Y?
    231 RCL 11
    232 RTN

    330*LBL 11
    331 DELREC
    332 SF 25
    333 INSREC
    334 FS?C 25
    335 RTN
    336 2
    337 GTO "F"
```

Bild 3-45 Listing P

Besonderheiten

Das Modul **P** setzt voraus, daß das ASCII-File **A** das working file ist.

Die Zeilen 330 bis 337 gehören auch zu dem Modul **E**.

Modifikation

176-223, 330-337	Hier können beliebige Bedingungen und Abkürzungen programmiert werden. Der Test des Datensatznamens (Zeilen 187 bis 190) muß jedoch immer ausgeführt werden. **Bild 3-46** zeigt das Modul **P** des recht komfortablen **GALERIE**-Systems (zur Verwaltung von Einladungsadressen). Interessant ist hier die speicherplatzsparende Codierung der Postleitzahlen - die Abkürzungen der Ortsnamen wirken auf zwei Felder (PLZ und ORT).

```
      147♦LBL 05
148 "DATEN FALSCH"
      149 XEQ "E"

      150♦LBL "P"
      151 12
      152 XEQ 13
      153 X=Y?
      154 GTO 05
      155 XEQ 10
      156 SIGN
      157 XEQ 13
      158 X≠Y?
      159 XEQ 11
      160 5
      161 XEQ 13
      162 X≠Y?
      163 XEQ 11
      164 2
      165 XEQ 13
      166 X≠Y?
      167 XEQ 12
      168 6
      169 XEQ 13
      170 X≠Y?
      171 XEQ 12
      172 11
      173 XEQ 13
      174 X=Y?
      175 GTO 05
      176 4
      177 XEQ 13
      178 X=Y?
      179 GTO 05
      180 8
      181 SEEKPT
      182 GETREC
      183 ATOX
      184 61
      185 X≠Y?
      186 GTO 14
      187 4
      188 SEEKPT
      189 GETREC
      190 8
      191 SEEKPT
      192 GTO 09

      193♦LBL 14
      194 9
      195 XEQ 13
      196 X=Y?
      197 RTN
      198 RCL 05
      199 85
      200 X≠Y?
      201 RTN
202 "UND :FRAU :GEMA"
      203 "├HLIN"
      204 GTO 09

      205♦LBL 10
      206 ALENG
      207 X≠0?
      208 RTN
  209 "M(:MEMMINGEN"
      210 XEQ 10
      211 X=Y?
      212 RTN
213 "A(:M:M-:AMENDIN"
      214 "├GEN"
      215 XEQ 10
      216 X=Y?
      217 RTN
218 "E(:M:M-:EISENBU"
      219 "├RG"
      220 XEQ 10
      221 X=Y?
      222 RTN
223 "S(:M:M-:STEINHE"
      224 "├IM"
      225 XEQ 10
      226 X=Y?
      227 RTN
   228 "B):BUXHEIM"
      229 XEQ 10
      230 X=Y?
      231 RTN
 232 "G,:GR∠NENBACH"
      233 XEQ 10
      234 X=Y?
      235 RTN
    236 "├-:ZELLZ"
      237 18
      238 AROT
      239 XEQ 10
      240 X=Y?
      241 RTN
242 "W):WOLFERTSCHWE"
      243 "├NDEN"
      244 XEQ 10
      245 X=Y?
      246 RTN
  247 "O*:OTTOBEUREN"
      248 XEQ 10
      249 X=Y?
      250 RTN
   251 "K(:KEMPTEN"
      252 XEQ 10
      253 X=Y?
      254 RTN
255 "D=:DIETMANNSRIE"
      256 "├D"
      257 GTO 10

      258♦LBL 11
      259 ALENG
      260 X≠0?
      261 RTN
      262 "F:FRAU"
      263 XEQ 11
      264 X=Y?
      265 RTN
   266 "H:HERRN"
      267 XEQ 11
      268 X=Y?
      269 RTN
      270 ASTO 06
      271 "R"
      272 ARCL 06
273 "├ :RECHTSANWALT"
      274 XEQ 11
      275 X=Y?
      276 RTN
      277 "S"
      278 ARCL 06
279 "├ :STEUERBERATE"
      280 "├R"
      281 XEQ 11
      282 X=Y?
      283 RTN
      284 "A"
      285 ARCL 06
  286 "├ :ARCHITEKT"
      287 GTO 11

      288♦LBL 12
      289 ALENG
      290 X≠0?
      291 RTN
      292 "D:DR."
      293 XEQ 11
      294 X=Y?
      295 RTN
  296 "├ MED.M"
      297 9
      298 AROT
      299 XEQ 11
      300 X=Y?
      301 RTN
302 "├ DENT.Z"
      303 15
      304 AROT

      305♦LBL 11
      306 ATOX
      307 RCL 04
      308 X≠Y?
      309 RTN
      310 GTO 14

      311♦LBL 13
      312 SEEKPT
      313 GETREC
      314 RCL 15
      315 ATOX
      316 STO 04
      317 X≠Y?
      318 RCL 11
      319 RTN

      320♦LBL 10
      321 ATOX
      322 RCL 04
      323 X≠Y?
      324 RTN
      325 ATOX
      326 8900
      327 +
      328 STO 06
      329 XEQ 09
      330 11
      331 SEEKPT
      332 CLA
      333 ARCL 06

      334♦LBL 09
      335 DELREC
      336 SF 25
      337 INSREC
      338 FS?C 25
      339 RTN
      340 2
      341 GTO "F"
```

Bild 3-46 Listing P (GALERIE-System)

3.2.10 S - Save

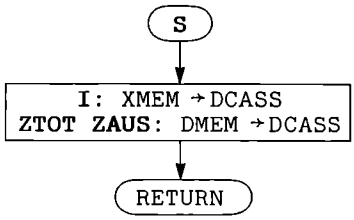


Bild 3-47 Ablaufplan S

```
347♦LBL "S"
348 "I"
349 SAVEAS
 350 ""
351 RCL 06
352 SEEKR
353 WRTRX
354 RTN
```

Bild 3-48 Listing S

Aufgabe

Das Modul **S** speichert das Inhaltsverzeichnis **I** sowie die Zahlen **ZTOT** und **ZAUS** auf der Datencassette.

Moduleigenschaften

aufrufende Module:	c e
working file:	I

Modifikation

350	"a" - Name des Datenregisterfiles

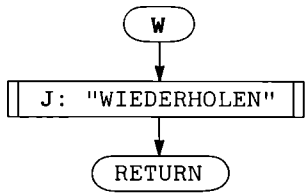


Bild 3-49 Ablaufplan W

```
398♦LBL "W"
399 "WIEDERHOLEN"
400♦LBL "J"
```

Bild 3-50 Listing W

3.2.11 W - Wiederholen-Prompt

Aufgabe

Der Aufruf des Modules **W** entspricht exakt der Wirkung des Modules **J** mit dem Textparameter WIEDERHOLEN .

Moduleigenschaften

aufrufende Module:	**IN E**	
aufgerufenes Modul:	**J**	
Parameter OUT:	Reg. Y	"JA"
	Reg. X	**ETEXT**

3.3 Systemkonstanten - ADRRE

Bild 3-51 zeigt die Belegung der Datenregister mit den Konstanten des ADRESS-Systems. Die Inhalte der Register 13 und 28 bis 31 sollten nicht verändert werden. Die anderen Register können im Rahmen der Programmoptimierung (vgl. 4.2) neu belegt werden.

Das Register 31 enthält den Initialisierungsstatus der Flags 00 bis 43 im RCLFLAG/STOFLAG-Format. Diese Textkonstante kann auf zwei Arten erzeugt werden:

- normal: den Rechnerstatus laut **Bild 3-52** setzen (etwa nach master clear: SF_08 CF_21 CF_28 CF_29 FIX_0), RCLFLAG STO_31

- synthetisch: CLA 31 XTOA 240 XTOA 8 XTOA 0 XTOA 2 XTOA 0 XTOA 8 XTOA RCL_M STO_31

Das Programm **ADRRE** (Barcode s. 6.2) speichert die Konstanten des ADRESS-Systems automatisch in den entsprechenden Datenregister.

```
R06=  0.001
R07=  1.002
R08=  1.004
R09=  13.000
R10=  29.000
R11=  32.000
R12=  45.255
R13= "A,"
R14=  63.000
R15=  94.000
R16=  4.000
R17=  8.000
R18=  36.000
R19=  38.000
R20=  58.000
R21=  65.000
R22=  70.000
R23=  72.000
R24=  73.000
R25=  78.000
R26=  97.000
R27=  100.000
R28= ",A"
R29= "Y,"
R30= ",Y"
R31= "pÅ"
```

Bild 3-51 Systemkonstanten

```
STATUS:
SIZE= 032
Σ= 11
DEG
FIX 0

FLAGS:
F 00  CLEAR
:     :
F 07  CLEAR
F 08  SET
F 09  CLEAR
:     :
F 25  CLEAR
F 26  SET
F 27  CLEAR
:     :
F 39  CLEAR
F 40  SET
F 41  CLEAR
:     :
F 54  CLEAR
F 55  SET
```

Bild 3-52 Rechnerstatus

4 Programmoptimierung

Besonders bei relativ großen Software-Systemen lohnen sich Überlegungen zur Programmoptimierung. Neben den allgemeineren Zielsetzungen wie Bedienungsfreundlichkeit und Sicherheit - sie werden bereits beim Programmentwurf berücksichtigt - sind zwei Optimierungsstrategien wichtig:

- **Minimierung des benötigten RAM-Bereiches:** Reduzierung der Programmlängen (in Bytes) und Verminderung der Anzahl der benützten Datenregister;

- **Verkürzung der Ausführungszeiten:** Verwendung schneller Anweisungen bzw. Anweisungsfolgen.

Diese beiden Zielsetzungen können sich gegenseitig verstärken (z. B. werden kurze Programme unter Umständen schneller geladen), wirken aber oft auch entgegengesetzt, wie das Beispiel der Zahl 1 zeigt (**Bild 4-1**): die um den Faktor 3 schnellere Anweisungsfolge CLX SIGN benötigt doppelt soviel Speicherplatz wie die konventionelle 1.

Anweisungen	**Speicherplatz** Bytes	**Ausf.-zeit** ms rel.	**Voraussetzung**	**Stacklift**	**LAST_X-Inhalt**
1	1	64,3	-	ja	unv.
E	1	58,2	-	ja	unv.
SIGN	1	13,0	X>0, Zahl	nein	X
ST/_X	2	41,8	X≠0, Zahl	nein	X
CLX_SIGN	2	22,7	-	nein	0

Bild 4-1 Verschiedene Möglichkeiten, die Zahl 1 zu laden

Offensichtlich ist die Häufigkeit des Auftretens einer zu optimierenden Anweisungsfolge ein wesentliches Kriterium für die optimale Entscheidung. Da aber Programmteile wiederholt durchlaufen werden können, ist die Häufigkeit je nach verfolgter Strategie in verschiedenen Ausprägungen zu betrachten:

- Für die Minimierung des **Speicherplatzes** ist die Zahl der Anweisungssequenzen **im Programmtext**, die **statische Häufigkeit**, relevant.

- Bei der Optimierung nach der **Ausführungszeit** ist die Anzahl der bei einem **Programmlauf** abgearbeiteten Anweisungen, die **dynamische Häufigkeit**, maßgebend.

Während die statische Häufigkeit leicht durch Zählen im Programmtext ermittelt werden kann, lassen sich für die dynamische Häufigkeit oft nur empirisch ermittelte Werte oder Größenordnungen angeben.

Beide hier vorgestellten Strategien können - unvernünftig angewandt - zu extrem unlesbaren Programmen führen, deren Wartung nahezu unmöglich wird. Die Grenzen der Programmoptimierung sind also nicht erst mit dem schnellsten und kürzesten Programm erreicht.

4.1 Optimierung der lokalen Sprünge

4.1.1 Theorie

Bei der Programmierung von Sprüngen unterscheidet man vier Arten der Adressierung:

- **absolut**: die Adresse des Sprungzieles wird codiert (z. B. Gto "Zeilennummer" bei programmierbaren Taschenrechnern von Texas Instruments);

- **relativ:** der Abstand zwischen Sprunganweisung und -ziel sowie die Orientierung des Sprunges werden angegeben (z. B. verzweigt beim HP-67/97 GTO_(i) mit -n im Register i um n Zeilen rückwärts - dies ist allerdings eine indirekte (s. u.) relative Adressierung);

- **symbolisch:** das Sprungziel wird mit einem Namen versehen, der Argument der Sprunganweisung ist (Label);

- **indirekt:** das Argument der Sprunganweisung steht in einem Register (z. B. GTO_IND_00 beim HP-41).

Da die symbolische Adressierung zwar bequem zu programmieren (die Sprungadressen bzw. -distanzen müssen nicht ausgerechnet werden), aber auch sehr ineffizient ist (das passende Label muß erst gesucht werden), verwendet das Betriebssystem des HP-41 neben der indirekten Adressierung, die im folgenden unberücksichtigt bleibt, eine Kombination aus symbolischer und relativer Adressierung /1/. Bei der ersten Ausführung eines GTOs oder XEQs wird das entsprechende Label gesucht und der Abstand zum Sprungziel sowie die Orientierung des Sprunges in die Sprunganweisung geschrieben. Das Label wird erst im Bereich zwischen Sprunganweisung und END, dann vom Programmanfang bis zur Sprunganweisung gesucht.

Das Betriebssystem übersetzt (compiliert) also symbolische in relative Sprünge, die direkt (ohne Labelsuche) ausgeführt werden können. Da die Sprungdistanzen und -orientierungen im compilierten Programm gespeichert sind, werden sie bei Transfers nach bzw. von Magnetkarte, IL-Massenspeicher, Extended Memory oder Barcode mit übertragen. Nach Änderungen im Programm werden sicherheitshalber alle Sprungdistanzen auf 0 gesetzt, um das Betriebssystem die Sprunganweisungen neu compilieren zu lassen. Hat ein Programm kein eigenes END (nur das permanente .END. des Programmspeichers), zerstört auch PACKING die Sprungdistanzen.

Mit Mitteln der synthetischen Programmierung oder durch Ausnützen eines Fehlers in älteren Betriebssystemen (Bug 8: wird nach Änderungen der PRGM-Modus durch einen Druck auf die ON-Taste verlassen, werden die Sprungdistanzen nicht gelöscht) lassen sich auch relative Sprünge ohne Labels programmieren /2/. Der hohe Aufwand bei der Programmierung und Wartung wird jedoch nur in Extremfällen durch Reduzierung der Programmlänge (ein oder zwei Bytes) und Geschwindigkeitserhöhung (ca. 40 % der Ausführungszeit einer Sprunganweisung mit Label) gerechtfertigt.

Normalerweise speichert der Rechner GTO-Anweisungen mit den Argumenten 00 bis 14 sowie die entsprechenden Labels im Kurzform-Format. Sie belegen dann zwei bzw. ein Byte des Programmspeichers. Ein Kurzform-GTO-Befehl kann jedoch nur Sprungdistanzen bis zu ±112 Bytes speichern.

Größere Sprungweiten können nur in Langform-GTOs (mit den Argumenten 15 bis 99, A bis J und a bis e) compiliert werden, die drei Bytes belegen. Die entsprechenden Labels benötigen zwei Bytes. Mit Hilfe der synthetischen Programmierung können weitere Langform-Anweisungen mit den Argumenten 100, 101 sowie T, Z, Y, X, L, M, N, O, P, Q, und |- (das sind - neben a bis e - die Bezeichnungen der Statusregister) erzeugt werden.

XEQ-Anweisungen belegen immer drei Bytes und können beliebig lange Sprungdistanzen speichern.

Während des Programmeditierens unterscheidet das Betriebssystem Kurzform- und Langform-Anweisungen anhand der Labelargumente: 00 bis 14 einerseits und 15 bis 101, A bis e andererseits. Dieser starre Zusammenhang zwischen Sprungbefehlen und Labels läßt sich mit Mitteln der synthetischen Programmierung aufheben. Sind z. B. 10 kurze und ein langer Sprung auf ein und das selbe Ziel zu programmieren, spart man mit einem Langform- und 10 Kurzform-GTOs sowie einem Kurzform-LBL gegenüber der normalen Programmierung 11 Bytes.

Im folgenden wird ein Verfahren beschrieben, das unter Berücksichtigung der statischen Häufigkeit die Kurzform-Argumente auf die zu programmierenden Sprünge optimal verteilt, also den belegten Programmspeicher reduziert. Die dynamische Häufigkeit ist nur bei sehr kurzen Schleifen zu beachten (Kurzform-GTO-Anweisungen sind ca. 1,4-mal schneller als die entsprechenden Langform-Befehle).

4.1.2 Verfahren zur Optimierung der Labelargumente

Die Voraussetzung für das folgende Schema ist ein gepacktes (GTO_..) Programm mit beliebig verteilten Labeln. Labelargumente, die nicht verändert werden dürfen (z. B. bei indirekten Sprüngen), bleiben im folgenden unberücksichtigt.

Programm	Markierung	optimiert
100 GTO 20	20L	GTO 14 (3 B.)
⋮		
200 LBL 02	↓K	LBL 00
⋮		
205 GTO 02	02K	GTO 00
⋮		
210 GTO 20	20K	GTO 14
⋮		
215 LBL 20	↑L ↑K	LBL 14
⋮		
220 GTO 03	03K	GTO 14
⋮		
225 LBL 03	↑K	LBL 14

K: Kurzform-Sprung
L: Langform-Sprung

Bild 4-2 Optimierung lokaler Sprünge

Markieren und Zählen

Auf einem Programmlisting (am günstigsten im NORM-Modus des Druckers erstellt) werden alle GTO- und XEQ-Anweisungen sowie die zugehörigen Labels markiert (vgl. **Bild 4-2**). Anschließend ist die Zahl derjenigen auf jedes Label zielenden GTO-Anweisungen zu ermitteln, die sich in Kurzform codieren lassen (statische Häufigkeit). Dies sind die Sprünge über eine Distanz von weniger als 113 Bytes. In Grenzfällen läßt sich die Sprungart wie folgt ermitteln:

- falls notwendig, GTO-Befehl und Label durch die entsprechenden Kurzform-Anweisungen ersetzen
- Programm packen: PACK
- Sprunganweisung ausführen: SST
- zurück zur Sprunganweisung und dieselbe noch einmal ausführen: GTO_.XYZ, SST-Taste kurze Zeit gedrückt halten
- Falls der zweite Sprung schneller ausgeführt wird als der erste (GTO_XY ist nur kurze Zeit zu sehen), war die Compilierung des Sprunges erfolgreich, die Kurzformanweisung also ausreichend.

Verteilen der Kurzform-Argumente

Die Kurzform-Labels 00 bis 14 werden nun auf die Sprungziele mit den meisten kurzen Sprüngen verteilt. Sollten Label-Anweisungen unbenannt bleiben, bekommen sie Langform-Argumente (15 bis 99, synthetisch: 100, 101, A bis e). Selbstverständlich können Labelargumente in einem Programm mehrmals verwendet werden. **Bild 4-3** zeigt die korrekten Positionen paralleler

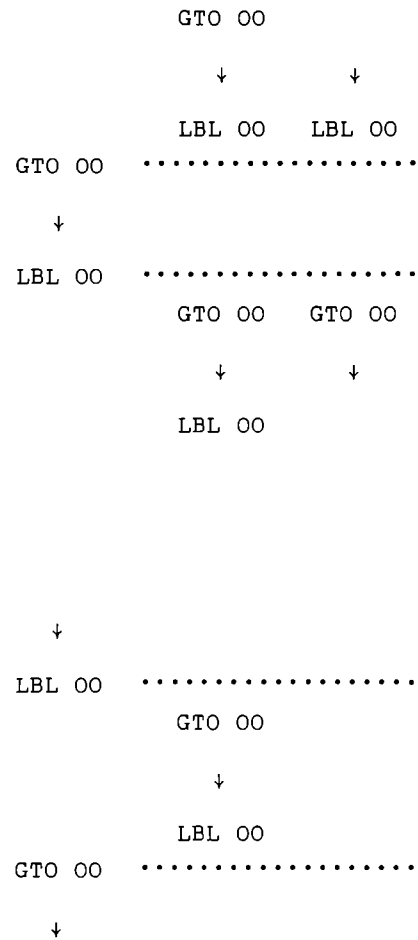


Bild 4-3 korrekte Positionen gleichnamiger Sprünge

Sprung- und Label-Anweisungen mit übereinstimmenden Argumenten. Alle anderen Kombinationen führen zu logischen Fehlern.

Mit zwei Ausnahmen wurden bei der hier vorgestellten Software die Labelargumente für Rückwärtssprünge mit 00 beginnend in aufsteigender Folge, die für Vorwärtssprünge mit 14 beginnend abwärts zählend verteilt. Diese Maßnahme erhöht die Lesbarkeit der Programme.

Ändern des Programmes

Die Sprung- und Label-Anweisungen sind entsprechend der neuen Verteilung der Labelargumente abzuändern. Für lange Sprünge auf Kurzform-Labels werden synthetische 3-Byte-GTO-Befehle verwendet (vgl. 5.6). Die Änderung einer Label- oder Sprunganweisung kann die Verlängerung oder Verkürzung der Distanzen anderer Sprünge bewirken. Diese Seiteneffekte können eine Neuverteilung der Labelargumente erforderlich machen.

4.1.3 Manipulation des Programmzeigers

```
09 ""
10 2
11 RCL b
12 "+"
13 READSUB
14 SAVEP
15 PCLPS
16 DSE Y
17 STO b
```

Mit RCL_b bzw. STO_b wird das Statusregister b, das neben zweieinhalb Rücksprungadressen den Programmzeiger enthält, direkt adressiert. Somit lassen sich absolute Sprünge programmieren, die sehr schnell ausgeführt werden. Allerdings lohnt sich der recht hohe Aufwand der programmgesteuerten Berechnung eines korrekten Register-b-Inhalts meist nicht.

Bild 4-4 RCL_b/STO_b

Nur bei Schleifen kann die Adresse des Sprungzieles einfach mit RCL_b ermittelt werden. Der Wert darf allerdings nicht in einem Datenregister zwischengespeichert werden, da er dann durch Normalisierung verändert werden kann. Meist wird der Inhalt des Registers b im Stack gespeichert. **Bild 4-4** erläutert diese Technik anhand eines Ausschnitts des Modules **IN.**

4.2 Optimierung der Konstanten

4.2.1 Theorie

Konstanten können direkt im Programmtext angegeben oder in Datenregistern gespeichert und aufgerufen werden. Die erste Möglichkeit ist relativ speicher- und zeitintensiv: pro **digit-entry-Operation** (das sind die einzelnen Ziffern, aber auch Vorzeichen, Dezimal-Komma bzw. -Punkt sowie Exponent-E) werden ein Byte Programmspeicher sowie zwischen 40 und 80 ms Rechenzeit benötigt. RCL-Anweisungen belegen ein oder zwei Bytes und werden etwa doppelt so schnell ausgeführt wie eine Ziffer.

Da ein belegtes Datenregister den für Programme freien Bereich um sieben Bytes reduziert, ist für jede Konstante zu entscheiden, ob sie im Programm oder in einem Register stehen soll.

Benützt ein System das Cassettenlaufwerk, können die Konstanten in einem DA-File gespeichert und beim Programmstart in die Datenregister des Rechners geladen werden. So entfallen eine Reihe von STO-Anweisungen. Da ein Record eines IL-Massenspeichers genau 32 Datenregister aufnehmen kann (vgl. 2.3.2), werden Bereiche von 32, 64, 96, ... Registern besonders effizient gespeichert und geladen.

Wie bei den Sprüngen und Labels gibt es auch bei der Registeradressierung Kurzform- und Langform-Anweisungen. Die Datenregister 00 bis 15 werden mit 1-Byte-Befehlen direkt adressiert. Zugriffe auf die anderen Register benötigen zwei Bytes und sind geringfügig langsamer. X<>__ und indirekte Anweisungen belegen immer zwei Bytes. Natürlich lassen sich auch Langform-RCLs und -STOs auf die Register 00 bis 15 synthetisieren. Im Gegensatz zu den Langform-GTOs auf Kurzform-Labels lassen sie sich jedoch nicht sinnvoll einsetzen.

Bei der Entscheidung, welche Konstanten in welchen Registern gespeichert werden sollen, hilft das folgende Schema, das die statische und die dynamische Häufigkeit berücksichtigt.

4.2.2 Schema zur Optimierung der numerischen Konstanten

Das Verfahren wird im folgenden anhand des ADRESS-Systems (ohne die Hilfsprogramme **ADRUP**, **ADRIN**, **ADRBE**) erläutert.

Ermittlung der Konstanten (**Bild 4-5**, Spalten 1 und 2)

Hierbei bleiben Konstanten, die ein Byte belegen und nur sehr selten benützt werden (z. B. die Fehlernummern 1, 2 und 3 als Parameter des Moduls **F**), unberücksichtigt. Die Werte 0 und 1 können meist durch die schnellen Anweisungen CLX bzw. SIGN oder CLX SIGN ersetzt werden.

In Spalte 2 werden die Häufigkeiten der Konstanten festgehalten. Die dynamische Häufigkeit läßt sich hier nur in Größenordnungen angeben. Es gelten folgende Abkürzungen:

M	einmal pro Aufruf des Modules;
S	einmal pro Verarbeitung eines Datensatzes;
F	einmal pro Verarbeitung eines Datenfeldes;
Z	einmal pro Verarbeitung eines Zeichens.

Wert	Auftreten	dyn. Häufigkeit Z	F	S	M	Rang	stat. Häufigkeit Zahl	Ersp.	Rang	Register
E-3	Sa Za Zä MS	6		3	3	1	4	8	1	06
3 E-3	Mc				4					
1,002	SE			5		15	1	4	4	07
1,004	SE			5		16	1	4	5	08
1,255	Sn			5		17	1	4	6	09
4	FE SK SP SP		1	3		14	4	0		16
5	Md SP			1	1					
6	Si Sl			2						
8	Sn Sn SP			3						
12	Ma Si SP			4	2	18	3	3		17
13	Sä Zd SK SP	2		6		4	4	4	7	10
17	MD				2					
21	Md				2					
23	Mc Md				4					
25	Md				2					
29	Sä Zd SK	2		4		5	3	3	9	11
32	Fd Fd Fd SK 5SP		6	12		10	5	5	3	12
35	Sa Si			4		19	2	2		18
36	Zd	2				9	1	1		19
38	Sa Ma			2	2					
44	Si Sl			4		20	2	2		20
58	Zd SK	2		2		6	2	2		21
61	SP			2						
63	Ma Ma Za Mä Zä SK	4		2	6	2	6	6	2	14
65	Zd SK	2		2		7	2	2		22
70	2SP			4		21	1	1		23
72	2SP			4						
73	Fd		2			12	1	1		24
78	Md Fd		2		2	11	2	2		25
79	SK			2						
83	Md				2					
85	SK			2						
94	Fd Zd Sn 5SP	2	2	12		3	4	4	8	15
97	Zd Sk	2		2		8	2	2		26
E2	Fd		2			13	1	1		27

Bild 4-5 Optimierung der numerischen Konstanten

Ein vorangestellter Faktor erhöht die dynamische Häufigkeit. Die angehängte Modulbezeichnung erleichtert die Neuverteilung der Konstanten nach Programmänderungen.

Bestimmung der freien Datenregister

Beim ADRESS-System werden die Register 06 bis 12 und 14 bis 27 für die Speicherung numerischer Konstanten verwendet. In den Registern 13 und 29 stehen ALPHA-Konstanten, die im Modul ä über die Konstanten 13 und 29 indirekt adressiert werden.

Optimierung nach der dynamischen Häufigkeit (**Bild 4-5**, Spalte 3)

Für die Entscheidung, ob eine Konstante überhaupt in einem Datenregister zu speichern ist (hier stehen für 35 Konstanten nur 21 Register zur Verfügung), ist die dynamische Häufigkeit maßgebend.

Da zur Ausführungszeit einer im Programmtext stehenden Konstanten sämtliche digit-entry-Anweisungen zählen, ist die Häufigkeit mit der Anzahl dieser Operationen zu multiplizieren. Die Auswertung erfolgt getrennt nach Größenordnungen.

Die dynamischen Häufigkeiten in Verbindung mit der Priorität der Größenordnungen (hier Z, F, S, M) ergeben eine Rangfolge. Die in diesem Sinne häufigsten Konstanten werden in den freien Datenregistern gespeichert, die restlichen in den Programmtext übernommen.

Optimierung nach der statischen Häufigkeit (**Bild 4-5**, Spalte 4)

Bei der Verteilung der Konstanten auf die in Kurzform adressierbaren Datenregister spielt neben der statischen Häufigkeit auch die Anzahl der durch die Speicherung in einem Kurzform-Register gesparten Bytes eine Rolle. Diese ist die um eins verminderte Zahl der digit-entry-Anweisungen. Sie wird mit den Werten der statischen Häufigkeit multipliziert.

Die Konstanten, deren Speicherung in Kurzform-Registern die größten Einsparungen bewirken, werden auf die Kurzform-Register verteilt. Sind zwei Konstanten in diesem Sinne gleichwertig, ist ihre Rangfolge nach der dynamischen Häufigkeit entscheidend.

Bild 4-5, Spalte 5 zeigt die endgültige Verteilung der Konstanten des ADRESS-Systems.

5 Synthetische Programmierung

Dieser Abschnitt soll keine Einführung in die synthetische Programmierung sein, sondern schrittweise zu den synthetischen Anweisungen des ADRESS-Systems führen. Je nach Wissensstand und Ausstattung kann der Leser auf verschiedenen Ebenen einsteigen:

Teil	Voraussetzungen	Ergebnis
5.1		(Grundlagen)
5.2		Byte Grabber (**BG**)
5.3	**BG**	Load Bytes-Programm (**LB**)
5.4	**LB**	Key Assignment-Programm (**KA**), **BG**
5.5	**KA**	Q-Loader (**TQ**, **LQ**, **GQ**, **XQ**)
5.6	**BG**, **TQ**, **LQ**, **GQ**, **XQ**, **LB**	beliebige Anweisungen

Sollte sich der Rechner während synthetischer Experimente seltsam verhalten (Crash, keine Reaktion auf Tastendruck) hilft meist die master clear-Prozedur. In seltenen Fällen ist - bei nicht angeschlossenem HP-82143A - kurz das Batterie- oder Akku-Paket herauszunehmen.

5.1 Grundlagen

Der Rechner verschlüsselt die programmierten Anweisungen in Bytes. Bei einfachen programmierbaren Taschenrechnern (etwa HP Serie 10) reichen die 256 Werte eines Bytes aus, um alle Anweisungen zu codieren. Die komplexeren Befehle des HP-41 werden in mehreren Bytes gespeichert. So kann ein Byte je nach

Kontext verschiedene Bedeutungen haben. Alle Informationen hierzu finden sich in der HP-41 QUICK REFERENCE CARD FOR SYNTHETIC PROGRAMMING (kurz: Hexcode-Tabelle, s. 6.5). Der Wert des ersten Bytes einer Anweisung entscheidet über ihren Typ:

- **1-Byte-Funktionen** stehen in den Zeilen 0 bis 8 (außer 1D bis 1F) der Hextabelle. Beispiele: LBL_12 (0D), ATAN (5E), ADV (8F)

- **normale 2-Byte-Anweisungen** bestehen aus Prefix (erstes Byte, Werte 90 bis 9F bzw. A8 bis AD sowie CE und CF) und einem Suffix (zweites Byte, Werte 00 bis 7F für direkte und 80 bis FF für indirekte Adressierung des Operanden). Beispiele: RCL_16 (90 10), RCL_IND_16 (90 90), X<>_IND_L (CE F4), ARCL_M (9B_75)

- Das Prefixbyte der **GTO_IND-** und **XEQ_IND-Anweisungen** hat den Wert AE. Suffixe mit Werten 00 bis 7F codieren GTO_IND-, solche mit Werten 80 bis FF XEQ_IND-Befehle. Beispiele: GTO_IND_00 (AE 00), XEQ_IND_e (AE FF)

- **Peripherie-Funktionen** werden als XROM a,b angezeigt (vgl. Handbuch des entsprechenden Geräts bzw. Moduls). Das Prefix-Byte nimmt den Wert 160 + abgerundet(a / 4) (dezimal) an und reicht von A0 bis A7. Das Suffix-Byte wird mit 64 * (a MOD 4) + b codiert. Beispiele: EMDIR (XROM 25,14, A6 4E), -EXT_FCN_1B (XROM 25,00, A6 40)

- In **Kurzform-GTO-Anweisungen** (Prefix B1 bis BF) stehen auch die Distanzen zu den entsprechenden Labels (vgl. 4.1.1). Beispiel (uncompiliert): GTO_01 (B2 00)

- Bei **3-Byte-GTO-** und **XEQ-Anweisungen** (Prefix D0 bis DF bzw. E0 bis EF) sind die Sprungdistanzen im mittleren Byte codiert (vgl. 4.1.1). Das dritte Byte enthält neben der Sprungorientierung das Label-Argument (00 bis 7F). Beispiele (uncompiliert): GTO_15 (D0 00 0F), XEQ_00 (E0 00 00)

- Das erste Byte der **Textanweisungen** hat Werte zwischen F0 und FF und gibt die Zahl (0 bis 15) der folgenden Zeichen an. Ein einzelnes F0-Byte dient als schnelle NOP-Anweisung (no operation) nach DSE- und ISG-Befehlen. Beispiele: "" (F0, NOP), "ADRBE" (F5 41 44 52 42 45), "├A" (F2 7F 41)

- Das Prefix-Byte der **globalen Label-Anweisungen** nimmt die Werte C0 bis CD an. Die niederwertige Hälfte dieses Bytes sowie das zweite Byte enthalten die Distanz zum nächsten globalen Label oder END (vom permanenten .END. aus in Richtung fallender Zeilennummern). Zur Synthetisierung von Labels genügen jedoch die Werte C0 und 00 für das erste bzw. zweite Byte. Beim PACKING werden die Distanzen "compiliert". Das dritte Byte ist ein Text-Prefix, das vierte Byte - es steht innerhalb der Text-Anweisung, die den Labelnamen enthält - codiert die Tastenzuordnung. Beispiel (ohne Key Assignment): LBL_"ADRBE" (C0 00 F6 00 41 44 52 42 45)

- Bei globalen **GTO**- und **XEQ-Anweisungen** (Prefixe 1D bzw. 1E) verschlüsselt die folgende Textanweisung den Labelnamen des Sprungzieles. Beispiel: GTO_"ADRBE" (1D F5 41 44 52 42 45)

- Die **END-Anweisung** wird wie ein globales Label verschlüsselt - das dritte Byte ist jedoch kein Text-Prefix. Beispiel: END (nicht-permanentes END eines gepackten und compilierten Programmes, CX YZ 0D)

5.2 BG - Byte Grabber

Das wichtigste Werkzeug der synthetischen Programmierung ist der Byte Grabber. Der Befehl gehört zur Gruppe der Byte Jumper, die alle nur als Tastenzuordnungen wirksam werden. Ein umständlicher zu bedienender Byte Jumper begründete die synthetische Programmierung **/3//4/**. Der Byte Grabber fand als (deutscher) Wolf-Befehl **/5/** weniger - zu wenig - Beachtung als unter der Bezeichnung Prefix Masker **/6/**.

Natürlich kann der Byte Grabber nur synthetisch einer Taste zugeordnet werden:

- master clear (MEMORY_LOST)
- ASN_"GTO"_LN ASN_"DEL"_$\sqrt{x}$
- PRGM
- CAT_1 sofort mit R/S anhalten
- ALPHA ← ALPHA, falls jetzt nicht 4094 RCL_01 angezeigt wird, wurde der CAT_1 zu spät gestoppt
- GTO_.005 (im USER-Modus LN . LN)
- DEL_003 (im USER-Modus $\sqrt{x}$ $\sqrt{x}$)
- "7AAAAAA", HP-41C und HP-41CV zeigen ohne X FUNCTIONS Modul nur "7$^{-------}$" an
- PRGM CAT_1 ASN_""_$\sqrt{x}$

Der Byte Grabber ist nun der LN-Taste zugeordnet. Der Befehl wird als XROM 28,55 angezeigt und ist im RUN-Modus wirkungslos.

Die verschiedenen Bezeichnungen des Byte Grabbers lassen seine Wirkungsweise ahnen. Der Befehl bindet das erste Byte einer Multi-Byte-Funktion (aber auch das Byte einer 1-Byte-Funktion), sodaß das folgende Byte selbst die Rolle eines Prefixes übernehmen und die nachfolgenden Bytes "grabben" kann. Damit der Byte Grabber nicht ein unsichtbares Nullbyte (00) bindet, ist vor seiner Anwendung meist PACK auszuführen, das alle freien Nullbytes beseitigt. Das folgende Beispiel veranschaulicht die Verwendung des Byte Grabbers.

Aktion/Programmtext	**Hexcodes**
Programm eingeben:	
01 LBL_"BG"	CE 00 F3 00 42 47
02 "BYTE GRABBER"	FC 42 59 54 45 20 47 52 41 42 42 45 52

Text-Prefix maskieren: GTO_.001 **BG**

```
01 LBL_"BG"          CE 00 F3 00 42 47
02 "¯7----X"         F7 00 37 00 00 00 00 FC
03 *                 42
04 SIN               59
05 CHS               54
06 X>Y?              45
07 RCL_00            20
08 Σ+                47
09 SQRT              52
10 -                 41
11 *                 42
12 *                 42
13 X>Y?              45
14 SQRT              52
```

Bytes ersetzen: GTO_.004 DEL_003 X≠Y? R↑ LN1+X GTO_.009 DEL_006 PI ABS FACT FACT LN1+X PI

```
01 LBL_"BG"          CE 00 F3 00 42 47
02 "¯7----X"         F7 00 37 00 00 00 00 FC
03 *                 42
04 X≠Y?              79
05 R↑                74
06 LN1+X             65
07 RCL_00            20
08 Σ+                47
09 PI                72
10 ABS               61
11 FACT              62
12 FACT              62
13 LN1+X             65
14 PI                72
```

Prefix de-masken und aufräumen: GTO_.001 **BG** DEL_002

```
01 LBL_"BG"          CE 00 F3 00 42 47
                     00 00 00 00 00 00 00 00 00 00 00 00 00 00
02 "BXXe GXabbeX"    FC 42 79 74 65 20 47 72 61 62 62 65 72
```

5.3 LB - Load Bytes Programm

Die oben beschriebene Methode der direkten Synthese ist recht umständlich und funktioniert nicht in allen Fällen. Mit dem in **Bild 5-1** gelisteten Programm **LB** (Autor: Gerhard Kruse, Barcode s. 6.2) können beliebige Bytewerte in den Programmspeicher geladen werden. Es enthält einige synthetische Anweisungen, die mit dem Byte Grabber erzeugt werden können:

Befehl	**Zeilennr.**	**Hex**	**Synthese**
RCL_b	003	90 7C	RCL_IND_16 MEAN BST BST **BG** ←
X<>M	004 041 043	CE 75	RCL_IND_78 RDN BST BST **BG** ←
X<>c	040 044	CE 7D	RCL_IND_78 SDEV BST BST **BG** ←
TONE_9	029	9F 59	RCL_IND_31 SIN BST BST **BG** ←

```
01♦LBL "LB"
02 CLA
03 RCL b
04 X<> [
05 ALENG
06 7
07 ATOX
08 GTO IND Z

09♦LBL 04
10 2
11 MOD
12 256
13 *
14 ALENG
15 2
16 -
17 ATOX
18 *
19 X<0?
20 CLX
21 +

22♦LBL 03
23 2
24 -
25 FRC
26 SF 22
27♦LBL 00
28 FS? 22
29 TONE 9
30 FS?C 22
31 STOP
32 FC? 22
33 .
34 XTOA
35 X<>Y
36 RDN
37 DSE Y
38 GTO 00
39 7
40 X<> c
41 X<> [
42 STO IND L
43 X<> [
44 X<> c
45 X<>Y
46 CLA
47 DSE L
48 FS? 22
49 GTO 00
50 BEEP
51 .END.
```

Bild 5-1 Listing LB

Bedienung **LB**

1. im RUN-Modus RTN
2. GTO zu der Zeile, ab der die Bytes geladen werden sollen
3. im PRGM-Modus XEQ_"LB" STOP und mindestens 7 * abgerundet (Byteanzahl / 7) + 26 Pufferbytes (z. B. ENTER↑) eingeben
4. GTO zurück zu der Programmzeile mit XEQ_"LB"
5. im RUN-Modus R/S
6. der zuletzt eingegebene Bytewert (beim ersten Mal 0) wird angezeigt
7. wenn alle Bytes geladen sind R/S (ohne Eingabe), weiter bei 9.
8. Bytewert eingeben, R/S, weiter bei 6.
9. nach Ende des Programmlaufes im PRGM-Modus die Anweisungen XEQ_"LB" STOP sowie die restlichen Pufferbytes löschen.

Während der Eingabe darf nur ein Wert eingetippt werden, da drei Stack-Register, LAST_X und das ALPHA-Register als Zwischenspeicher verwendet werden. Wurde ein Wert falsch eingegeben, kann er noch vor R/S mit ← korrigiert werden. Der Programmlauf sollte nicht manuell unterbrochen werden - zwischen den Zeilen 040 und 044 folgt sonst MEMORY_LOST. Hält das Programm in Zeile 008 mit der Fehlermeldung NONEXISTENT an, wurde entweder RTN (1.) nicht ausgeführt oder **LB** von der Tastatur aus gestartet.

5.4 KA - Key Assignment Programm

Synthetische Anweisungen lassen sich wegen der sofortigen Syntaxprüfung nicht mit ASN oder PASN Tasten zuordnen. Andererseits existieren synthetische Werkzeuge (Byte Grabber, Q-Loader), die nur als Key Assignment oder allenfalls als Anweisungs-Barcode funktionieren.

```
01♦LBL "KA"
02 "
03♦LBL 00
04 X<> [
05 SIGN
06 CLST
07 "PRE↑POST↑KE:
08 TONE 9
09 PROMPT
10 "ANUM"
11 PASN
12 RCL "
13 STO [
14 CLX
15  E
16 CHS
17 AROT
18 CLX
19 ATOX
20 X<> L
21 STO [
22 R↑
23 XTOA
24 R↑
25 XTOA
26 LASTX
27 XTOA
28 RCL \
29 X=0?
30 GTO 00
31 ATOX
32 X<> c
33 RCL [
34 STO IND c
35 X<>Y
36 STO c
37 CLA
38 CLST
39 .END.
```

Bild 5-2 Listing KA

Bild 5-2 zeigt ein **KA**-Programm (Autoren: Gerhard Kruse und M. G., Barcode s. 6.2), mit dem sich Tastenzuweisungen beliebiger 2-Byte-Kombinationen erzeugen lassen. Neben synthetischen Befehlen können auch normale 2-Byte-Funktionen wie SF_26 oder GTO_IND_Y zugeordnet werden.

In der Zeile 0 der Hexcode-Tabelle sind einige nicht programmierbare "Funktionen" aufgeführt, die natürlich keinen Programmcode benötigen. Ihre Codes für Key Assignments sind die Hexcodes der Anweisungen NULL (Nullbyte) bis LBL_14, welche nicht auf Tasten gelegt werden können. Einige dieser Zuordnungen zeigen ein seltsames Verhalten. So wird GTO_.. (01) nicht immer ausgeführt, ← (0B) löscht auch im RUN-Modus die aktuelle Programmzeile, die Wirkung der Zuordnung 0C hängt von der Taste ab (USER/PRGM/ALPHA).

Da **KA** eine synthetische Anweisung enthält, die sich so gut wie nicht mit dem Byte Grabber programmieren läßt, werden sinnvollerweise gleich alle synthetischen Befehle mit **LB** geladen:

Anweisung	Zeilennr.	Bytewerte für LB
"XX"	002	242 192 240
X<>M	004	206 117
TONE_9	008	159 89
RCL_├	012	144 122
STO_M	013	145 117
E	015	27
STO_M	021	145 117
RCL_N	028	144 118
X<>c	032	206 125
RCL_M	033	144 117
STO_IND_c	034	145 253
STO_c	036	145 125

Bedienung **KA**

1. im RUN-Modus XEQ_"KA"
2. Anzeige: PRE↑POST↑KEY
3. Eingabe Prefix-Byte, ENTER↑
4. Eingabe Postfix-Byte, ENTER↑ (entfällt bei 1-Byte-Anweisungen)
5. Eingabe Tastencode (wie bei ASN/PASN), R/S
6. Anzeige: PRE↑POST↑KEY
7. Eingabe des zweiten Key Assignments (wie 3. bis 5.)
8. Eingabe weiterer Tastenzuordnungen: R/S, weiter bei 2.

Mit jedem Programmlauf müssen zwei Tasten belegt werden. Das zweite Key Assignment kann mit ASN oder PASN gelöscht werden. Als einfaches Dummy-Assignment eignet sich CAT (nach PRE↑POST↑KEY nur den Keycode eingeben, R/S). Während der Eingabe ist der Stack frei (und mit 0 initialisiert), LAST_X darf nicht verändert werden. Der Programmlauf sollte nicht manuell unterbrochen werden. Zwischen den Zeilen 032 und 036 droht MEMORY_LOST; nach einem STOP zwischen den Zeilen 012 und 013 wird ein falsches Key Assignment gespeichert.

Innerhalb der Tastenzuordnungen dürfen keine Lücken entstehen. Dies ist sicher erfüllt, wenn

- noch keine Tastenzuordnungen gespeichert sind (z. B. nach CLKEYS) oder
- das jeweils neueste Key Assignment gelöscht wurde oder
- sofort nach dem Löschen einer Tastenzuordnung mit PASN (nicht mit ASN oder **KA**) ein neues Assignment gespeichert wurde.

Die **KA**-Werte des Byte Grabbers sind 247 und 55.

5.5 TQ LQ GQ XQ - Q-Loader

Die Q-Loader dienen der schnellen Programmierung kurzer synthetischer Texte in Textanweisungen sowie globalen Sprüngen und Labels. Sie können mit dem **KA**-Programm beliebigen Tasten zugeordnet werden:

Anweisung	**KA-Werte**	**Verwendungszweck**
TQ Text-Q-Loader	27 0	"Text", Exponent-E
LQ Label-Q-Loader	205 0	LBL_"Text"
GQ GTO-Q-Loader	29 0	GTO_"Text"
XQ XEQ-Q-Loader	30 0	XEQ_"Text"

Als Text-Q-Loader eignen sich sämtliche digit-entry-Anweisungen (**KA**-Werte 16 bis 28, jedoch sind bei der Ziffer 0 die Codes zu vertauschen: 0 16). Mit dem hier gewählten Text-Q-Loader läßt sich ein einzelnes Exponent-E auf einfache Weise programmieren.

Die beiden folgenden Anweisungen werden in Verbindung mit den Q-Loadern häufig benützt:

Anweisungen	KA-Werte	Verwendungszweck
RCL_M	144 117	RCL Statusregister M
STO_Q	145 121	STO Statusregister Q

Bedienung Q-Loader

- zu programmierenden Text in umgekehrter Reihenfolge im ALPHA-Register aufbauen
- RCL_M STO_Q
- im PRGM-Modus die entsprechende Q-Loader-Taste drücken
- beim Text-Q-Loader das Exponent-E löschen

Sind mehrere Texte mit dem Text-Q-Loader einzugeben, programmiert man sie in umgekehrter Reihenfolge. Alle überflüssigen Bytes (E) bis auf eines stehen dann in einer Zeile und können mit ← gelöscht werden.

5.6 Die synthetischen Anweisungen des ADRESS-Systems

Die **Bilder 5-3** bis **5-7** zeigen alle synthetischen Befehle des ADRESS-Systems und die Methoden ihrer Programmierung.

Anweisung	Vorkommen	Bytewerte für LB	Synthese
"nNEUAUFNAHME"	m023	252 110 78 69 85 65 85 70 78 65 72 77 69	"NNEUAUFHNAHME" BST **BG** SST ← RND BST BST **BG** DEL_002
"äAENDERN"	m027	248 22 65 69 78 68 69 82 78	"AAENDERN" BST **BG** SST ← 6 PACK BST BST **BG** DEL_002
"lLOESCHEN"	m029	249 108 76 79 69 83 67 72 69 78	"LLOESCHEN" BST **BG** SST ← HMS BST BST **BG** DEL_002
"zZAEHLEN"	m033	248 122 90 65 69 72 76 69 78	"ZZAEHLEN" BST **BG** SST ← SIGN BST BST **BG** DEL_002
"iINHALT	m035	247 105 73 78 72 65 76 84	"IINHALT" BST **BG** SST ← FRC BST BST **BG** DEL_002
"pPROGRAMM"	m037	249 112 80 82 79 71 82 65 77 77	"PPROGRAMM" BST **BG** SST ← CLΣ BST BST **BG** DEL_002
"X&X1H"	m006	245 27 38 107 49 72	**TQ** (72 49 107 38 27)
"#"	a253 ä228 d029 l194 l240 I376	241 35	STO_IND_Z RCL_03 BST BST **BG** ← oder **TQ** (35)
"|-#"	a323 n039	242 127 35	STO_IND_Y CLD RCL_03 BST BST **BG** ← **TQ** (35 127)
"&"	a247 a342 I382	241 38	STO_IND_Z RCL_06 BST BST **BG** ← oder **TQ** (38)
"|-&"	a325	242 127 38	STO_IND_Y CLD RCL_06 BST BST **BG** ← oder **TQ** (38 127)
"!"	a344	241 33	STO_IND_Z RCL_01 BST BST **BG** ← oder **TQ** (33)
"|-!"	a291	242 127 33	STO_IND_Y CLD RCL_01 BST BST **BG** ← **TQ** (33 127)
"µ"	l220 l234 n033	241 12	STO_IND_Z LBL_11 BST BST **BG** ← oder **TQ** (12)
"a"	BE002 IN002 IN076 UP002 UP040 c107 S350	241 225	**TQ** (225)

Bild 5-3/1 synthetische Textanweisungen des ADRESS-Systems

Anweisung	Vorkommen	Bytewerte für LB	Synthese
"A"	BE039 IN009 IN023 IN081 UP009 UP027 UP048 m062	241 193	TQ (193
"├A"	IN012 IN086 UP012 UP054 m389 m395	242 127 193	TQ (193 127)
"AA"	m001	242 193 193	TQ (193 193)

Bild 5-3/2 synthetische Textanweisungen des ADRESS-Systems

Anweisung	Vorkommen	Bytewerte für LB	Synthese
LBL "E"	E233	192 0 242 0 69	LQ (69)
LBL "F"	F064	192 0 242 0 70	LQ (70)
LBL "I"	I366	192 0 242 0 73	LQ (73)
LBL "J"	J400	192 0 242 0 74	LQ (74)
LBL "a"	a236 m391	192 0 242 0 97	LQ (97)
LBL "ä"	ä075 m392	192 0 242 0 22	LQ (22)
LBL "c"	c792	192 0 242 0 99	LQ (99)
LBL "d"	d002 m387	192 0 242 0 100	LQ (100)
LBL "e"	e075	192 0 242 0 101	LQ (101)
LBL "i"	i448 m394	192 0 242 0 105	LQ (105)
LBL "l"	l185 m386	192 0 242 0 108	LQ (108)
LBL "n"	n002 m390	192 0 242 0 110	LQ (110)
LBL "p"	p246 m388	192 0 242 0 112	LQ (112)
LBL "z"	z422 m393	192 0 242 0 122	LQ (122)
LBL "A"	m010	192 0 242 0 193	LQ (193)
LBL "ĀA"	n001	192 0 243 0 193 193	LQ (193 193)
LBL "ĀĀA"	d001	192 0 244 0 193 193 193	LQ (193 193 193)

Bild 5-4 synthetische Labelanweisungen des ADRESS-Systems

Anweisung	Vorkommen	Bytewerte für LB	Synthese
GTO_"F"	ä209 n029 n074 E337	29 241 70	GQ (70)
GTO_"A"	a250 a299 a313 ä231 c116 d047 i463 l229 l237 n069 p247 z437 F073	29 241 193	GQ (193)
XEQ_"c"	IN030 UP030 K125	30 241 99	XQ (99)
XEQ_"A"	i449 z423	30 241 65	XQ (65)
XEQ_"D"	a260 ä100 d004 n004	30 241 68	XQ (68)
XEQ_"E"	a266 ä106 d006 n007 P174	30 241 69	XQ (69)
XEQ_"I"	a258 l230 n005	30 241 73	XQ (73)
XEQ_"J"	a238 a262 ä080 ä096 d008 d014 l187 l212 l227 m054 A489 I368	30 241 74	XQ (74)

Bild 5-5 synthetische globale Sprunganweisungen des ADRESS-Systems

Anweisung	Vorkommen	Bytewerte für LB	Synthese
GTO_00	IN107 UP079 m048	208 0 0	STO_IND_80 VIEW_00 BST BST BG ←
GTO_01	d114 d122 d134 m044 m051	208 0 1	STO_IND_80 VIEW_01 BST BST BG ←
GTO_06	E281 E285	208 0 6	STO_IND_80 VIEW_06 BST BST BG ←
GTO_08	a407	208 0 8	STO_IND_80 VIEW_08 BST BST BG ←
GTO_11	P208 P218 P223	208 0 11	STO_IND_80 VIEW_11 BST BST BG ←
GTO_14	E240	208 0 14	STO_IND_80 VIEW_14 BST BST BG ←

Bild 5-6 synthetische 3-Byte-GTO-Anweisungen des ADRESS-Systems

Anweisung	Vorkommen	Bytewerte für LB	Synthese
RCL_b	IN011 IN085 UP011 UP053 a246 d094 l233 D361 E279 E340 F068 I375	144 124	RCL_IND_16 MEAN BST BST **BG** ←
STO_b	IN017 IN091 UP017 UP060 a256 d097 l245 D364 E322 E346 F074 I385	145 124	STO_IND_17 MEAN BST BST **BG** ←
STO_c	IN108 UP080 e077	145 125	STO_IND_17 SDEV BST BST **BG** ←
E	n028 E291	27	EEX BST ENTER↑ **BG** ← oder **TQ** SST ←
TONE_7	F069	159 57	STO_IND_31 STO_09 BST BST **BG** ←
TONE_2	F070	159 72	STO_IND_31 Σ- $\overline{\text{BST}}$ BST **BG** ←

Bild 5-7 diverse synthetische Anweisungen des ADRESS-Systems

6 Anhang

6.1 Cross Reference

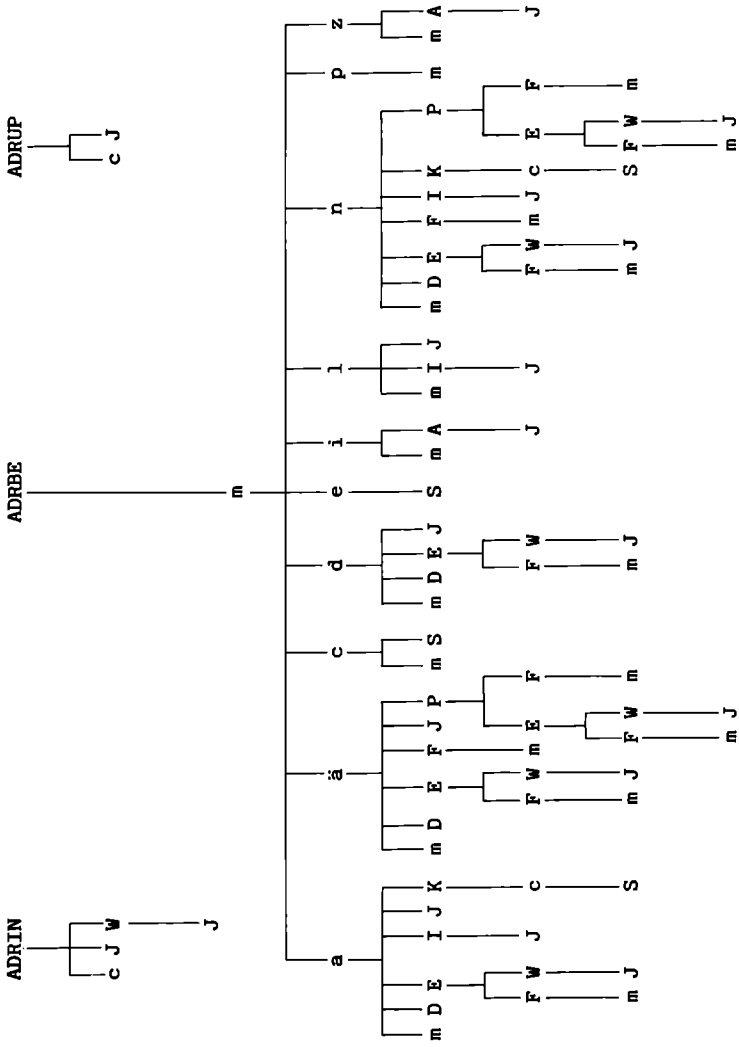


Bild 6-1 Cross Reference: Modulstruktur

Mod	Reg. 04	Reg. 05	Flag 05	Flag 06	Flag 07	Flag 08	Flag 09	Flag 10
IN	**KOBE** als Zeichen	**KUNT** als Zeichen	ENDE gewählt	KOPIE gewählt	IN (**c**)			
UP			ENDE gewählt	KOPIE gewählt	IN (**c**)			
a	Zeig. a. **AZEICHEN/ VZEICHEN**	OUT (**K**)	OUT (**E**)	IN (**E**)	AUSTAUSCH gewählt		**AZEICHEN** i. letzt. des Felds	mind. 1 **DSN** vor-ausgew.
ä	Zeig. a. **AZEICHEN/ VZEICHEN**		OUT (**E**)	IN (**E**)	RELATIV gewählt		**AZEICHEN** i. letzt. des Felds	
c					IN: ohne Schreib-/ Leseoper.			
d	**AFELD**	Zeiger a. **AFELD/ VFELD**	OUT (**E**)	IN (**E**)	NUR GROSS gewählt	**VFELD** ist leer	1. Zeile des Da-tensatzes DIALOG gewählt	SCHNELL gewählt
l	Zeiger auf **DSN**							
m						erster Durchgang		
n	**CODE**	OUT (**K**)	OUT(**E**)	IN(**E**)				
E	Zeiger auf **AFELD**	**NUMMER**	OUT: Eingabe erfolgte	IN: keine Änderung des **DSN**			R.sprung-adresse unbekannt	
K	OUT: **DSN**				IN (**c**)			
P	1. Zeich. in **AFELD**		OUT (**E**)	IN (**E**)				

Register 00 bis 03 enthalten **ZTOT** und **ZAUS** sowie **KOBE** und **KUNT** (ASCII-Codes).

Bild 6-2 Cross Reference: variable Datenregister und Flags

6.2 Barcodes

```
FORMAT
PRGM REGS NEEDED: 42
```

ROW 1 (1-3)

ROW 2 (4-8)

ROW 3 (8-10)

ROW 4 (10-11)

ROW 5 (11-14)

ROW 6 (14-17)

ROW 7 (17-19)

ROW 8 (20-22)

ROW 9 (23-25)

ROW 10 (26-34)

ROW 11 (35-43)

ROW 12 (43-44)

ROW 13 (44-50)

FORMAT
PRGM REGS NEEDED. 42

ROW 14 (51-60)

ROW 15 (61-66)

ROW 16 (66-68)

ROW 17 (69-76)

ROW 18 (77-85)

ROW 19 (86-98)

ROW 20 (99-101)

ROW 21 (102-110)

ROW 22 (110-113)

ROW 23 (114)

```
ADRBE
PRGM REGS NEEDED:  13
```

ROW 1 (1-3)

ROW 2 (4-7)

ROW 3 (7-14)

ROW 4 (15-21)

ROW 5 (21-29)

ROW 6 (29-36)

ROW 7 (36-41)

ADRIN
PRGM REGS NEEDED: 43

ROW 1 (1-3)

ROW 2 (4-11)

ROW 3 (12-17)

ROW 4 (18-22)

ROW 5 (23-26)

ROW 6 (27-31)

ROW 7 (32-38)

ROW 8 (39-42)

ROW 9 (42-47)

ROW 10 (47-53)

ROW 11 (54-58)

ROW 12 (58-64)

ROW 13 (65-68)

ROW 14 (69-76)

ROW 15 (77-83)

ROW 16 (83-89)

```
ADRIN
PRGM REGS NEEDED:  43
```

ROW 17 (90-94)

ROW 18 (94-99)

ROW 19 (99-105)

ROW 20 (105-112)

ROW 21 (113-114)

ROW 22 (114-121)

ROW 23 (122-131)

ROW 24 (131)

ADRUP
PRGM REGS NEEDED: 28

ROW 1 (1-3)

ROW 2 (4-11)

ROW 3 (12-17)

ROW 4 (18-22)

ROW 5 (22-27)

ROW 6 (27-31)

ROW 7 (32-36)

ROW 8 (36-42)

ROW 9 (43-47)

ROW 10 (48-54)

ROW 11 (54-60)

ROW 12 (61-65)

ROW 13 (65-71)

ROW 14 (71-76)

ROW 15 (77-81)

ROW 16 (81)

A
PRGM REGS NEEDED: 147

ROW 1 (1-6)

ROW 2 (6-11)

ROW 3 (11-15)

ROW 4 (15-21)

ROW 5 (22-23)

ROW 6 (23-25)

ROW 7 (26-28)

ROW 8 (28-30)

ROW 9 (30-32)

ROW 10 (32-34)

ROW 11 (35-37)

ROW 12 (37-39)

ROW 13 (39-41)

ROW 14 (41-45)

ROW 15 (45-51)

ROW 16 (52-60)

A
PRGM REGS NEEDED: 147

ROW 17 (60-65)

ROW 18 (65-69)

ROW 19 (69-75)

ROW 20 (75-79)

ROW 21 (79-82)

ROW 22 (83-93)

ROW 23 (93-94)

ROW 24 (94-101)

ROW 25 (102-108)

ROW 26 (109-115)

ROW 27 (115-119)

ROW 28 (119-122)

ROW 29 (123-127)

ROW 30 (128-134)

ROW 31 (135-139)

ROW 32 (139-147)

A
PRGM REGS NEEDED: 147

ROW 33 (148-156)

ROW 34 (156-166)

ROW 35 (167-173)

ROW 36 (173-175)

ROW 37 (175-181)

ROW 38 (181-187)

ROW 39 (188-194)

ROW 40 (195-201)

ROW 41 (202-208)

ROW 42 (209-216)

ROW 43 (216-222)

ROW 44 (222-227)

ROW 45 (228-234)

ROW 46 (235-241)

ROW 47 (242-250)

ROW 48 (250-253)

A
PRGM REGS NEEDED: 147

ROW 49 (254-256)

ROW 50 (256-258)

ROW 51 (258-262)

ROW 52 (262-264)

ROW 53 (264-266)

ROW 54 (266-271)

ROW 55 (272-277)

ROW 56 (277-283)

ROW 57 (284-292)

ROW 58 (292-299)

ROW 59 (299-306)

ROW 60 (307-312)

ROW 61 (313-321)

ROW 62 (321-329)

ROW 63 (330-337)

ROW 64 (337-345)

A
PRGM REGS NEEDED: 147

ROW 65 (346-350)

ROW 66 (351-356)

ROW 67 (357-363)

ROW 68 (364-367)

ROW 69 (367-368)

ROW 70 (369-377)

ROW 71 (377-385)

ROW 72 (386-388)

ROW 73 (388-391)

ROW 74 (391-394)

ROW 75 (394-398)

ROW 76 (398-399)

ROW 77 (399-403)

ROW 78 (404-412)

ROW 79 (412-414)

ROW 80 (414)

AA
PRGM REGS NEEDED: 142

ROW 1 (1-4)

ROW 2 (4-7)

ROW 3 (7-13)

ROW 4 (14-22)

ROW 5 (22-29)

ROW 6 (29-36)

ROW 7 (37-44)

ROW 8 (45-49)

ROW 9 (49-57)

ROW 10 (57-64)

ROW 11 (65-71)

ROW 12 (71-76)

ROW 13 (77-79)

ROW 14 (79-85)

ROW 15 (85-90)

ROW 16 (91-95)

AA
PRGM REGS NEEDED: 142

ROW 17 (95-101)

ROW 18 (101-104)

ROW 19 (104-109)

ROW 20 (110-117)

ROW 21 (117-123)

ROW 22 (124-131)

ROW 23 (132-140)

ROW 24 (140-147)

ROW 25 (148-154)

ROW 26 (154-161)

ROW 27 (162-169)

ROW 28 (170-177)

ROW 29 (177-184)

ROW 30 (185-191)

ROW 31 (192-198)

ROW 32 (199-208)

AA
PRGM REGS NEEDED: 142

ROW 33 (209-216)

ROW 34 (217-224)

ROW 35 (225-231)

ROW 36 (232-237)

ROW 37 (237-238)

ROW 38 (239-247)

ROW 39 (247-254)

ROW 40 (254-261)

ROW 41 (261-263)

ROW 42 (264-265)

ROW 43 (265-270)

ROW 44 (271-279)

ROW 45 (280-287)

ROW 46 (288-294)

ROW 47 (295-301)

ROW 48 (301-310)

AA
PRGM REGS NEEDED: 142

ROW 49 (311-318)

ROW 50 (318-325)

ROW 51 (325-332)

ROW 52 (333-340)

ROW 53 (340-347)

ROW 54 (348-356)

ROW 55 (356-363)

ROW 56 (363-371)

ROW 57 (372-379)

ROW 58 (379-386)

ROW 59 (387-394)

ROW 60 (394-402)

ROW 61 (403-410)

ROW 62 (410-420)

ROW 63 (421-424)

ROW 64 (424)

AA
PRGM REGS NEEDED: 142

ROW 65 (425-427)

ROW 66 (428-429)

ROW 67 (429-435)

ROW 68 (436-443)

ROW 69 (443-449)

ROW 70 (450-457)

ROW 71 (458-466)

ROW 72 (467-474)

ROW 73 (474-479)

ROW 74 (480-487)

ROW 75 (487-488)

ROW 76 (489-491)

ROW 77 (492)

AAA
PRGM REGS NEEDED: 111

ROW 1 (1-3)

ROW 2 (4-6)

ROW 3 (7-10)

ROW 4 (10-13)

ROW 5 (13-20)

ROW 6 (21-27)

ROW 7 (28-35)

ROW 8 (36-43)

ROW 9 (44-51)

ROW 10 (51-58)

ROW 11 (59-66)

ROW 12 (66-74)

ROW 13 (75-81)

ROW 14 (82-89)

ROW 15 (90-96)

ROW 16 (96-103)

AAA
PRGM REGS NEEDED: 111

ROW 17 (104-111)

ROW 18 (111-118)

ROW 19 (118-124)

ROW 20 (125-132)

ROW 21 (132-139)

ROW 22 (140-149)

ROW 23 (150-159)

ROW 24 (159-167)

ROW 25 (168-175)

ROW 26 (176-183)

ROW 27 (184-186)

ROW 28 (186-191)

ROW 29 (192-199)

ROW 30 (199-207)

ROW 31 (207-214)

ROW 32 (214-222)

AAA
PRGM REGS NEEDED: 111

ROW 33 (222-226)

ROW 34 (226-230)

ROW 35 (231-237)

ROW 36 (238-245)

ROW 37 (245-251)

ROW 38 (252-264)

ROW 39 (265-277)

ROW 40 (278-290)

ROW 41 (291-303)

ROW 42 (304-316)

ROW 43 (317-329)

ROW 44 (330-342)

ROW 45 (343-355)

ROW 46 (356-368)

ROW 47 (369-381)

ROW 48 (382-394)

AAA
PRGM REGS NEEDED: 111

ROW 49 (395-407)

ROW 50 (408-420)

ROW 51 (421-433)

ROW 52 (434-446)

ROW 53 (447-459)

ROW 54 (460-472)

ROW 55 (473-485)

ROW 56 (486-498)

ROW 57 (499-511)

ROW 58 (512-524)

ROW 59 (525-536)

ROW 60 (537)

```
ADRRE
PRGM REGS NEEDED:  19
```

ROW 1 (1-4)

ROW 2 (4-10)

ROW 3 (10-16)

ROW 4 (16-23)

ROW 5 (23-30)

ROW 6 (30-36)

ROW 7 (37-43)

ROW 8 (43-48)

ROW 9 (49-54)

ROW 10 (54-56)

ROW 11 (57)

```
LB
PRGM REGS NEEDED:  13
```

ROW 1 (1-5)

ROW 2 (6-14)

ROW 3 (14-25)

ROW 4 (26-33)

ROW 5 (34-41)

ROW 6 (42-49)

ROW 7 (49-51)

KA
PRGM REGS NEEDED: 13

ROW 1 (1-5)

ROW 2 (6-7)

ROW 3 (7-12)

ROW 4 (13-21)

ROW 5 (21-29)

ROW 6 (30-36)

ROW 7 (37-39)

6.3 Literaturhinweise

/1/ Gehret, Michael: Sprünge und Labels, in: PRISMA, CCD e. V., 1982, Nr. 10/11, S. 12/29 und Nr. 12, S. 8

/2/ Gehret, Michael: Sprünge ohne Labels, in: PRISMA, CCD e. V., 1982, Nr. 12, S. 7/8

/3/ William C. Wickes: Byte Jumping, in: PPC Calculator Journal, 1980, V7N4, S. 26-28

/4/ William C. Wickes: Synthetic Programming on the HP-41C, 1981, Larken Publications, 4517 NW Queens Ave., Corvallis OR 97330, U.S.A.

/5/ Matthias Grabiak: Brief v. 14.09.80 an den Hewlett-Pakkard Anwender-Club

/6/ Gosteli, Erwin: Generalized Fn Byte Maskers, in: PPC Calculator Journal, 1981, V8N5, S. 11/12

weitere Literaturempfehlungen:

Jarett, Keith: HP-41 Synthetic Programming Made Easy, 1982, Synthetix, P.O. Box 113, Manhattan Beach CA 90266, U.S.A.

Dearing, John (ed.): Calculator Tips and Routines, Corvallis Software, Inc., P.O. Box 1412, Corvallis OR 97339-1412, U.S.A.

Aktuelle Informationen bieten die beiden folgenden Clubs:

CCD Computerclub Deutschland e. V., Postfach 2129, 6242 Kronberg 2

PPC, 2545 West Camden Place, Santa Ana CA 92704, U.S.A.
German Chapter: PPC Dortmund, Heinz-Jörg Plegge, Am Roggen Kamp 20, D-4400 Münster

Eine handliche Quick Reference Card ist von Synthetix, P.O. Box 113, Manhattan Beach CA 90266, U.S.A., erhältlich

6.4 Register

HP-41C QUICK REFERENCE CARD FOR SYNTHETIC PROGRAMMING

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
8	DEG IND 00 128 ◆	RAD IND 01 129 ⌐	GRAD IND 02 130 ¯	ENTER↑ IND 03 131 ←	STOP IND 04 132 α	RTN IND 05 133 β	BEEP IND 06 134 Γ	CLA IND 07 135 ↓	ASHF IND 08 136 Δ	PSE IND 09 137 σ	CLRG IND 10 138 ◆	AOFF IND 11 139 ␣	AON IND 12 140 μ	OFF IND 13 141 √	PROMPT IND 14 142 ▾	ADV IND 15 143 ↟	8
9	RCL IND 16 144 θ	STO IND 17 145 Ω	ST+ IND 18 146 δ	ST− IND 19 147 ä	ST ✳ IND 20 148 ä	ST/ IND 21 149 ä	ISG IND 22 150 ä	DSE IND 23 151 ö	VIEW IND 24 152 ö	Σ REG IND 25 153 Ü	ASTO IND 26 154 ü	ARCL IND 27 155 ß	FIX IND 28 156 æ	SCI IND 29 157 ≠	ENG IND 30 158 £	TONE IND 31 159 ▒	9
A	XR 0-3 IND 32 160	XR 4-7 IND 33 161 !	XR8-11 IND 34 162 "	X12-15 IND 35 163 #	X16-19 IND 36 164 $	X20-23 IND 37 165 %	X24-27 IND 38 166 &	X28-31 IND 39 167 '	SF IND 40 168 (	CF IND 41 169)	FS?C IND 42 170 *	FC?C IND 43 171 +	FS? IND 44 172 ,	FC? IND 45 173 −	GTO XEQ IND IND 46 174 .	SPARE IND 47 175 /	A
B	SPARE IND 48 176 0	GTO 00 IND 49 177 1	GTO 01 IND 50 178 2	GTO 02 IND 51 179 3	GTO 03 IND 52 180 4	GTO 04 IND 53 181 5	GTO 05 IND 54 182 6	GTO 06 IND 55 183 7	GTO 07 IND 56 184 8	GTO 08 IND 57 185 9	GTO 09 IND 58 186 :	GTO 10 IND 59 187 ;	GTO 11 IND 60 188 <	GTO 12 IND 61 189 =	GTO 13 IND 62 190 >	GTO 14 IND 63 191 ?	B
C	GLOBAL IND 64 192 @	GLOBAL IND 65 193 A	GLOBAL IND 66 194 B	GLOBAL IND 67 195 C	GLOBAL IND 68 196 D	GLOBAL IND 69 197 E	GLOBAL IND 70 198 F	GLOBAL IND 71 199 G	GLOBAL IND 72 200 H	GLOBAL IND 73 201 I	GLOBAL IND 74 202 J	GLOBAL IND 75 203 K	GLOBAL IND 76 204 L	GLOBAL IND 77 205 M	X<>-- IND 78 206 N	LBL -- IND 79 207 O	C
D	GTO -- IND 80 208 P	GTO -- IND 81 209 Q	GTO -- IND 82 210 R	GTO -- IND 83 211 S	GTO -- IND 84 212 T	GTO -- IND 85 213 U	GTO -- IND 86 214 V	GTO -- IND 87 215 W	GTO -- IND 88 216 X	GTO -- IND 89 217 Y	GTO -- IND 90 218 Z	GTO -- IND 91 219 [	GTO -- IND 92 220 \	GTO -- IND 93 221]	GTO -- IND 94 222 ↑	GTO -- IND 95 223 _	D
E	XEQ -- IND 96 224 ▾	XEQ -- IND 97 225 a	XEQ -- IND 98 226 b	XEQ -- IND 99 227 c	XEQ -- IND100 228 d	XEQ -- IND101 229 e	XEQ -- IND102 230 f	XEQ -- IND103 231 g	XEQ -- IND104 232 h	XEQ -- IND105 233 i	XEQ -- IND106 234 j	XEQ -- IND107 235 k	XEQ -- IND108 236 l	XEQ -- IND109 237 m	XEQ -- IND110 238 n	XEQ -- IND111 239 o	E
F	TEXT 0 IND T 240 p	TEXT 1 IND Z 241 q	TEXT 2 IND Y 242 r	TEXT 3 IND X 243 s	TEXT 4 IND L 244 t	TEXT 5 IND M 245 u	TEXT 6 IND N \ 246 v	TEXT 7 IND O] 247 w	TEXT 8 IND P↑ 248 x	TEXT 9 IND Q_ 249 y	TEXT10 IND ⊢▾ 250 z	TEXT11 IND a 251 ▪	TEXT12 IND b 252 |	TEXT13 IND c 253 →	TEXT14 IND d 254 Σ	TEXT15 IND e 255 ⊢	F
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	

For price information and a list of dealers in your area, send a self-addressed stamped envelope to: SYNTHETIX, 1540 Mathews Ave., Manhattan Beach, CA 90266, USA

Structure of multi-byte instructions

Two-byte instructions
STO 16=145,16 DSE IND 55 =151,183
LBL e =207,127 FS?C IND Y =170,242
RCL b =144,124 TONE 89 =159,89
X<>M =206,117 ST+ IND N =146,246
LBL Q =207,121 VIEW H(109)=152,109
Two-byte special cases
GTO IND=174,reg. XEQ IND=174,128+r
GTO IND 09=174,9 XEQ IND X=174,243

XROM i,j =160+i/4,64(i mod 4)+j
WSTS =XROM 30,10 =167,138
short form GTO =177+label,0
GTO 12 =189,0
Three-byte instructions
long form GTO =208,0,label
GTO 32 =208,0,32
XEQ =224,0,label
XEQ D =224,0,105
END =192,0,9+sum of status indicators
32(.END.), 4(rePACK), 2(decompile)

Variable length instructions
TEXT =240+n, n character bytes
Append symbol counts as first char.
▾& =241,38 ▾⊢)? =243,127,41,63
GTO ▾ =29,240+n, n character bytes
GTO ▾XYZ =29,243,88,89,90
XEQ ▾ =30,240+n, n character bytes
XEQ ▾A =30,241,65 (synthetic)
LBL ▾ =192,0,241+n, (key), n chars.
LBL ▾: =192,0,242,0,58 (synthetic)

HP-41C QUICK REFERENCE CARD FOR SYNTHETIC PROGRAMMING

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	CAT	@c (GTO..)	DEL	COPY	CLP	R/S	SIZE	BST	SST	ON	PACK	←(PRGM)	USR/P/A	2 _ _	SHIFT	ASN	
0	NULL 00 ⁻ 0 ◆	LBL 00 01 x̄ 1 ×	LBL 01 02 ▩ 2 x̄	LBL 02 03 ▩ 3 ←	LBL 03 04 x̂ 4 α	LBL 04 05 x̂ 5 ß	LBL 05 06 ŷ 6 Γ	LBL 06 07 ▩ 7 ↓	LBL 07 08 ▩ 8 Δ	LBL 08 09 ▩ 9 σ	LBL 09 10 ▩ 10 ◆	LBL 10 11 ▩ 11 ↘	LBL 11 12 ⁻ 12 ↵	LBL 12 13 ⁻ 13 ⁻	LBL 13 14 ▩ 14 ⁻	LBL 14 15 ▩ 15 ↟	0
1	0 16 ▩ 16 θ	1 17 ▩ 17 Ω	2 18 ▩ 18 δ	3 19 ▩ 19 Å	4 20 ▩ 20 ä	5 21 ▩ 21 Ä	6 22 ▩ 22 ö	7 23 ▩ 23 Ö	8 24 ▩ 24 ö	9 25 ▩ 25 Ü	. 26 ▩ 26 ü	EEX 27 ▩ 27 ß	NEG 28 ▩ 28 ⁻	GTO ᵀ 29 ⁻ 29 ≠	XEQ ᵀ 30 ▩ 30 £	W ᵀ 31 ▩ 31 ▩	1
2	RCL 00 32 32	RCL 01 33 ; 33 !	RCL 02 34 " 34 "	RCL 03 35 ⁄ 35 #	RCL 04 36 % 36 $	RCL 05 37 % 37 %	RCL 06 38 & 38 &	RCL 07 39 ' 39 '	RCL 08 40 (40 (	RCL 09 41) 41)	RCL 10 42 * 42 *	RCL 11 43 + 43 +	RCL 12 44 , 44 ,	RCL 13 45 - 45 –	RCL 14 46 . 46 -	RCL 15 47 / 47 /	2
3	STO 00 48 0 48 0	STO 01 49 1 49 1	STO 02 50 2 50 2	STO 03 51 3 51 3	STO 04 52 4 52 4	STO 05 53 5 53 5	STO 06 54 6 54 6	STO 07 55 7 55 7	STO 08 56 8 56 8	STO 09 57 9 57 9	STO 10 58 : 58 :	STO 11 59 ; 59 ;	STO 12 60 < 60 <	STO 13 61 = 61 =	STO 14 62 > 62 >	STO 15 63 ? 63 ?	3
4	+ 64 @ 64 @	– 65 A 65 A	* 66 B 66 B	/ 67 C 67 C	X<Y? 68 D 68 D	X>Y? 69 E 69 E	X≤Y? 70 F 70 F	Σ+ 71 G 71 G	Σ- 72 H 72 H	HMS+ 73 I 73 I	HMS- 74 J 74 J	MOD 75 K 75 K	% 76 L 76 L	%CH 77 M 77 M	P→R 78 N 78 N	R→P 79 O 79 O	4
5	LN 80 P 80 P	X↑2 81 Q 81 Q	SQRT 82 R 82 R	Y↑X 83 S 83 S	CHS 84 T 84 T	E↑X 85 U 85 U	LOG 86 V 86 V	10↑X 87 W 87 W	E↑X-1 88 X 88 X	SIN 89 Y 89 Y	COS 90 Z 90 Z	TAN 91 [91 [	ASIN 92 \ 92 \	ACOS 93] 93]	ATAN 94 ↑ 94 ↑	→DEC 95 _ 95 _	5
6	1/X 96 ⁻ 96 ⁻	ABS 97 a 97 a	FACT 98 b 98 b	X≠0? 99 c 99 c	X>0? 100 d 100 d	LN1+X 101 e 101 e	X<0? A ▩ 102 f	X=0? B ▩ 103 g	INT C ▩ 104 h	FRC D ▩ 105 i	D→R E ▩ 106 j	R→D F ▩ 107 k	→HMS G ▩ 108 l	→HR H ▩ 109 m	RND I ▩ 110 n	→OCT J ▩ 111 o	6
7	CLΣ T ▩ 112 p	X<>Y Z ▩ 113 q	PI Y ▩ 114 r	CLST X ▩ 115 s	R↑ L ▩ 116 t	RDN M [▩ 117 u	LASTX N \ ▩ 118 v	CLX O] ▩ 119 w	X=Y? P↑ ▩ 120 x	X≠Y? Q_ ▩ 121 y	SIGN ⊢ ᵀ ▩ 122 z	X≤0? a ▩ 123 ⁻	MEAN b ▩ 124	SDEV c ▩ 125 →	AVIEW d Σ 126 Σ	CLD e ⊢ 127 ⊢	7
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	
	00 01 02 03	04 05 06 07	08 09 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31	32 33 34 35	36 37 38 39	40 41 42 43	44 45 46 47	48 49 50 51	52 53 54 55	← bit numbers in a 7-byte register		

FLAGS (Register d)

00-10 general purpose
11 auto execute
12 doublewide
13 lower case
14 overwrite
15-16 IL printer
0 0 MAN
0 1 NORM
1 0 TRACE
1 1 TR/STACK
17 record incomplete
18] general use
19] cleared at
20] turn-on
21 prtr enable
22 num. entry
23 alpha entry
24 range ignore
25 error ignore
26 audio enable
27 USER mode
28 dec./comma
29 digit grouping
30 CAT
31 timer DMY/MDY
32 manual IL I/O
33 IL absolute manual
34 not used
35 not used
36-39 number of digits
40-41 display
0 0 SCI
0 1 ENG
1 0 FIX
1 1 FIX/ENG
42-43 trig mode
0 0 DEG
0 1 RAD
1 0 GRAD
1 1 RAD
44 cont. ON
45 system data entry
46 partial key sequence
47 SHIFT
48 ALPHA
49 low BAT
50 message
51 SST
52 PGRM
53 I/O
54 PSE
55 printer existence

Michael Gehret

Softwareentwicklung am Beispiel einer Dateiverwaltung (HP-41)

Das Buch erfüllt mehrere Funktionen:

- Für den reinen Anwender enthält es ein bewährtes Programmpaket zur komfortablen Verwaltung von Adressenmaterial.
- Der Programmierer kann aus den Modulen der Adressenverwaltung nach einfachen Modifikationen ein System zur Verwaltung eines beliebigen strukturierten Datenbestandes zusammenstellen und optimieren.
- Das Buch zeigt den Einsatz moderner Methoden der Softwareentwicklung (Modularisierung, schrittweise Verfeinerung) mit dem HP-41.
- Schließlich findet der interessierte Leser zwei einfache Verfahren zur Programmoptimierung, zahlreiche Tricks zur Verringerung der Rechenzeit und des Speicherbedarfs sowie interessante Details wie etwa die Speicherformate des Rechners und des IL-Massenspeichers. Außerdem wird ein kurzer, praxisbezogener Zugang zur synthetischen Programmierung vermittelt.

ISBN 3-528-04286-9