Das erweiterte DATAS - System

Datenstrukturen in assoziativer Speicherung

von

G. Weck

Institut für Angewandte Mathematik
und Informatik der Universität des
Saarlandes

D-66 Saarbrücken
Bundesrepublik Deutschland

Oktober 1975

Bericht Nr. A 75-11

Inhaltsverzeichnis

Vorwort

DATAS (Datenstrukturen in Assoziativer Speicherung) ist eine assoziative Datenstruktur. Die Entwicklung eines implementierungsfähigen Konzepts der Struktur wurde als Diplomarbeit für Herrn Dipl.-Ing. V. Glatzer vergeben. Die Diplomarbeit erfolgte 1971 am Lehrstuhl und Institut für Informationsverarbeitung der Technischen Universität Berlin. In Zusammenarbeit der Forschungsgruppe Rechnerorganisation und Computer Graphics mit der Forschungsgruppe Schaltkreise und Schaltwerke im Fachbereich Angewandte Mathematik und Informatik der Universität des Saarlandes wurde die Diplomarbeit von Herrn Glatzer überarbeitet und die dabei entstandene Version von DATAS für die Telefunken TR440 implementiert /1/.

In der folgenden Zeit wurde das System erweitert, um die Simulation beliebiger Assoziativer Datenstrukturen auf der Basis eines mathematischen Modells (SAD-Strukturunabhängige Assoziative Datenstruktur) zu ermöglichen /2/. Der praktische Einsatz von DATAS zeigte im Laufe der Zeit einige Unzulänglichkeiten des ursprünglichen Implementierungskonzeptes auf, die durch eine neue Überarbeitung und Erweiterung des Systems inzwischen behoben wurden. Als wesentlichste Änderungen gegenüber dem ursprünglichen, in /1/ beschriebenen DATAS-System sind zu nennen :

1. Erweiterung der Befehlsliste und Vereinfachung der Syntax zur Bequemlichkeit des Benutzers;
2. Erweiterung der Systemkapazität;
3. Verringerung des Speicherbedarfs der System-Datei;
4. Beschleunigung und Optimierung des Seitentransports;
5. Bereitstellung von Datenmanipulationsdiensten und Mengenoperationen;
6. Vereinfachung der Retrieval-Operationen bei gleichzeitiger Erweiterung der Retrieval-Möglichkeiten;
7. Überlaufbehandlung im Namenspeicher und Überlaufsimulation im Triadenspeicher;
8. nachträgliche dynamische Erweiterung der Struktur ohne Informationsverlust;
9. bessere Kontrollmöglichkeiten für benutzerspezifische Programme, die mit der DATAS-Bibliothek arbeiten;

10. Erstellung von Kommandos für die TR440, die die Datei-Behandlung übernehmen und die Integrität der Datenbank auch bei Rechnerausfall schützen;
11. Erstellung eines standardisierten Anschlusses für benutzerspezifische Erweiterungen (Eine solche erweiterte DATAS-Version, die u.a. die Manipulation beliebig großer Mengen ermöglicht, wurde z.B. in Zusammenarbeit mit dem Institut für Angewandte Geodäsie, Frankfurt, entwickelt; weitere anwendungsorientierte Erweiterungen sind konzipiert.)

Der Umfang dieser Änderungen und Erweiterungen ließ den Wunsch nach einer Beschreibung des erweiterten DATAS-Systems entstehen. Diesem Wunsch soll mit dem vorliegenden Bericht, der sich auf die Version DATAS (10.24) vom 30.9.1975 bezieht, entsprochen werden.

Ich möchte an dieser Stelle allen danken, die durch Ideen, Anregung und Kritik zu der Arbeit an DATAS und damit zum Zustandekommen dieses Berichtes beigetragen haben. Besonderer Dank gebührt jedoch Herrn Prof. Dr. J. Encarnaçao, der mich bei diesem Projekt betreut und gefördert hat und in vielen fruchtbaren Diskussionen half, die Ziele zu stecken und Wege aufzuzeigen, sie zu erreichen. Schließlich möchte ich den Mitarbeitern des Rechenzentrums der Universität des Saarlandes, insbesondere Herrn B. Kett, meinen Dank für ihre Unterstützung und Hilfe bei der Behebung von Schwierigkeiten und bei der Fehlersuche aussprechen.

Das erweiterte DATAS - System

(Datenstrukturen in assoziativer Speicherung)

G. Weck

[Gültig für Systemversion (10.24), ab 30.9.1975]

1. Systembeschreibung

DATAS (DATen in Assoziativer Speicherung) ist ein System zur schnellen und bequemen Arbeit mit beliebigen, assoziativ verknüpften Datenmengen. Diese Arbeit umfaßt Ein- und Ausgabe, Veränderung, Löschung und Suchen von Daten, die bekannt oder mit bekannten Daten assoziiert sein müssen. Die verschiedenen in DATAS möglichen Assoziationen werden in den folgenden Abschnitten kurz charakterisiert; für eine ausführliche Beschreibung muß auf den vollständigen Bericht über die Implementierung von DATAS hingewiesen werden /1/.

1.1 Systemkomponenten

1.1.1 Grundelemente

Die Grundelemente von DATAS sind:

Namen
Entities
Triaden
(Facts)

Dabei werden unter Entities die kleinsten adressierbaren Informationseinheiten in DATAS verstanden. Vom Benutzer aus werden sie durch Namen (Zeichenfolgen, deren erste 10 Zeichen einschließlich blanks ausgewertet werden; Namen, die mit zwei Blanks beginnen, sind nicht erlaubt) adressiert; jede Entity darf mehrere (bis zu 15) Namen haben, die dann als Synonyme bezeichnet werden, sowie beliebige Daten enthalten. Triaden sind Tripel von Verweisen auf Entities; sie beschreiben die Assoziationen zwischen diesen. Facts repräsentieren Assoziationen zwischen beliebig vielen Entities;

sie werden durch Mengen von Triaden dargestellt.

1.1.2 Speichereinteilung

Der DATAS zur Verfügung stehende Speicherraum wird in drei Bereiche aufgeteilt :

Namenspeicher (Zuordnung von Entities zu Namen)
Triadenspeicher (Darstellung von Assoziationen durch Triaden)
Datenspeicher (enthält die Daten der Entities)

Jeder dieser drei Speicherbereiche ist in gleichgroße Seiten eingeteilt,deren jede 2048 maximal 15-Bit lange INTEGER-Zahlen enthält. Alle Seiten befinden sich auf einem externen, index-sequentiell organisierten Speichermedium (Trommel oder Platte). Dabei entspricht eine Seite genau einem Satz auf diesem Speichermedium, dessen Satzbau somit genau 1024 bzw. 2048 Maschinenworte ist, je nachdem, ob Halb- oder Ganzwortdarstellung für INGEGER-Zahlen gewählt wurde.

Die logische Organisation dieses Speichers ist dabei die, daß der erste Satz generelle Information über den Zustand des Systems sowie über den Zustand der ersten 2039 Seiten enthält; darauf folgen die Seiten der drei Speicherbereiche, jeweils von 1 an laufend durchnumeriert. Hat das System mehr als 2039 Seiten, so wird der Zustand der restlichen Seiten in den letzten Sätzen gespeichert, andernfalls entfallen diese.

Bei der Arbeit mit DATAS befindet sich jeweils je eine Seite der drei verschiedenen Typen im Kernspeicher. Der Seitentransport wird automatisch ohne Zutun des Benutzers und ohne daß dieser ihn bemerkt, von einem paging-system durchgeführt. Der Benutzer hat jedoch auch die Möglichkeit, gezielt eine bestimmte, gewünschte Seite in den Kernspeicher zu holen. Zur Beschleunigung dieses Seitentransportes werden generell nur die Seiten vom Kernspeicher auf das externe Medium zurückgeschrieben, die während ihres Aufenthaltes im Kernspeicher geändert wurden.

1.1.3 Programmstruktur

Das System arbeitet auf drei Programmebenen :

Dialogprogramm (bestehend aus einem Interpreter)
Subroutines (vom Benutzer in Batch-Anwendungen aufzurufen)
Hilfsroutinen (für den Benutzer verborgen)

Die Subroutines ermöglichen die Manipulation von Daten, Namen, Mengen und Relationen, sowie das Suchen (Retrieval) dieser Objekte, Initialisierung und Maintenance der Struktur sowie den Seitentransport. Diese Subroutines können vom Dialogprogramm sowie von jedem von einem Benutzer geschriebenen Anwendungsprogramm im Batch- oder Dialogbetrieb aufgerufen werden, sofern dieses den Aufruf von FORTRAN IV-Subroutines gestattet.

Das Dialogprogramm ermöglicht es dem Benutzer, die Subroutines direkt unter Benutzung Assembler-ähnlicher Befehle von einem Terminal aus aufzurufen. Zu diesem Zweck wurde für DATAS eine Assembler-ähnliche Datenmanipulationssprache definiert.

Die Hilfsroutinen arbeiten direkt mit Namen, Triaden und Entities. Sie sollten vom Benutzer nicht verwendet werden, da sie nicht abprüfen, ob eine von ihnen ausgeführte Operation legal ist oder die Struktur zerstört. Diese Abprüfung geschieht nur auf der Ebene der Subroutines, die diese Hilfsroutinen benötigen.

DATAS steht zur Verfügung als FORTRAN IV-Quelle nach IBM-Standard, als Bibliothek übersetzter, relativ adressierter Unterprogramme und als lauffähiges Objektprogramm für Dialogbetrieb, letzteres für die Telefunken TR 440.

1.2 Entities

DATAS kennt drei Klassen von Entities :

Klasse 1 - Relation-Entities
Klasse 2 - Single-Value-Entities
Klasse 3 - Multi-Value-Entities

Relation-Entities repräsentieren Assoziationen zwischen Entities; diese Assoziationen werden durch die im nächsten Abschnitt beschriebenen Triaden dargestellt.

Single-Value-Entities enthalten nur vom Benutzer definierte Daten oder überhaupt keine Daten; sie können durch Komponenten von Triaden oder als Elemente von Mengen referiert werden.

Multi-Value-Entities stellen Mengen dar und enthalten als Daten Zeiger auf die Elemente.

Keine Entity kann gleichzeitig mehreren dieser drei Klassen angehören, jedoch wird die Klasse von Single-Value-Entities, die als Relation oder als Menge angesprochen werden, automatisch umgeändert. Eine Umänderung zwischen Relation und Menge ist jedoch nicht vorgesehen.

Jede Entity darf mehrere (bis zu 15) Namen (Synonyme) haben und darf beliebige Daten enthalten, die lediglich der Bedingung unterliegen, daß die Daten einer Entity auf derselben Seite des Datenspeichers liegen müssen; Überlauf ist nicht vorgesehen. Daraus resultiert ein absolutes Maximum von 1786 Maschinenworten (entsprechend 3572 Zeichen) als Daten für eine Entity. Je nach der zur Verfügung stehenden Speichermenge - und bei Relation- und Multi-Value-Entities auch nach der Menge der in der Entity enthaltenen Information - kann dieser Wert unterschritten werden, da der Datenspeicher dynamisch verwaltet wird. Die genannte Systeminformation besteht bei Multi-Value-Entities aus der Anzahl und den Zeigern auf die Elemente und bei Relation-Entities aus Information über die Seiten im Triadenspeicher, die Triaden der Relation enthalten.

Intern werden die Entities direkt durch ID genannte Paare ganzer, positiver Zahlen adressiert; die erste dieser Zahlen ist die Nummer der Seite im Datenspeicher, auf der die Entity steht, die zweite ist eine (von 1 an gezählte) laufende Nummer auf dieser Seite.

Eine alternative Adressierungsmöglichkeit besteht in den Namenadressen (NA), den Adressen der Entity-Namen im ebenfalls in Seiten eingeteilten Namensspeicher, der die Zuordnung von Namen zu Entities (genauer: zu IDs) bewerkstelligt. Namenadressen bestehen, wie die IDs, aus Paaren ganzer, positiver Zahlen; die erste dieser Zahlen ist die Nummer der Seite im Namenspeicher, auf der der Name steht, die zweite ist die (von 1 an gezählte) Nummer des Speicherplatzes des Namens auf dieser Seite.

Wird eine Entity irgendwo in der Struktur durch ihr ID referiert, so wird bei Bezugnahme auf diese Referenz der Name der Entity, der dieser zuletzt zugeordnet wurde, (das aktuellste Synonym) ausgegeben. Wird die Entity durch ein NA referiert, so wird genau der auf dieser Namenadresse abgespeicherte Name (das referierte Synonym) ausgegeben.

1.3 Triaden

1.3.1 Speichermodus

Logische Assoziationen zwischen Entities werden als Elemente binärer Relationen aufgefaßt und als Tripel von IDs und/oder NAs dargestellt. Diese geordneten Tripel werden als Triaden ABC bezeichnet :

$$A \subset B' \times C'$$

$$A := \{(B,C) \mid B \in B', C \in C', H(B,C)\} = \bigcup_{\substack{B \in B' \\ C \in C' \\ H(B,C)}}^{B \atop C} \{(B,C)\}$$

Die Relation A ist dabei eine Menge geordneter Paare (B,C), die eine Bedingung H(B,C) erfüllen. B' und C' sind dabei Vor- bzw. Nachbereich der Relation; sie können gleich der Menge E aller Entities sein. Das Paar (B,C) selbst wird, wenn es in der Relation A enthalten ist, als Entry dieser Relation bezeichnet.

Jede Triade ABC besitzt einen eigenen Speichermodus s, der angibt, welche ihrer drei Komponenten IDs und welche NAs sind. Dieser Speichermodus ist eine ganze Zahl zwischen 0 und 7, deren Einsen in binärer Verschlüsselung die NA-Komponenten der Triade bezeichnen, während die Nullen den IDs entsprechen, so daß folgender Zusammenhang zwischen Speichermodus s und Aufbau der Triade besteht :

s		Triade
0	000	(ID_A, ID_B, ID_C)
1	001	(ID_A, ID_B, NA_C)
2	010	(ID_A, NA_B, ID_C)
3	011	(ID_A, NA_B, NA_C)

s		Triade
4	100	(NA_A, ID_B, ID_C)
5	101	(NA_A, ID_B, NA_C)
6	110	(NA_A, NA_B, ID_C)
7	111	(NA_A, NA_B, NA_C)

Dieser Speichermodus muß beim Abspeichern, Suchen und Löschen von Triaden explizit angegeben werden, wenn er von 0 verschieden ist; beim Suchen sind jedoch die Bitstellen, die den gesuchten Komponenten zugeordnet sind, irrelevant.

1.3.2 Permutationstyp

Jede Relation besitzt einen Permutationstyp p (0,1 oder 2), der bestimmt, welche Permutationen von Triaden dieser Relation abgespeichert werden, und daher auch, welche Typen assoziativer Fragen zu dieser Relation beantwortet können.

Bei einer Relation A mit p=0 wird jeweils nur die Permutation ABC der Triaden abgespeichert, so daß die Fragetypen

F0 (A,B,C) : existiert der Zusammenhang ABC, d.h. A ∋ (B,C) ?

F1 (A,B,?) : welche Entity (oder Entities) steht (stehen) zu B in der Relation A ?

F4 (A,?,?) : gibt es Entities, die in der Relation A zueinander stehen ?

beantwortet werden können. Bei einer Relation mit p=1 wird zu-

sätzlich zur Permutation ABC auch die Permutation BCA abgespeichert, so daß zusätzlich die Fragetypen

F3 (?,B,C) : in welcher Relation stehen B und C zueinander ?

F6 (?,B,?) : gibt es eine Relation, in der irgendeine Entity zu B steht ?

beantwortet werden können. Bei einer Relation A mit p=2 schließlich werden alle 3 zyklischen Permutationen ABC, BCA und CAB der Triaden gespeichert, so daß auch noch die restlichen beiden Fragetypen

F2 (A,?,C) : zu welcher Entity (zu welchen Entities) steht C in der Relation A ?

F5 (?,?,C) : gibt es eine Relation, in der C zu irgendeiner Entity steht ?

beantwortet werden können.

Der Permutationstyp und damit die Art der erlaubten Fragen sind für alle Triaden derselben Relation gleich, können aber bei verschiedenen Relationen verschieden angegeben werden.

1.3.3 Frageklassen

Die im letzten Abschnitt genannten 7 Fragetypen lassen sich in 3 Klassen einteilen :

Klasse 0 : F0 : gilt ein bestimmter, angegebener Relationszusammenhang ?

Klasse 1 : F1,F2,F3: welche Entity (Entities) steht (stehen) zu zwei vorgegebenen Entities in einem Relationszusammenhang ?

Klasse 2 : F4,F5,F6: gibt es Entities, die zu einer vorgegebenen Entity in einem Relationszusammenhang stehen ?

DATAS gestattet die Beantwortung aller drei Frageklassen. Bei Fragen der Klasse 0 erhält man als Antwort die Feststellung, ob der genannte Relationszusammenhang besteht. Die Art der Antwort auf

eine Frage der Klasse 1 hängt vom Fragemodus und davon, ob eine oder mehrere Entities gefunden wurden, ab. Fragen dieser Klasse werden im nächsten Abschnitt besprochen. Bei Fragen der Klasse 2 erhält man schließlich Listen aller Entities in den Ringstrukturen, die die angegebene Entity enthalten, so daß man durch weitere Fragen der Klasse 1 detaillierte Informationen einholen kann.

1.3.4 Fragemodus

Fragen an DATAS haben einen sogenannten Fragemodus m, der die Form des Retrieval jedoch nur bei Fragen der Klasse 1 beeinflußt.

Bei m=0 hängt hier die Form der Antwort auf die Frage von der Anzahl der Entities ab, die den gefragten Relationszusammenhang erfüllen. Ist diese Anzahl 1, so wird der Name dieser einen gefundenen Entity als Antwort übergeben, andernfalls erhält der Benutzer eine Liste der assoziierten Entities und, falls gewünscht, Information über die sie enthaltenden Ringstrukturen. Mit Hilfe dieser Information kann der Benutzer Schritt für Schritt jede der Entities, die mit den beiden in seiner Frage angegebenen Entities assoziiert sind, laden und untersuchen oder manipulieren, oder er kann sich eine Liste aller dieser Entities für einen Gesamtüberblick ausgeben lassen.

Bei m=2 wird automatisch eine Multi-Value-Entity, deren Name zur Kennzeichnung mit einem Stern beginnt, erzeugt, und diese enthält als Elemente die gefundenen Entities.

Fragemodus m=1 ist eine Mischform der beiden anderen Modi; wenn genau eine Entity gefunden wurde, wird ihr Name ausgegeben (entspricht m=0); wurden mehrere Entities gefunden, so wird, wie bei m=2, eine Multi-Value-Entity erzeugt.

1.4 Facts

Facts repräsentieren n-tupel von Entities; dabei ist die Komponentenzahl n beliebig. Ein Fact f wird dargestellt durch eine Relation-Entity mit dem Namen f, die Element einer Multi-Value-Entity *F*, der Menge aller Facts, ist. f wird mit seinen Komponenten durch eine Menge von Triaden des Permutationstyps 2 assoziiert; diese Triaden enthalten als weitere Komponenten Single-Value-Entities, deren Namen jeweils gleich der Komponenten-Nummer sind, so daß ein Fact

$$f = (f_1,\ldots,f_n)$$

durch eine Menge von Triaden

$$\{(f,1,f_1),\ldots,(f,n,f_n)\}$$

repräsentiert wird. Der Speichermodus der Triaden ist dabei 0 oder 1, je nachdem, ob die betreffende Komponente des Facts durch ID oder NA zu repräsentieren ist; die beiden ersten Komponenten werden generell durch ihr ID repräsentiert.

Retrieval bei Facts kann mit Hilfe der üblichen Triadenoperationen ausgeführt werden; eigene Operationen für Facts sind nur für Abspeichern und Löschen notwendig.

1.5 Speicherstrukturen

1.5.1 Namenspeicher

Der erste Teil jeder Namenspeicher-Seite wird über einen Hashalgorithmus adressiert, der eine Adresse aus den ersten 10 Zeichen des abzuspeichernden oder zu suchenden Namens berechnet. Dieser Algorithmus kann leicht, ohne daß weitere Änderungen am System nötig wären, gegen einen anderen ausgetauscht werden, um für Spezialanwendungen von DATAS eine Anpassung an dieser Stelle zu ermöglichen. Jede Zelle dieses Hash-Bereiches besteht aus 8 Maschinenworten; die ersten 5 enthalten die Zeichen des dort abgespeicherten Namens, die beiden nächsten das ID der Entity, die zu die-

sem Namen gehört. Das letzte Wort ist 0 oder enthält die Nummer der ersten Zelle einer Conflict-Liste, die zu dieser Zelle im Hash-Bereich gehört.

Der Rest der Seite enthält diese Conflict-Listen, während die leeren Zellen dieses Conflict-Bereiches eine sogenannte vacant-Liste bilden. Wenn die vacant-Liste leer ist und eine Conflict-Situation auftritt, so wird die Liste auf der letzten Seite des Namenspeichers fortgesetzt. Diese Seite wird vom System als gemeinsamer Overflow-Pool für alle Namenspeicher-Seiten eingerichtet, wenn der Namenspeicher mehr als 3 Seiten umfaßt. Ist auch die Overflow-Seite voll oder hat der Namenspeicher höchstens drei Seiten, so ist die Conflict-Behandlung bei leerer vacant-Liste nicht möglich und eine Fehlermeldung wird ausgegeben. In diesem Fall kann der Benutzer den Namenspeicher nachträglich durch zusätzliche Overflow-Seiten erweitern, muß dabei aber in Kauf nehmen, daß auf diesen Seiten das Retrieval geringfügig länger dauert.

Die Einteilung der Namenspeicher-Seiten in Hash- und Conflict-Gebiet ist willkürlich und darf für die einzelnen Seiten verschieden gewählt werden. Der Benutzer kann sie jedoch auch bei der Initialisierung des Systems global wählen, indem er die Anzahl n_c der Zellen des Hash-Gebietes für eine Seite angibt. Es ist empfehlenswert, für n_c eine Primzahl zwischen 100 und 200 zu wählen; $n_c>254$ ist nicht zulässig.

Die Struktur der einzelnen Namenspeicher-Seiten wurde gegenüber der ursprünglichen Version von DATAS nicht wesentlich geändert, so daß für weitergehende Information auf /1/ verwiesen werden kann.

1.5.2 Triadenspeicher

Die Seiten des Triadenspeichers werden jeweils in drei Bereiche aufgeteilt:

ein freier Speicher F1
ein Hash-adressierter Speicher HAS
ein freier Speicher F2

Verwendet man die Bezeichnungen der Permutation ABC einer Triade ABC (der sogenannten α-Permutation), so läßt sich sagen, daß F1 die A-Zellen in Form eines occupied-Ringes enthält, während die leeren Zellen in F1 wieder in Form eines vacant-Ringes organisiert sind. Ein weiterer Hashalgorithmus, der ebenfalls leicht ausgetauscht werden kann, berechnet eine Adresse in HAS aus A- und B-ID bzw. -NA. Ist die Zelle mit dieser Adresse leer, so wird hier die B-Zelle gespeichert. Andernfalls ist diese Zelle der Kopf eines Conflict-Rings, der in F2 liegt und alle die B-Zellen enthält, die zu dieser Adresse in HAS gehören. Davon abgesehen, bilden alle die B-Zellen, die zu einer bestimmten A-Zelle gehören, einen B-Ring mit der A-Zelle als Kopf. Gehören zu einer bestimmten A-B-Assoziation mehr als eine C-Zelle, so bilden diese einen in F2 liegenden C-Ring mit der B-Zelle als Kopf. Andernfalls enthält die B-Zelle das C-ID bzw. -NA und wird als Terminal bezeichnet.

Die Einteilung der Seiten in die 3 Bereiche F1, HAS und F2 ist wieder willkürlich und darf für die einzelnen Seiten verschieden gewählt werden. Der Benutzer kann sie jedoch auch bei der Initialisierung oder bei einer eventuellen Erweiterung des Systems global wählen, indem er die Anzahlen n_r und n_h der Zellen von F1 bzw. HAS auf einer Seite angibt. Es ist empfehlenswert, $n_r < 100$ und n_h als Primzahl zwischen 100 und 200 zu wählen; $2n_h + n_r \geq 680$ ist nicht zulässig.

Die hier beschriebene, ziemlich komplizierte Struktur wurde aus den Speicherstrukturen von LEAP und SAM entwickelt. Sie ermöglicht sehr schnelles Abspeichern, Wiederfinden und Löschen von Triaden. Garbage Collection ist etwas langsamer (< 100 ms pro Seite), braucht aber nur nach dem Löschen größerer Mengen von Triaden durchgeführt zu werden und kann auch im off-line-Modus laufen. Die Seitenstruktur wurde gegenüber der ursprünglichen Version von DATAS nicht wesentlich geändert, so daß die in /1/ gegebene genauere Beschreibung noch weitgehend zutrifft.

1.5.3 Datenspeicher

Die ersten 255 Worte jeder Datenspeicher-Seite enthalten ein doppeltes Ring-System, das aus einem vacant-Ring und einem occupied-Ring besteht, deren Zellen über die IDs adressiert werden. Jede Zelle des occupied-Ringes enthält einen Zeiger auf das restliche Gebiet dieser Seite, das die Daten und Namen der auf dieser Seite abgespeicherten Entities in kontinuierlicher Form abgespeichert enthält. Dieser zusammenhängende Datenbereich wird nur durch einzelne Worte zwischen den Entities unterbrochen; diese Worte enthalten Zeiger auf den Anfang der nächsten Entity in diesem Bereich.

Diese Speicherstruktur ermöglicht nachträgliche Erweiterung oder Kürzung der zu einer Entity gehörenden Daten sowie das Abspeichern und Löschen ganzer Entities, ohne daß die Struktur reorganisiert werden müßte und ohne daß im Datenspeicher eine Garbage-Collection nötig wäre.

Gegenüber der ursprünglichen Version von DATAS wurde das Ring-System um eine Zelle verkürzt, so daß der kontinuierliche Speicher schon im Wort mit der Nummer 256 auf den Seiten des Datenspeichers beginnt (statt bei 258). Davon abgesehen, ist die Speicherstruktur und der Aufbau der Entities unverändert geblieben, so daß für genauere und weitergehende Informationen wieder auf /1/ verwiesen werden kann.

Als Daten, die im kontinuierlichen Bereich abgespeichert werden dürfen, sind alle auf der betreffenden Maschine erlaubten Datentypen zulässig; im Dialogbetrieb mit Eingabe über den Assoziativ-Assembler ist jedoch zur Zeit nur eine Eingabe von Zeichenfolgen möglich, und alle auszugebenden Daten werden als Zeichenfolge interpretiert. Diese Einschränkung entfällt bei der Arbeit mit DATAS durch vom Benutzer geschriebene Programme vollständig; eine entsprechende Erweiterung des Assemblers ist geplant.

1.5.4 Irrelevante Information

Werden Entities gelöscht, deren IDs und/oder NAs als Verweise in Multi-Value-Entities und/oder Triaden stehen, ohne daß auf die Existenz dieser Verweise Rücksicht genommen wird, so sind die betreffenden Verweise ungültig und vom Benutzer aus nicht mehr erreichbar ("Leichen"). Wird auf einen solchen Verweis in einer Multi-Value-Entity durch den Befehl, Elemente dieser Menge aufzulisten, zugegriffen, so wird der Verweis vom System als ungültig erkannt und automatisch gelöscht. Ähnlich werden beim Retrieval gefundene ungültige Triaden automatisch gelöscht; dies geschieht allerdings nur bei Fragen der Klasse 1, bei denen entweder der Fragemodus 1 oder 2 ist oder bei denen genau eine Assoziation gefunden wurde. Bei Fragen, die nur Information über das Ring-System ausgeben, erfolgt dagegen keine Überprüfung der Gültigkeit der Triaden.
Wird dagegen das gelöschte ID bzw. NA durch Abspeichern einer neuen Entity oder eines neuen Namens neu belegt, so sind die Verweise wieder gültig, beziehen sich nun aber auf das neue Objekt. Aus diesem Grunde ist vor leichtfertigem Löschen von Entities und Namen, auf die noch Verweise existieren, zu warnen; ehe diese Objekte gelöscht werden dürfen, sind die Verweise auf sie zu löschen.

1.5.5 Kapazität

Jede Seite des Namenspeichers kann maximal 255 Namen enthalten, jede Seite des Triadenspeichers bis zu etwa 500 Triaden (abhängig von Seiteneinteilung und assoziativer Struktur der abgespeicherten Zusammenhänge), und jede Seite des Datenspeichers kann bis zu 127 Entities mit zusammen bis zu 1786 Maschinenworten als Daten enthalten. Diese Zahlen können ohne größere Änderungen am System nicht erhöht werden.

Die Gesamtzahl der Seiten, die vom System bearbeitet werden können, also für Namen-, Triaden- und Datenspeicher zusammen, wurde auf 2039 Seiten festgesetzt. Durch Änderung weniger Befehle in der FORTRAN-Quelle kann dieser Wert erheblich vergrößert werden. Dabei sind zur Zeit als absolutes Maximum 16383 Seiten für den Da-

tenspeicher möglich; eine noch größere Erweiterung erfordert dagegen erhebliche Änderungen am System, vor allem im Triadenspeicher. Die Anzahl der Seiten, die für Namen- und Triadenspeicher erlaubt ist, ist prinzipiell nicht nach oben beschränkt.

Wurde das System mit einer bestimmten Seitenanzahl initialisiert und erweist es sich im Lauf der Arbeit als zu klein, so können ohne irgendwelche Änderungen in der Struktur weitere Seiten für Namen-, Triaden- und/oder Datenspeicher hinzugefügt werden; die nachträgliche Erweiterung des Namenspeichers ist wegen der längeren Suchzeit für dort abgespeicherte Namen jedoch nur als Notmaßnahme gedacht.

1.6 Beispiel einer Assoziation in DATAS

Gegeben sei eine Relation

A = {(B,C) | B ∈ "Fahrzeuge";C ∈ "Flächen";H(B,C) : "C ist eine Fläche von B"} .

Ihre Elemente seien durch folgende Terme spezifiziert :

A1 = {(B1,C1), (B2,C1), (B2,C2)}

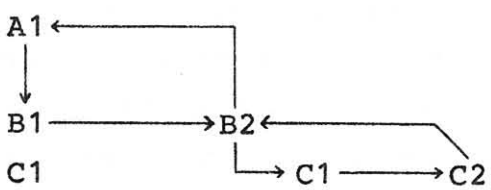
= {(Auto,Tür), (Bus,Tür), (Bus,Dach)}

Dabei können die Basismengen "Fahrzeuge" und "Flächen" durch die Menge aller Entities ersetzt werden, so daß auch Bildungen wie (A1,B1) als Elemente von A1 erlaubt sind.

Diese Relation wird im DATAS-System in der Form

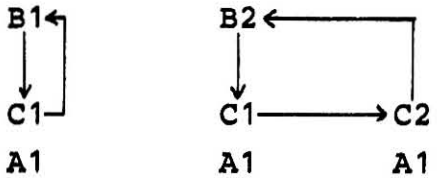
A1 = {(B1,C1), (B2,{C1,C2})}

durch Zusammenfassen gleicher erster Komponenten von Paaren gespeichert. Die im Triadenspeicher aufgebaute Struktur bietet folgendes Bild (beim Permutationstyp 0) :



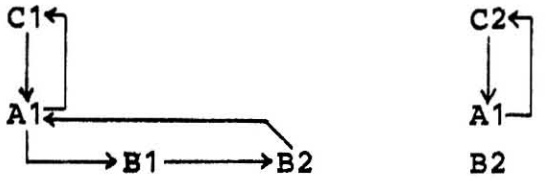
(α-Permutation)

Beim Permutationstyp 1 wird zusätzlich folgende Struktur abgespeichert :



(β-Permutation)

Schließlich wird beim Permutationstyp 2 als dritte Struktur die folgende abgespeichert :



(γ-Permutation)

Wenn man vom System nun Informationen über Fahrzeugflächen erhalten will, kann man die Frage (A1,?,?) stellen und erhält als Antwort eine Liste, die aus dem Elementen B1 (Auto) und B2(Bus) besteht, sowie auf Wunsch - für Sonderbearbeitungen - die Adresse des Anfanges des zu A1 gehörenden B-Ringes. Interessiert man sich nun für das Auto, so erhält man auf die Frage (A1,B1,?) die Antwort C1 (Tür). Will man dagegen Informationen über den Bus, so erhält man auf die Frage (A1,B2,?) die Liste der mit A1-B2 assoziierten Entities C1 (Tür) und C2 (Dach), sowie wieder auf Wunsch, die Anfangsadresse des diese Entities referierenden C-Ringes.

Man kann jedoch die Frage (A1,B2,?) mit der Nebenbedingung stellen, daß das System eine Menge erzeugen soll, die die Antworten als Elemente enthält, und in diesem Fall gibt das System als Antwort den Namen dieser Menge aus, der zum Beispiel *C sein kann. Listet man die Elemente von *C auf, so erhält man wieder C1 (Tür) und C2 (Dach). *C kann dabei als der Sammelbegriff "Flächen eines Bus" aufgefaßt werden.

Das hier genannte Beispiel wird zur Demonstration des DATAS-Systems im Abschnitt 3.4 aufgelistet.

2. Benutzung des Systems

2.1 Dialog- und Stapelbetrieb

2.1.1 Interpreter

Der DATAS-Interpreter ermöglicht Aufbau und Manipulation der DATAS-Struktur sowie Information-Retrieval in der Struktur unter Benutzung der im 3. Kapitel beschriebenen Assoziativ-Assemblersprache. Die Befehle dieser Sprache bestehen aus jeweils einem (im allgemeinen dreibuchstabigen) Mnemocode, dem, durch Komata voneinander abgetrennt, formatfrei weitere Parameter folgen können und/oder müssen. Der Interpreter besteht aus einem FØRTRAN-Hauptprogramm, das, zusammen mit einigen die Umcodierung der Eingabezeile bewirkenden Unterprogrammen, aus den Befehlen des Assoziativ-Assemblers die gewünschten Unterprogramm-Aufrufe erzeugt.

Der Interpreter kann im Dialogbetrieb, im Stapelbetrieb und im gemischten Betrieb laufen, d.h. die Befehle können einzeln oder in Gruppen (deren Länge nur durch die Bildschirmgröße begrenzt ist) von einem Terminal eingegeben werden; sie können aber auch als gesamtes Assembler-Programm von einer Datei oder einem Eingabemedium wie Karte, Lochstreifen oder Kassette eingelesen werden. Die Ausgabe-Informationen des Systems können ins Druckerprotokoll oder auf beliebige Dateien gegeben werden.

Steht der Interpreter als lauffähiges Objektprogramm dem Job zur Verfügung und sind sämtliche benötigten Datei-Bearbeitungen eröffnet, so kann der Interpreter gestartet werden. Für die Telefunken TR 440 steht außerdem die im Abschnitt 2.2.1 beschriebene Kommando-Prozedur DATAS zur Verfügung, die das Laden des Interpreters, der DATAS-Bibliothek und eventueller System- und Quelldateien übernimmt, alle benötigten Datei-Bearbeitungen eröffnet, den Interpreter startet und eventuell nötige Fehlerbehandlungen übernimmt.

Der Interpreter ist für anwenderspezifische Erweiterungen der Assemblersprache eingerichtet; wird ein dem Interpreter unbekannter

Befehl eingelesen, so erfolgt ein Sprung in eine Subroutine EXTSYS, die bei der hier beschriebenen Standardversion des Systems nur ein Dummy ist und eine Fehlermeldung ausgibt. Eine anwenderspezifische Erweiterung des Befehlsvorrats kann daher, ohne Änderung am Interpreter selbst, durch Austausch des Unterprogramms EXTSYS erfolgen.

2.1.2 Arbeit mit der DATAS-Bibliothek

Für spezielle Anwendungen des Systems können auch Programme geschrieben werden, die direkt die Subroutines der DATAS-Bibliothek aufrufen. Diese Anwenderprogramme können ebenfalls im Dialog- und/oder Stapelbetrieb laufen; sie unterliegen lediglich der Bedingung, daß sie den Anschluß von FØRTRAN IV-Subroutines gestatten. Da dem Benutzer hier keinerlei Beschränkungen in Bezug auf die von diesen Programmen durchzuführenden Aufgaben auferlegt werden sollten, erschien es sinnvoll, ihm auch die Eröffnung aller Datei-Bearbeitungen zu überlassen. Das Laden der DATAS-Bibliothek kann auf der TR 440 durch das Kommando BDATAS, das als Kommando-Prozedur zur Verfügung steht, übernommen werden. Dieses Kommando wird im Abschnitt 2.2.2 beschrieben.

Die Steuerung der DATAS-Operationen geschieht durch Unterprogrammaufrufe, die ihrerseits selbstverständlich von allen Steuerungsmöglichkeiten, die die aufrufende Sprache bietet, abhängig gemacht werden können. Der Datenverkehr mit den DATAS-Unterprogrammen geschieht auf zwei Weisen: durch Parameterübergabe einerseits und durch Zugriff auf globale Speichergebiete - CØMMØN-Blöcke - andererseits. Um die Geschwindigkeit des Systems hoch zu halten, wird dabei, wenn möglich, die zweite Form des Datenverkehrs benutzt, besonders wenn auf größere Datenmengen zugegriffen wird.

Daten, die als Parameter übergeben werden, sind hauptsächlich Namen und/oder Adressen (IDs, NAs) von Entities sowie INTEGER-Zahlen, deren Wert den Ablauf einer DATAS-Operation steuert bzw. Information über deren Ergebnis zurückmeldet. Als Adressierungsmöglichkeiten für Entities sind dabei, wie in der Assemblersprache, die Zeichenfolgen, die deren Namen darstellen, vorgesehen. Zusätzlich besteht jedoch bei den Operationen, bei denen

dies sinnvoll erschien, die Möglichkeit, existierende Entities, deren ID bekannt ist, direkt über dieses den Operationen zu übergeben. Auf diese Art und Weise kann die in vielen Sprachen ziemlich umständliche Manipulation von Zeichenfolgen durch die einfachere und schnellere Manipulation von INTEGER-Zahlen ersetzt werden. Zusätzlich wird noch der Zeitbedarf für das Suchen der Zuordnung Name-ID im Namenspeicher sowie für die Überprüfung, ob die angegebene Entity existiert, in diesem Falle eingespart.

Daten, die dem System übergeben werden sollen, sind in einem bestimmten CØMMØN-Block aufzubauen, und in demselben Block, der somit etwa der "User Working Area" des CODASYL-Konzepts entspricht, liefert das System beim Retrieval gefundene Daten zur Weiterverarbeitung durch den Benutzer ab. Dieser für die Datenübergabe vorgesehene Bereich wurde mit 1800 Worten gerade etwas größer als die maximale Datenkapazität einer Entity gewählt.

Eine im CØMMØN liegende Fehlervariable gibt durch ihren Wert 0 an, daß eine Operation korrekt ausgeführt wurde; bei Auftreten eines Fehlers wird sie dagegen auf einen von der Art des Fehlers abhängigen Wert gesetzt, um so den weiteren Programmablauf steuern zu können. Eine Liste der möglichen Fehlermeldungen mit den zugehörigen Werten der Fehlervariablen wird im Abschnitt 3.3 gegeben.

2.2 DATAS-Kommandos für die TR 440

2.2.1 Das Kommando DATAS

2.2.1.1 Parameter

Laden des Systems, Eröffnung der Dateibearbeitung, Starten des Interpreters — DATAS

DATEI = Angabe der Systemdatei
*Voreinstellung: -STD-

datei[-p] Systemdatei liegt mit auftragsspezifischem Benutzerkennzeichen (und eventuellem Paßwort p) in der LFD

bkz.datei[-p] Systemdatei liegt mit Benutzerkennzeichen bkz (und eventuellem Paßwort p) in der LFD; bkz muß ebenfalls im Parameter DATENBASIS angegeben werden.

l-datei[-p] Systemdatei soll auf dem angegebenen Träger (mit eventuellem Paßwort p) kreiert werden oder ist keine LFD-Datei; ist l*10, so ist im Parameter DNUMMER 10Ul anzugeben.

l-bkz.datei[-p] Systemdatei soll auf dem angegebenen Träger (mit eventuellem Paßwort p) in der Datenbasis bkz bzw. mit dem Benutzerkennzeichen bkz kreiert werden oder ist keine LFD-Datei; ist l*10, so ist im Parameter DNUMMER 10Ul anzugeben; bkz muß ebenfalls im Parameter DATENBASIS angegeben werden.

-STD- Systemdatei wird auf dem angegebenen Träger mit dem Namen NØNAME kreiert; ist der Träger LFD, so erhält die Datei das auftragsspezifische Benutzerkennzeichen

SATZZAHL = Anzahl der Sätze der Systemdatei
*Voreinstellung: M10

Gn Genau n Sätze
Un Ungefähr n Sätze
Mn Maximal n Sätze

TRAEGER Informationsträger der Systemdatei
*Voreinstellung: P

T Trommel
P Platte
LFD LFD (langfristige Datenhaltung)
t(kz)[a.p-gb] Trägertyp mit Kennzeichen kz

t: MB Magnetband auf Gerät B52
U52 Magnetband auf Gerät U52

B60 *) Magnetband auf Gerät MBG263/264
B60H H = hohe Schreibdichte
B60N N = niedrige Schreibdichte

*) Spezifikationswerte nur angeben, wenn Geräte physikalisch vorhanden

{W14[AZ] / W30[AZ]} Wechselplatte mit Vielfachzugriff, bei Angabe AZ im Alleinzugriff

a.p: Der a-te Dateiabschnitt der p-ten Datei einer Datenträgerreihe

gb: Grenzblock, bis zu dem ein Dateiabschnitt auf jedem Magnetband geschrieben werden soll

DNUMMER = Umbenennung logischer Gerätenummern

*Voreinstellung: -

\- keine Umbenennung

mUn Die Nummer m wird in Nummer n umbenannt. Mehrere Angaben sind durch Apostroph zu trennen.

DATEN = Abgeschlossenes Programm in der DATAS-Assemblersprache

*Voreinstellung: -

\- keine Daten

/f Fremdstring: DATAS-Assemblerprogramm

/f␣/ Diese Form nur, wenn noch weitere Spezifikationen folgen. Wird das DATAS-Programm im DATEN-Parameter übergeben, so ist im Parameter DNUMMER 8U5 anzugeben.

DATENBASIS = Kreation einer Datenbasis

*Voreinstellung: PØØL

name Name der zu kreierenden Datenbasis (max. 6 Zeichen) Wird in einem der Parameter DATEI und/oder INFØRMATIØN ein Benutzerkennzeichen bkz angegeben, so muß bkz ebenfalls im Parameter DATENBASIS angegeben werden.

INFØRMATIØN = Angabe von Ein- und/oder Ausgabedateien für den DATAS-Assembler und Zuordnung logischer Gerätenummern

*Voreinstellung: -

\- keine Zuordnung

l-datei[-p] Datei in der Standard-Datenbasis oder LFD-Datei mit auftragsspezifischem Benutzerkennzeichen (mit eventuellem Paßwort p)

l-bkz.datei[-p]Datei in der Datenbasis bkz oder LFD-Datei mit Benutzerkennzeichen bkz (mit eventuellem Paßwort p) bkz muß ebenfalls im Parameter DATENBASIS angegeben werden; außerdem sind im Parameter DNUMMER die entsprechenden logischen Gerätenummern umzudefinieren.
Mehrere Angaben sind durch Apostroph zu trennen.

Bei Benutzung der Voreinstellungen sind alle Parameter optional. Beispiele für die Form des DATAS-Kommandos sind die folgenden:

```
␣DATAS
␣DATAS,1O-PØØL.SYFILE-DATAS,U25,LFD,8U1,,,1-PØØL.INFILE
␣DATAS,DNU.=8U5,DATEN=/
␣DATAS,DATEI=1O-SCRATCH,SATZZ.=M5,T.=T
␣DATAS,1O-N11111.TEST,DATENBASIS=N11111
```

2.2.1.2 Wirkungsweise

Das Kommando DATAS benützt die Wahlschalter WS5,WS6 und WS7; sein Ablauf hängt vom logischen Wert der Wahlschalter WS6 und WS7 ab. Zuerst wird in jedem Falle die Programm-Bibliothek einschließlich des Interpreters geladen. Falls WS6 und WS7 gelöscht sind, wird eine Datei mit dem im Parameter DATEI angegebenen Namen in der LFD angemeldet, wenn sie dort existiert. Sie wird dann auf eine Scratch-Datei 'SCR' in der Standard-Datenbasis, deren Satzzahl anzugeben ist und gleich der der LFD-Datei sein muß, kopiert. Dann wird der Interpreter gestartet. Nach erfolgreicher Beendigung des DATAS-Programms oder -Dialogs wird die Datei SCR auf die LFD-Datei zurückkopiert; trat jedoch ein Fehler auf, der zum Abbruch des Interpreter-Laufs führte, so unterbleibt das Zurückkopieren, um auf der LFD-Datei den vor Beginn des Programms noch vorhandenen korrekten Systemzustand zu erhalten. Nach erfolgreichem Programm und erfolgtem Zurückkopieren von SCR auf die LFD-Datei wird SCR gelöscht. Die hier beschriebene Strategie ermöglicht vor allem die Aufrechterhaltung von Konsistenz und Integrität der Datenbank, wenn der Rechner während des Laufes eines Assembler-Programms, das schon Teile der Struktur geändert hat, diese Änderungen aber noch nicht korrekt und

vollständig abgeschlossen hat, zusammenbricht. Nach einem solchen Rechnerzusammenbruch befindet sich die Datenbank genau in dem Zustand, in dem sie vor Beginn des Programms war. Da weiterhin die LFD-Datei während des eigentlichen DATAS-Laufes abgemeldet ist, ist auch gleichzeitiges, die Struktur nicht veränderndes Information Retrieval mehrerer voneinander unabhängiger Benutzer möglich. Gleichzeitige Veränderungen der Struktur durch mehrere Benutzer sind zwar möglich, aber nicht sinnvoll, da der daraus resultierende Systemzustand nur die Veränderungen durch den Benutzer berücksichtigt, dessen Programm als letztes fertig wurde. Nach erfolgreichem DATAS-Lauf in diesem Kopiermodus werden alle benötigten Dateien wieder abgemeldet, die Wahlschalter WS5, WS6 und WS7 und die im Parameter DATENBASIS angegebene Datenbasis gelöscht.

Wurde im Parameter DATEI ein Name angegeben, zu dem keine LFD-Datei existiert, so wird eine Datei mit dem angegebenen Namen auf dem angegebenen Träger kreiert und Wahlschalter WS7 gesetzt. Wurde im Parameter DATEI überhaupt nichts angegeben (Voreinstellung: -STD-), so wird die Systemdatei mit dem Namen 'NØNAME' erzeugt und Wahlschalter WS6 gesetzt. In beiden Fällen erfolgt der DATAS-Lauf ohne Erstellung einer Zwischenkopie direkt auf der Datenbank, da diese ja erst gerade erzeugt wurde und somit erst während des Programmlaufs Informationen erhält, so daß hier ein Retten der Datenbank bei einem Systemzusammenbruch nicht nötig ist. Nach erfolgreichem Programmlauf werden alle beteiligten LFD-Dateien abgemeldet, und die angegebene Datenbasis wird gelöscht; dagegen bleibt Wahlschalter WS7 bzw. WS6 gesetzt.

Falls bei Eingabe des Kommandos DATAS Wahlschalter WS7 gesetzt war, wird der DATAS-Lauf sofort mit der im Parameter DATEI angegebenen Systemdatei gestartet; diese muß in der Standard-Datenbasis bzw. in einer im Parameter DATENBASIS angegebenen Datenbasis liegen. Falls WS6 gesetzt war, wird der DATAS-Lauf sofort mit der Datei '&STDDB.NØNAME' gestartet. In beiden Fällen unterbleibt das Anlegen einer Zwischenkopie. Nach erfolgreichem Programmablauf werden wieder alle beteiligten LFD-Dateien abgemeldet, und die angegebene Datenbasis wird gelöscht; Wahlschalter WS7 bzw. WS6 bleibt gesetzt.

Das Kommando DATAS schaltet in jedem Falle vor Beginn des eigentlichen DATAS-Laufes das Druckerprotokoll im DC2 ein.

2.2.2 Das Kommando BDATAS

2.2.2.1 Parameter

Keine

2.2.2.2 Wirkungsweise

Das Kommando BDATAS benützt den Wahlschalter WS8; sein Ablauf hängt vom logischen Wert dieses Wahlschalters ab.

Wenn der Wahlschalter WS8 gelöscht ist, wird die DATAS-Bibliothek einschließlich des Interpreters geladen und anschließend WS8 gesetzt. Ist WS8 schon gesetzt, so wirkt das Kommando BDATAS wie eine leere Eingabe. Die weiteren Operationen, die nötig sind, um den Interpreter oder ein vom Anwender geschriebenes DATAS-Programm zu starten, einschließlich der Datei-Dienste, müssen vom Anwender selbst übernommen werden. Für den Anschluß von DATAS-Programmen an andere Programme ist dabei jeweils das aktuelle Listing der DATAS-Quelle zu benützen.

2.3 Technische Voraussetzung für den Einsatz von DATAS

2.3.1 Speicherplatz

Hier kann kein genereller Wert angegeben werden, da je nach der verwendeten Maschine und je nach der von der Maschine zur Verfügung gestellten Software erhebliche Unterschiede im Umfang des erzeugten Objektprogrammes bestehen können. Als Werte für die Telefunken TR 440 sind hier zu nennen: 28k Kernspeicher für das lauffähige Interpreter-Programm, 32k für ein Gespräch, das den Interpreter benutzt (bei 48 Bit Wortlänge). Nimmt man eine Erhöhung der Rechenzeiten um bis zu 50 % in Kauf, so läßt sich der Speicherbedarf für ein DATAS-Gespräch von 32k auf 26k drücken, indem eine (auf der TR 440 ebenfalls vorhandene) segmentierte Form des Interpreters verwendet wird. Durch noch weitergehende Segmentierung läßt sich, bei entsprechender Erhöhung des Rechen-

zeitbedarfs, der Speicherplatzbedarf des Interpreter-Programms bis auf 21...23k verringern.

Über den von Anwenderprogrammen benötigten Speicherplatz läßt sich vernünftigerweise keine Aussage machen, da dieser völlig von der Art des Anwenderprogrammes abhängt.

2.3.2 Wortlänge

Das System setzt eine minimale Wortlänge von 16 Bit voraus, da alle Daten in Form von positiven INTEGER-Zahlen von maximal 15 Bit Länge abgespeichert werden. Auf eine Verwendung des 16-ten Bits, des Vorzeichen-Bits also, wurde verzichtet, um das System nicht auf Maschinen einer bestimmten Wortlänge und einer bestimmten Darstellung negativer Zahlen festzulegen. Das System arbeitet in Bezug auf den Speicherplatz somit am rationellsten bei 16-Bit-Maschinen wie zum Beispiel der Varian V 73. Durch das Setzen der Single-Option beim Aufruf des FØRTRAN-Compilers kann bei den meisten Maschinen mit kurzer Wortlänge (z.B. 24 Bit bei der Control Data CD 3300) die Länge von INTEGER-Zahlen auf ein Wort beschränkt werden, bei Maschinen großer Wortlänge (z. B. IBM) auf ein Halbwort, so daß der verschenkte Speicherplatz relativ klein gehalten werden kann. Bei der vorliegenden Implementierung für die Telefunken TR 440 wird dies durch eine generelle Deklarierung fast aller Variablen und aller Felder als INTEGER*2, also als aus Halbworten bestehend, erreicht. Da wegen eines Fehlers in der FØRTRAN-I/Ø nur INTEGER*4-Felder, also Ganzwort-Bereiche, korrekt auf das externe Speichermedium übertragen werden, mußten die dort abzuspeichernden Felder über EQUIVALENCE-Anweisungen zusätzlich als INTEGER*4-Felder der halben Länge deklariert werden. Bei Verwendung eines korrekten Compilers oder bei Verzicht auf die Halbwort-Abspeicherung kann diese doppelte Deklaration natürlich entfallen.

2.3.3 Externe Geräte und Dateien, Ein- und Ausgabe

Das System kennt folgende logische Gerätenummern:

4 : Parallele Ausgabe von Meldungen und Daten auf Terminal und Schnelldrucker
6 : Ausgabe von Dumps auf dem Schnelldrucker
8 : Eingabe von Befehlen und Daten vom Terminal aus
9 : Ausgabe von Aufforderungen an den Benutzer auf das Terminal allein
10: Datei auf einem externen Speichermedium, die die Seitentabelle des Systems sowie die Seiten von Namen-, Triaden- und Datenspeicher enthält

Diese Gerätenummern entsprechen - mit Ausnahme der 10 - den an der Telefunken TR 440 üblichen. Bei anderen Maschinen müssen gegebenenfalls diese Gerätenummern umdefiniert werden, sofern nicht vorgezogen wird, die Nummern in den - relativ wenigen - READ- und WRITE-Anweisungen abzuändern.

Die READ-Anweisungen enthalten ERR- und END-Parameter, um Eingaben im falschen Format abzufangen und die Ausgabe von Meldungen an den Benutzer zu steuern. Bei Maschinen, die diese Möglichkeit nicht vorsehen, sind die Parameter aus den Eingabe-Anweisungen zu entfernen. Gleiches gilt für die WRITE-Anweisungen, die den Seitentransport übernehmen und durch ERR-Parameter abgesichert wurden, um die versehentliche Definition eines DATAS-System-Files, der größer als der zur Verfügung stehende Speicherplatz ist, zu verhindern.

Diese unter der logischen Gerätenummer 10 anzusprechende Systemdatei muß direkten Zugriff auf einen bestimmten Satz der Datei (random access) und die Abspeicherung von 2048 16 Bit-INTEGER-Zahlen in einem Satz gestatten. Bei der Telefunken TR 440 entspricht dies den folgenden Charakteristiken:

Typ: RAN : Random-Datei mit Zugriff über Satznummer
Zugriffskoordinierung: P: Privatdatei
G: Gemeinschaftsdatei

Satzzahl: Mn oder Un: maximal oder ungefähr n Sätze
Satzbau: G1024W (genau 1024 Maschinenworte) bei der verwendeten Halbwort-Darstellung von INTEGER-Zahlen

Die Satzzahl muß mindestens gleich der Anzahl der Seiten für Namen-, Triaden- und Datenspeicher zusammen +1 sein. (Bei einer Erweiterung des Systems auf mehr als 2039 Seiten wird vom System zusätzlich für je 2048 weitere Seiten die Datei um einen Satz erweitert, der die Werte dieser Seiten für die Seitentabelle enthält; diese Sätze werden an den Schluß der Datei angehängt.) Der erste Satz der Datei enthält die Seitentabelle, solange diese nicht mehr als 2039 Seiten beschreibt, sonst deren ersten Teil. Die folgenden Sätze enthalten die einzelnen Seiten von Namen-, Triaden- und Datenspeicher.

Die Sätze der Datei werden durch READ- und WRITE-Anweisungen mit direktem (Random-)Zugriff gelesen und, falls sie im Kernspeicher verändert wurden, zurückgeschrieben. Bei Maschinen, deren FØRTRAN-Compiler keinen Direct-Access-Parameter in I/Ø-Anweisungen vorsieht, müssen Positionierung und Ein- und Ausgabe-Vorgang getrennt vorgenommen werden.

2.3.4 Externbezüge

Das System setzt das Vorhandensein einer INTEGER-Funktion, die zwei INTEGER-Zahlen bitweise durch Exklusiv-Oder verknüpft, in der FØRTRAN-Programm-Bibliothek des Rechners voraus. Diese Funktion wird bei Telefunken durch LØGAUT bezeichnet; da der Funktionsaufruf innerhalb des ganzen Systems nur in drei Anweisungen erfolgt, erscheint es zumutbar, ihn geeignet abzuändern, wenn die Funktion auf anderen Maschinen anders bezeichnet ist (etwa nach ISA-Norm als IEØR).

Die Dump-Routine ruft eine Subroutine NØKØPF aus der Telefunken-Bibliothek auf. Diese Subroutine schaltet die bei Telefunken übliche Überschriftzeile des Ablaufprotokolls aus, um beim Dump Platz und Papier zu sparen. Steht die Routine NØKØPF nicht zur Verfügung, so kann die Anweisung, die sie aufruft, ohne weiteres

gestrichen werden.

Die Fehler-Routine ERRØR des DATAS-Systems ist in der Lage, durch den Aufruf CALL ABØRT einer nicht vorhandenen Subroutine ABØRT einen Fehlerstop der DATAS-Programme zu erzwingen; dieser offene Externbezug ist bei der Erstellung lauffähiger DATAS-Programme zu beachten. Sind auf einer Maschine offene Externbezüge nicht erlaubt, so ist die Anweisung CALL ABØRT durch STØP oder eine Anweisung, die einen ebenfalls zum Abbruch des Programms führenden Fehler erzeugt, zu ersetzen.

2.4 Portabilität der Implementierung

2.4.1 Anforderungen an den Rechner und den Compiler

Da DATAS in FØRTRAN IVH ohne besondere Benutzung maschinenspezifischer Besonderheiten implementiert wurde, ist eine weitgehende Portabilität des Systems gewährleistet. Die über FØRTRAN IV hinausgehenden Anforderungen an den Compiler und die Maschinen-Software wurden schon in den letzten Abschnitten kurz angedeutet:

1. READ- und WRITE-Anweisungen enthalten zum Teil ERR- und/oder END-Parameter.
2. Einige READ- und WRITE-Anweisungen enthalten einen Direct-Access-Parameter.
3. Einige DATA-Anweisungen besetzen ganze INTEGER-Felder auf einmal und nicht durch Aufzählung der einzelnen Elemente.
4. Als Fehlerstop wird der Aufruf eines nicht vorhandenen Unterprogramms benutzt.
5. Deklarationen von Variablen und Feldern benutzen die Typen INTEGER*2 (Halbwort) und INTEGER*4 (Ganzwort); diese sind z.T. durch EQUIVALENCE-Anweisungen miteinander verknüpft.

Die an den Rechner gestellten Anforderungen sind recht schwach:

1. INTEGER-Zahlen müssen als Festkommazahlen zur Basis 2 dargestellt werden.
2. Die Wortlänge von INTEGER-Zahlen muß mindestens 16 Bit betragen, oder es muß möglich sein, INTEGER-Zahlen abzuspeichern, die sich über mehrere Worte erstrecken.
3. Die Maschine sollte wenigstens 30k Worte Kernspeicher bereit-

stellen können oder die Bildung von Overlays ermöglichen.
4. Es sollte ein externes Großraum-Speichermedium (Massenkernspeicher, Platte, Trommel, Band oder Kassette) vorhanden sein.

Diese Anforderungen werden mit Ausnahme extremer Kleinrechner von fast jedem heutigen Rechner erfüllt.

2.4.2 Notwendige Änderungen beim Übergang auf andere Rechner

Aus dem bisher Gesagten folgt, daß beim Übergang auf andere Rechner eventuell folgende Änderungen zum Teil oder alle nötig werden:

1. Entfernen der Anweisung CALL NØKØPF
2. Ersetzen der Anweisung CALL ABØRT durch STØP oder eine äquivalente Anweisung
3. Umdefinition logischer Gerätenummern oder deren Ersatz in I/Ø-Anweisungen
4. Entfernung von END- und ERR-Parametern in I/Ø-Anweisungen; eventuell auch deren Ersatz durch geeignete Unterprogramme auf Maschinenebene
5. Ersetzen der Typ-Deklarationen INTEGER*2 und INTEGER*4 durch INTEGER; Korrektur der Feldgrenzen der durch EQUIVALENCE assoziierten Felder
6. Entfernen der Anweisung IMPLICIT INTEGER*2 (I-N,P); explizite INTEGER-Deklaration des Feldes PAGTAB.
7. Ersetzen des Direct-Access-Parameters in einigen I/Ø-Anweisungen durch eine Positionierungs-Routine auf Maschinenebene.
8. Umbenennung der FUNCTIØN LØGAUT; eventuell Schreiben eines solchen Unterprogramms auf Maschinenebene

Da die zu ändernden Anweisungen in keinem Fall sehr zahlreich sind und da alle Änderungen auf genau lokalisierbare Anweisungen oder Gruppen von Anweisungen beschränkt bleiben, erscheint eine gute Portabilität des Systems gewährleistet.

3. Syntax und Befehlsvorrat des Assoziativ-Assemblers

3.1 Form der Befehle

Der Dialog mit dem DATAS-Assoziativ-Assembler wird durch Starten des Hauptprogramms DIALØG eröffnet; dieses gibt als erstes die Meldung

========= BITTE BUFFERLAENGE EINGEBEN

aus. Die Bufferlänge, d.h. die Anzahl der maximal pro Zeile zu interpretierenden Zeichen, wobei auch Zwischenräume (blanks) zählen, ist als aus zwei Ziffern bestehende INTEGER-Zahl einzugeben. Falls dabei ein Fehler gemacht wird, wo wird die Meldung (maximal 10mal) wiederholt und eine neue Eingabe erwartet. Wurde als Bufferlänge eine Zahl ≤ 30 oder >80 angegeben, so wird der Dialog mit einer Fehlermeldung abgebrochen.

Im folgenden werden Anweisungen mit der Meldung

========= BITTE ANWEISUNGEN EINGEBEN

angefordert, während Datenzeilen, die die Fortsetzung von Anweisungszeilen darstellen, je nach der vorangegangenen Anweisung durch

========= BITTE NÄCHSTEN DATENBLØCK EINGEBEN

bzw.

========= BITTE LETZTEN DATENBLØCK EINGEBEN
========= MAX.LAENGE = ...

oder

========= BITTE WEITERE KØMPØNENTEN EINGEBEN

bzw.

========= BITTE WEITERE ELEMENTE EINGEBEN

angefordert werden können. Anweisungen, die solche Fortsetzungszeilen erlauben, sind in der folgenden Befehlsliste durch ein vor den Abschnitt geseztes + gekennzeichnet.

Die ersten drei Zeichen einer Anweisungszeile (dabei zählen auch blanks!) werden als Befehlscode interpretiert; die hier erlaubten

Codes werden in den folgenden Abschnitten angegeben. Diese Liste erhebt keinen Anspruch auf dauernde Vollständigkeit, da je nach Bedarf neue Befehle in den Assembler aufgenommen werden.

Ein unerlaubter Befehlscode führt, ebenso wie die Eingabe eines Zeichens, das nicht zum erkennbaren Zeichenvorrat

␣(blank) 0...9 A...Z +-*/.(),='

gehört, oder auch wie die Eingabe von mehr als 23 Kommas in einer Zeile, ein Fehler bei einer Zahleneingabe oder das Fehlen eines obligaten Parameters dieses Befehls, zu einer Fehlermeldung und zum Ignorieren dieser Zeile. Falls Parameter angegeben werden, so folgen sie formatfrei auf den Befehlscode, jeweils durch Kommata abgetrennt. Nach dem letzten Parameter dürfen Kommata oder ein Apostroph folgen; dieses beendet die Umcodierung der Zeile an dieser Stelle. Bei einigen Befehlen (STS,STD,STE,STN,CHS und CHD) können als letzte Parameter Daten (beliebige Zeichenketten) eingegeben werden, die sich über mehrere Zeilen erstrecken dürfen. Kommata und die ersten drei Zeichen in der Zeile haben keine Funktion in Datenzeilen. Die Gesamtmenge der übertragbaren Daten beträgt maximal 1786 Zeichen in jedem dieser Befehle; sollen weniger Daten übertragen werden, so kann die Dateneingabe durch ein Apostroph beendet werden. Die restlichen Befehle, die sich über mehrere Zeilen erstrecken, sind die Befehle STC,STM,STF und DLM. Hier ist lediglich zu beachten, daß Zeilenwechsel wie ein Komma wirkt, also Parameter voneinander abtrennt.

Die Form der Befehle ist (mit Ausnahme der Befehle LIST und NØLIST, die die Ausgabe ein- und ausschalten) einheitlich:

XXX,par1,...,parI,...,[parJ],...,parM[']

Dabei ist XXX der aus drei Zeichen bestehende Befehlscode, der in den ersten drei Zeichen der Anweisungszeile stehen muß, par1 bis parM sind die Parameter des Befehls. Dabei sind (in diesem Beispiel) die Parameter par1,parI und parM obligatorisch; ihr Weglassen führt zu einer Fehlermeldung. parJ wurde durch eckige Klammern als optional gekennzeichnet, er kann ohne weiteres weggelassen werden. Sein Wert wird dann vom System automatisch vorgegeben. Diese Werte, die das System bei den einzelnen Parametern dann ein-

setzt, werden in der folgenden Befehlsliste jeweils angegeben; die entsprechenden Zeilen sind durch davorgesetzte Sternchen gekennzeichnet.

Achtung: Werden optionale Parameter weggelassen und dahinterstehende Parameter geschrieben, so müssen die den weggelassenen Parameter begrenzenden Kommata beide geschrieben werden!

Im Dialog können beliebig viele Befehle auf einmal gegeben werden; die Befehle müssen dann durch Zeilenwechsel voneinander abgetrennt werden. Bei der Eingabe über Lochkarten muß entsprechend jeder Befehl auf einer neuen Karte anfangen.

Wird nicht als erster Befehl der Befehl SYI (System-Initialisierung) gegeben, so versucht das System einen Restart mit den Informationen des System-Files. Enthält dieser noch überhaupt keine DATAS-Struktur, so erfolgt eine Fehlermeldung und Abbruch der Bearbeitung. Das Assembler-Programm wird durch den Befehl SYØ (System-out) beendet.

Als Namen von Entities können beliebige Zeichenfolgen angegeben werden; es werden jedoch nur die ersten 10 Zeichen übertragen. Dabei werden auch blanks als Zeichen gewertet; Namen, deren beide erste Zeichen blanks sind, sind nicht erlaubt.

Ein Vergleich der hier angegebenen Befehlsliste mit der in /1/ angegebenen des ursprünglichen Systems läßt den inzwischen erheblich größeren Sprachumfang des Assemblers erkennen; dabei konnte aber weitgehend die Kompatibilität zu allen früheren System-Versionen gewahrt werden.

3.2 Der Befehlsvorrat des Assoziativ-Assemblers

3.2.1 SY - SYSTEM-BEFEHLE

3.2.1.1 Initialisierung

SYI, $[n_n]$, $[n_t]$, $[n_d]$, $[n_r]$, $[n_h]$, $[n_c]$

*: n_n=1, n_t=1, n_d=1, n_r=30, n_h=101, n_c=199

Stellt eine leere Datenstruktur zur Verfügung.

n_n, n_t, n_d: Anzahl der Seiten für Namen-, Triaden- und Datenspeicher; bzgl. der erlaubten Werte siehe Abschnitt 1.5.5

n_r : Anzahl der Zellen im Speicherbereich F1 einer Seite des Triadenspeichers, entspricht der maximal auf der Seite abzuspeichernden Anzahl von Relationen; $n_r<100$ ist empfehlenswert.

n_h : Anzahl der Zellen in der Hash-Area HAS einer Seite des Triadenspeichers; $100<n_h<200$ und n_h Primzahl ist empfehlenswert; $2n_h+n_r \geq 680$ ist nicht zulässig.

n_c : Anzahl der Zellen in der Hash-Area einer Seite des Namenspeichers; $100<n_c<200$ und n_c Primzahl ist empfehlenswert; $n_c>254$ ist nicht zulässig.

3.2.1.2 Expandierung

SYE,[n_n],[n_t],[n_d],[n_r],[n_h],[n_c]

*: $n_n=0, n_t=0, n_d=0, n_r=30, n_h=101, n_c=47$

Erweitert die Datenstruktur um n_n Seiten im Namenspeicher n_t Seiten im Triadenspeicher und n_d Seiten im Datenspeicher. Eine Erweiterung des Namenspeichers bezieht sich nur auf den Overflow-Bereich (siehe Abschnitte 1.5.1 und 1.5.5) Bezüglich der Gesamtzahl der Seiten des Systems gilt auch hier das in Abschnitt 1.5.5 Gesagte; für die erlaubten Werte von n_r, n_h und n_c gilt dasselbe wie beim Befehl SYI.

3.2.1.3 System-Out

SYØ

Beendet das Assembler-Programm und überträgt die noch im Kernspeicher befindliche Information auf den System-File.

3.2.1.4 Garbage-Collection im Triadenspeicher

SYG

3.2.1.5 Page-Transport

SYP,n,nr

Die Seite mit der Nummer nr des Seitentyps n (1 für Namen-, 2 für Triaden- und 3 für Datenspeicher) wird in den Kernspeicher geladen. nr wird für jeden Seitentyp von 1 an gezählt.

3.2.1.6 System-State

SYS

Gibt den aktuellen Zustand des Systems (entsprechend der Seitentabelle) aus.

3.2.1.7 Dump

SYD,n

Die jeweils aktuelle (d.h. im Kernspeicher befindliche) Seite des Typs n (wie bei SYP) wird im Oktalformat auf dem Schnelldrucker ausgegeben.

3.2.1.8 List

SYL,n

Die auf der jeweils aktuellen Seite des Typs n (wie bei SYP und SYD) abgespeicherten Namen bzw. Entities (im Triadenspeicher nur die des A-Ringes) werden aufgelistet.

3.2.2 ST - STØRE-BEFEHLE

+ 3.2.2.1 Single-Value-Entity

STS,name[,daten][']
*: keine Daten, d.h. Kette der Länge 0

Es wird eine Single-Value-Entity mit dem Namen "name" und, falls angegeben, den Daten "daten" erzeugt. Die Daten können sich über mehrere Zeilen erstrecken und müssen, wenn sie weniger als 1786

Zeichen umfassen, mit Apostroph abgeschlossen sein.

+ 3.2.2.2 Entity-Element

STE,name,namecl[,daten]['] *: keine Daten

Es wird, falls noch nicht vorhanden, eine Single-Value-Entity mit dem Namen "name" und, falls angegeben, den Daten "daten" erzeugt. Ihr ID wird als Elementverweis in die Entity "namecl" eingetragen, die zu einer Multi-Value-Entity umdefiniert wird, falls die eine Single-Value-Entity ist.

+ 3.2.2.3 Name-Element

STN,name,namecl[,daten]['] *: keine Daten

Es wird, falls noch nicht vorhanden, eine Single-Value-Entity mit dem Namen "name" und, falls angegeben, den Daten "daten" erzeugt. Ihr NA wird als Elementverweis in die Entity "namecl" eingetragen, die zu einer Multi-Value-Entity umdefiniert wird, falls sie eine Single-Value-Entity ist.

+ 3.2.2.4 Data

STD,name[,i][,daten]['] *: i=-1, keine Daten

Falls die Entity "name" noch nicht existiert, wird sie als Single-Value-Entity mit den Daten "daten", falls angegeben, erzeugt. Andernfalls werden die angegebenen Daten hinter das i-te Wort des Datenbereichs von "name" eingeschoben, ohne schon vorhandene Daten zu löschen. Ist i negativ oder größer als die Anzahl der schon vorhandenen Datenwörter dieser Entity, so werden die neuen Daten an die alten hinten angehängt.

3.2.2.5 Relation-Entry

STR,name A,name B,name C[,code]
*:code = 2

Es werden, soweit noch nicht vorhanden, zwei Single-Value-Entities "name C" und "name B" und eine Relation-Entity "name A" ohne Daten erzeugt; ist "name A" schon als Single-Value-Entity vorhanden, so wird diese zu einer Relation-Entity umdefiniert. Zu diesen Entities wird nach den durch die Zahl "code" bestimmten Regeln eine Triade gebildet. "code" ist eine zweistellige INTEGER-Zahl, deren Stellen folgende Bedeutung haben :

code-Einerstelle: Permutationscode p der Relation; wird nur ausgewertet, wenn die Entity "name A" neu erzeugt oder umdefiniert wurde:

0: es wird nur die Triade ABC abgespeichert
1: es werden die Triaden ABC und BCA abgespeichert
2: es werden die Triaden ABC, BCA und CAB abgespeichert

(vgl. Abschnitt 1.3.2)

code-Zehnerstelle: Speichermodus s der Triade; wird die Zehnerstelle binär geschrieben, so geben die "1" an, welche der Entities A,B,C in der Triade durch ihr NA, die "0", welche durch ihr "ID" repräsentiert werden:

4,5,6,7: A wird durch NA statt ID repräsentiert
2,3,6,7: B wird durch NA statt ID repräsentiert
1,3,5,7: C wird durch NA statt ID repräsentiert

(vgl. Abschnitt 1.3.1)

Man hat also "code" = 10s + p

\+ 3.2.2.6 Many Relation-Entries

STM,name A, name B,[code],n ||,name C ||^{n}
*: code = 2

Der Befehl wirkt wie n-facher Aufruf von STR mit immer denselben Entities "name A" und "name B", aber verschiedenen Entities "name C"; es wird also ein C-Ring erzeugt. Die Eingabe der Werte "name C" darf sich über mehrere Zeilen erstrecken; weiterhin muß $0<n\leq 300$ gelten.

"code" hat dieselbe Bedeutung wie bei STR, jedoch wird die niedrigste Bitstelle von s nicht ausgewertet. C-Entities, die durch ihr NA repräsentiert werden sollen, müssen durch ein vor den Namen gesetztes N= gekennzeichnet werden; C-Entities, die durch ihr ID repräsentiert werden sollen, können durch ein vor den Namen gesetztes I= kenntlich gemacht werden.

+ 3.2.2.7 Class

STC,namecl,n||,name||^{n}

Der Befehl wirkt wie n-facher Aufruf von STE bzw. STN mit immer derselben Menge "namecl",aber verschiedenen Elementen "name". Die Eingabe der Elementnamen darf sich über mehrere Zeilen erstrecken; weiterhin muß $0<n\leq300$ gelten. Elemente, die durch ihr NA repräsentiert werden sollen, müssen durch ein vor den Namen gesetztes N= gekennzeichnet werden; Elemente, die durch ihr ID repräsentiert werden sollen, können durch ein vor den Namen gesetztes I= kenntlich gemacht werden.

+ 3.2.2.8 Fact

STF,name,n||,namec||^{n}

Es wird eine Entity mit dem Namen "name" als Element in die Menge *F* gespeichert, und die das Fact repräsentierende Triadenstruktur (vgl. Abschnitt 1.4) wird erzeugt. Die Eingabe der Komponentennamen darf sich über mehrere Zeilen erstrecken; weiterhin muß $0<n\leq99$ gelten. Komponenten, die durch ihr NA repräsentiert werden sollen, müssen durch ein vor den Namen gesetztes N= gekennzeichnet werden; Komponenten, die durch ihr ID repräsentiert werden sollen, können durch ein vor den Namen I= kenntlich gemacht werden.

3.2.3 DC - DECLARE-BEFEHLE

3.2.3.1 Synonym

DCS, name neu, name alt

Die Entity "name alt" wird auch unter "name neu" ansprechbar.

3.2.3.2 Class

DCC,namecl

Es wird eine Multi-Value-Entity mit dem Namen "namecl" ohne Elemente und ohne Daten als leere Menge erzeugt.

3.2.3.3 Relation

DCR,name[,code]
*: code = 2

Es wird eine Relation-Entity mit dem Namen "name" ohne Daten erzeugt. Die einstellige INTEGER-Zahl "code" wird als Permutationscode p eingetragen; sie hat dieselbe Bedeutung wie die code-Einerstelle der Befehle STR und STM.

3.2.4 DL - DELETE-BEFEHLE

3.2.4.1 Class

DLC,namecl

Die Multi-Value-Entity "namecl" und alle Elemente, die Single-Value-Entities sind, werden gelöscht. Elemente, die durch ihr ID repräsentiert sind, werden als Entities gelöscht; bei durch NA repräsentierten Elementen wird nur, soweit noch Synonyme existieren, der Name gelöscht.

3.2.4.2 Name

DLN,name[,code]
*:code = 0

Falls die Entity "name" mehrere Synonyme besitzt, wird der Name "name" gelöscht. Falls die INTEGER-Zahl "code" ≠ 0 ist und die Entity keine Synonyme besitzt, wird sie gelöscht.

3.2.4.3 Single-Value-Entity

DLS,name

Die Entity "name" wird gelöscht.

3.2.4.4 Relation-Entry

DLR,name A,name B,name C[,code]
*:code = 0

Die Triade ABC wird in allen abgespeicherten Permutationen gelöscht. Die zweistellige Zahl "code" spezifiziert den Löschmodus näher. Die code-Zehnerstelle gibt den Speichermodus s der Triade an; ihre Bedeutung ist die gleiche wie bei den Befehlen STR und STM. Die code-Einerstelle bestimmt, ob auch Namen oder Entities zu löschen sind:

code-Einerstelle:

0: nur die Triade wird gelöscht
1: außerdem wird "name C" gelöscht
2: außerdem werden "name C" und "name B" gelöscht
3: außerdem werden "name C", "name B" und "name A" gelöscht

Dabei gilt die Regel, daß Komponenten der Triade, die in dieser durch ihr ID repräsentiert werden, als Entities gelöscht werden; bei Komponenten, die durch ihr NA repräsentiert sind, werden nur, soweit Synonyme existieren, die Namen gelöscht.

+ 3.2.4.5 Many Relation-Entries

DLM,name A,name B,[code],n||,name C ||^{n}
*: code = 0

Der Befehl wirkt wie n-facher Aufruf von DLR mit immer denselben Entities "name A" und "name B", aber verschiedenen Entities "name C". Die Eingabe der Werte "name C" darf sich über mehrere Zeilen erstrecken; weiterhin muß $0<n\leq 300$ gelten. "code" hat dieselbe Bedeutung wie bei DLR, jedoch wird die niedrigste Bitstelle der code-Zehnerstelle, des Speichermodus s also, nicht ausgewertet. C-Entities, die durch ihr NA repräsentiert sind, müssen durch ein vor den Namen gesetztes N= gekennzeichnet werden; C-Entities, die durch ihr ID repräsentiert sind, können durch ein vor den Namen gesetztes I= kenntlich gemacht werden.

3.2.4.6 Data

DLD,name[||,n1,[n2]||$^{\infty}$]
*: n1=1, n2 = Anzahl der Datenwörter von "name"

Im Datenbereich der Entity "name" werden die Datenwörter vom n1-ten bis zum n2-ten einschließlich gelöscht und die folgenden nach vorne geschoben, so daß die Lücke geschlossen wird. Der Datenbereich der Entity wird entsprechend verkürzt. Es dürfen mehrere Paare (n1,n2) als Parameter angegeben werden; dabei muß der Wert n2≥n1 sein und der Wert n1' des nächsten Paares muß n1'>n2 erfüllen. Wird ein Wert n1 angegeben, das dazugehörige n2 jedoch nicht, so wird vom System n2=n1 gesetzt. Werden überhaupt keine Parameter n1,n2 angegeben, so werden alle Daten der Entity gelöscht.

3.2.4.7 Element

DLE,name,namecl[,code]
*: code = 0

Der Elementeintrag von "name" in der Multi-Value-Entity "namecl" wird gelöscht. "code" ist eine zweistellige INTEGER-Zahl, die den Löschmodus näher spezifiziert:

code-Einerstelle:

0: nur der Elementeintrag wird gelöscht
1: auch das Element "name" wird gelöscht, und zwar als Entity, wenn es durch das ID repräsentiert wird, und als Name, wenn es noch Synonyme besitzt und durch sein NA repräsentiert wird.

code-Zehnerstelle:

0: das Element wird durch sein ID repräsentiert
1: das Element wird durch sein NA repräsentiert

3.2.4.8 Fact

DLF,name

Das Fact "name" wird gelöscht.

3.2.5 CH - CHANGE-BEFEHLE

+ 3.2.5.1 Single-Value-Entity

CHS,name[,daten][']
*: keine Daten

Die Entity "name" wird gelöscht und durch eine Single-Value-Entity mit demselben Namen, ohne Synonyme und, falls angegeben, mit den Daten "daten" ersetzt.

3.2.5.2 Name

CHN,name neu,name alt

"name alt" wird ohne Änderung der Daten und ohne Wirkung auf die Synonyme in "name neu" umbenannt. Dieser Name wird als aktuellstes Synonym eingesetzt.

+ 3.2.5.3 Data

CHD,name[,daten][']
*: keine Daten

Die Daten der Entity "name" werden gelöscht und, falls angegeben, durch die Daten "daten" ersetzt.

3.2.6 CP - CØPY-BEFEHLE

3.2.6.1 Additionally

CPA,ziel,quelle,[n1],[n2],[i]
*: n1=1,n2 = Anzahl der Datenwörter von "quelle", i=-1

Die Datenwörter der Entity "quelle" vom n1-ten bis zum n2-ten einschließlich werden hinter das i-te Wort der Entity "ziel" eingeschoben, ohne schon vorhandene Daten zu löschen. Ist i negativ oder größer als die Anzahl der schon vorhandenen Datenwörter der Entity "ziel", so werden die neuen Daten an die alten hinten angehängt. Werden n1 und/oder n2 weggelassen, so werden sie durch Anfang bzw. Ende des Datenbereichs von "quelle" ersetzt. Existiert die Entity "ziel" noch nicht, so wird sie mit den zu kopierenden Daten als Single-Value-Entity erzeugt.

3.2.6.2 Deleting

CPD,ziel,quelle,[n1],[n2]
*: n1=1, n2 = Anzahl der Datenwörter von "quelle"

Die Wirkung ist dieselbe wie beim Befehl CPA, jedoch werden zuerst alle eventuell vorhandenen Daten der Entity "ziel" gelöscht.

3.2.7 RD - READ-BEFEHLE

3.2.7.1 Entity

RDE,name

Die vom Benutzer angegebenen Daten der Entity "name", ihr Umfang und die Klasse der Entity werden ausgegeben. Eine (gemäß /1/ verlangte) etwa angegebene Datenbereichsnummer wird ignoriert.

3.2.7.2 Synonym

RDS,name,n

Das Synonym von "name" mit der Nummer "n" und die Gesamtzahl des Synonyms von "name" werden ausgegeben.

3.2.7.3 Name

RDN,n1,n2[,n]
*: n = 0

Der zum ID oder NA (n1,n2) gehörende Name wird ausgegeben; dabei spezifiziert n die Bedeutung von (n1,n2):

n=0: (n1,n2) ist ID
n=1: (n1,n2) ist NA

3.2.7.4 ID

RDI,name

NA und ID von "name" werden ausgegeben

3.2.8 FD - FIND-BEFEHLE

3.2.8.1 Element

FDE,name,namecl[,code]
*: code = 0

Es wird festgestellt, ob "name" Element einer Menge "namecl" ist. "code" ist dabei eine einstellige INTEGER-Zahl, die angibt, wie das Element repräsentiert ist:

code = 0: das Element wird durch sein ID repräsentiert
code = 1: das Element wird durch sein NA repräsentiert

3.2.8.2 Relation-Entry

FDR,[name A],[name B],[name C],[code],[list]
*: name A = ?, name B = ?, name C = ?, code = 0, list = 0

Es wird festgestellt, ob der gefragte Relationszusammenhang besteht. Wurden Namen weggelassen, so wird gesucht, ob ein Zusammenhang zwischen den restlichen Entities besteht. Wurden zwei Namen weggelassen (Frageklasse 2), so wird der Ring, der die Antworten, d.h. die mit der angegebenen Entity assoziierten Entities, enthält, aufgelistet. Bei Fragen, bei denen nur ein Name weggelassen wurde (Frageklasse 1), hängt die Form der Antwort von der Anzahl der Entities, die gefunden wurden, und von dem in der Zahl "code" angegebenen Retrieval-Modus m ab. Wurde genau eine Entity gefunden, so wird bei m=0 oder m=1 deren Name ausgegeben. Wurden bei m=1 mehrere Entities gefunden oder war m=2, so werden alle gefundenen Entities als Elemente in eine Multi-Value-Entity abgespeichert, die vom System automatisch erzeugt wird. Der Name dieser Entity(,der zur Kennzeichnung mit einem Stern beginnt,) wird als Antwort übergeben.

Wurden bei m=0 mehrere Entities gefunden, so werden sie aufgelistet. Da durch die unten besprochene Überlauf-Simulation Mischformen dieser Antwort-Typen auftreten können, werden gefundene Einzel-Triaden und erzeugte Mengen generell zuerst ausgegeben, dann folgt die Trennzeile '******** TRIAD(E) GEFUNDEN' und dann erst eventuelle Ringlisten. Insgesamt ergibt sich für die Form

der Antworten bei den verschiedenen Fragemodi bei Fragen der Klasse 1 die folgende Übersicht:

Anzahl der Entities \ Modus m	0	1	2
1	Name	Name	Menge
> 1	Ring-Liste	Menge	Menge

(vgl. Abschnitt 1.3.3 und 1.3.4)

Bei allen Fragen der Klasse 1, die als Antwort einen Namen oder eine Menge zurückbringen, werden etwa gefundene ungültige Triaden (die ein nicht belegtes ID oder NA enthalten) automatisch gelöscht, da sie die Struktur inkonsistent machen. Verteilen sich die Antworten zu einer Frage auf mehrere Seiten des Triadenspeichers, so wird bei einem Aufruf von FDR für jede Seite, auf der Triaden gefunden wurden, eine Antwort ausgegeben; auf diese Art wird ein Seitenüberlauf im Triadenspeicher unnötig, da er beim Retrieval simuliert wird.

"code" ist eine zweistellige INTEGER-Zahl, die Speichermodus s und Retrieval-Modus m angibt. Die Zehnerstelle gibt den Speichermodus s an und entspricht somit der code-Zehnerstelle der Befehle STR,STM,DLR und DLM. Dabei werden jedoch nur diejenigen Bitstellen ausgewertet, für die Namen angegeben wurden. Die Einerstelle bestimmt den Fragemodus m; sie kann demnach die Werte 0, 1 und 2 haben.

"list" ist eine einstellige INTEGER-Zahl, die, wenn sie ≠ 0 angegeben wurde, die zusätzliche Ausgabe der Anfangsadressen aufgelisteter Ringe bewirkt.

3.2.8.3 Ring-Top

FDT[,i1,i2]

*: aktuelle, d.h. zuletzt von FDR oder FDN ausgegebene Werte von Seitennummer und Index im Triadenspeicher

Es wird der Name der Entity übergeben, die auf der Adresse (i1,i2) im Triadenspeicher steht.

3.2.8.4 Next

FDN[,i1,i2]
*: aktuelle Werte

Es werden Name und Adresse der nächsten Entity auf dem durch die Adresse (i1,i2) laufenden Ring ausgegeben, falls die angegebene Adresse nicht die der Bottom-Zelle (der letzten Zelle) des Rings war.

3.2.9 LI - LIST-BEFEHLE

3.2.9.1 Element

LIE,namecl[,n]
*: n = 0

Der Name des Elements von "namecl" mit der Nummer "n" und die Anzahl der Elemente von "namecl" werden ausgegeben. War n=0, so werden alle Elemente von "namecl" aufgelistet. Wird LIE auf einen ungültigen Elementeintrag oder ein nicht belegtes ID oder NA enthält) positioniert, so wird dieser automatisch gelöscht, da er die Struktur inkonsistent macht.

3.2.9.2 Ring

LIR[,i1,i2]
*: aktuelle Werte

Die Namen aller Entities auf dem durch die Adresse (i1,i2) laufenden Ring werden, beginnend bei dieser Adresse, bis zur Bottom-Zelle aufgelistet.

3.2.9.3 Synonyms

LIS,name

Alle Synonyme von "name" und ihre Anzahl werden aufgelistet.

3.2.10 SØ - SET-ØPERATION-BEFEHLE

3.2.10.1 Union

SØU,name A,name B,name C
SØ+,name A,name B,name C
} wahlweise

Wenn A und B Multi-Value-Entities sind, wird ihre Vereinigungsmenge A∪B gebildet und als Multi-Value-Entity C abgespeichert. Dabei darf C auch gleich A oder B sein. Single-Value-Entities A und/oder B werden vorher zu Multi-Value-Entities umdefiniert.

3.2.10.2 Difference

SØD,name A,name B,name C
SØ-,name A,name B,name C
} wahlweise

Wenn A und B Multi-Value-Entities sind, wird ihre mengentheoretische Differenz A∖B gebildet und als Multi-Value-Entity C abgespeichert. Dabei darf C auch gleich A sein. Single-Value-Entities A und/oder B werden vorher zu Multi-Value-Entities umdefiniert.

3.2.10.3 Intersection

SØI,name A,name B,name C
SØ*,name A,name B,name C
SØ.,name A,name B,name C
} wahlweise

Wenn A und B Multi-Value-Entities sind, wird ihr Durchschnitt A∩B gebildet und als Multi-Value-Entity C abgespeichert. Dabei darf C auch gleich A oder B sein. Single-Value-Entities A und/oder B werden vorher zu Multi-Value-Entities umdefiniert.

3.2.11 Steuerung von Ausgabe und Fragen an den Benutzer

3.2.11.1 LIST,NØLIST

LIST Die normale DATAS-Ausgabe von Meldungen und Namen erfolgt, bis der Befehl durch den Befehl NØLIST aufgehoben wird.

NØLIST Die Ausgabe von Namen und von Fehlermeldungen der DATAS-Programme wird unterdrückt, bis der Befehl durch den Befehl LIST aufgehoben wird. Fragen und Meldungen über Fehler,die zu

einer Beendigung des Dialogs führen, werden jedoch auch weiterhin ausgegeben.

*: Wurde noch keiner der beiden Befehle gegeben, so ist der Modus LIST eingestellt.

3.2.11.2 ASK,NØQ

ASK — Werden Rückfragen an den Benutzer notwendig, so erwartet das System die Antwort J oder ØK für "ja" bzw. N für "nein".

NØQ[,code] (No Questions)

*:code=0

Werden Rückfragen an den Benutzer notwendig, so wird zwar der Fragetext ausgegeben; das System nimmt aber generell an, daß die Antwort "nein" bei code=0 bzw. "ja" bei code=1 lautet. Eine Reaktion des Benutzers wird nicht erwartet; eventuelle Eingaben werden ignoriert.

*: Jeder dieser drei Modi gilt solange, bis einer der Befehle ASK oder NØQ ihn durch einen anderen Modus ersetzt. Wurde noch keiner der beiden Befehle gegeben, so ist der Modus **ASK** eingestellt.

3.3 Fehlermeldungen und Fehlerstops

3.3.1 Fehlermeldungen

Die im folgenden angegebene Liste der DATAS-Fehlermeldungen ist nach den Programmen geordnet, die die Fehlervariable setzen und gegebenenfalls eine Fehlermeldung ausgeben. Nach mehreren Operationen hat dabei die Fehlervariable den ihr zuletzt zugewiesenen Wert, der nur dann 0 ist, wenn die letzte ausgeführte DATAS-Operation korrekt abgeschlossen wurde. Die beiden höchsten Ziffern (in einigen Fällen auch noch die dritte) der Fehlernummer sind dem Programm zugeordnet, das die Fehlervariable setzt, die anderen beiden Ziffern (bzw. nur die letzte) sind eine laufende Nummer innerhalb des Programmes.

Bei Fehlern, die zu einer Ausgabe einer Fehlermeldung (im Ausgabezustand LIST) führen, ist die Meldung angegeben; diese Zeilen sind in der folgenden Liste durch die beiden Sternchen vor und hinter der Fehlernummer kenntlich. Bei den anderen Fehlern wird nur eine kurze Beschreibung der Fehlerursache angegeben. Danach wird jeweils der DATAS-Befehl (bzw. die Befehle) der Assemblersprache angegeben, der diesen Fehler verursachen kann; dabei bedeutet die Abkürzung "gen." (generell), daß dieser Fehler bei vielen oder den meisten Befehlen auftreten kann. Ist kein Befehl angegeben, so kann der betreffende Fehler in einem Assembler-Programm nicht auftreten.

Die hier angegebenen Fehler führen sämtlich nicht zum Abbruch eines DATAS-Laufes, sondern nur zum Abbruch der Bearbeitung des betreffenden Befehls. Daher können sie im Batch-Betrieb durch Abfrage der Fehlervariablen, im Dialog nach Entscheidung des Benutzers, zur Steuerung des weiteren Programm-Ablaufs benutzt werden. Eine Beschreibung der einzelnen, zur DATAS-Bibliothek gehörenden Programme ist für die Telefunken TR 440 im Anhang gegeben.

3.3.2 Fehlerliste

Programm	F.-Nr.	Meldung bzw. Beschreibung	Befehl(e)
DIALOG	**1001**	FALSCHE PARAMETERANZAHL	gen.
	1002	FALSCHE SEITENNUMMER	SYP
	1003	FALSCHE ZAHLENEINGABE	gen.
	(**5002**	ENTITY EXISTIERT NICHT	RDI)
EXTSYS	** -1**	UNBEKANNTE ANWEISUNG	gen.
ØCDUMP	**1241**	FALSCHER SEITENTYP	SYD
DINCØD	**1301**	KØMMAFEHLER	gen.
	1302	ZEICHEN NICHT IDENTIFIZIERBAR	gen.
DØUTCD	**1401**	DATENSATZ ZU LANG	-
ASK	**1501**	FRAGETEXT ZU LANG	-
NUM	**1701**	FALSCHE ZAHLENEINGABE	gen.
RTRNID	**5001**	ILLEGALER NAME	gen.
	5002	Entity existiert nicht	gen.
ALCNAM	**5101**	NAMENSPEICHER VØLL	gen.
STENT	**5201**	ILLEGALER NAME	gen.
	5202	DATENSPEICHER VØLL	gen.
	5203	UEBERSPEICHERN (falls nicht erlaubt)	STS
	5204	UEBERSPEICHERN (bei Synonym,falls nicht erlaubt)	-
	5205	UEBERLAUF BEI SYNØNYM-SPEICHERUNG	-
	5206	UEBERLAUF BEI SYNØNYM-SPEICHERUNG	DCS,CHN
	5207	ZU VIELE SYNØNYME	DCS,CHN
DSHIFT	**5301**	SEITENUEBERLAUF IM DATEN-SPEICHER	gen.
DELENT	5401	Entity nicht vorhanden	DLS,DLN
	5402	ILLEGALER LØESCHVERSUCH	DLN,DLE,DLC,DLR,DLI
STDATA	**5801**	UEBERLAUF BEI DATEN-SPEICHERUNG	STD,CHD,CPA,CPD
	5802	ILLEGALE DATENGRENZEN	CPA,CPD

Programm	F.-Nr.	Meldung bzw. Beschreibung	Befehl(e)
DLDATA	**5901**	ENTITY EXISTIERT NICHT	DLD
	5902	ILLEGALE DATENGRENZEN	DLD
CHNAME	**6001**	KEINE ENTITY DIESES NAMENS DA	CHN
	6002	UEBERSPEICHERN (falls nicht erlaubt)	CHN
DECSYN	**6101**	KEINE ENTITY DIESES NAMENS DA	DCS
	6102	UEBERSPEICHERN (falls nicht erlaubt)	DCS
STELEM	**6201**	RELATIØN KANN NICHT MENGE SEIN	STE,STC
	6202	ELEMENTBEZIEHUNG BESTEHT SCHØN	STE,STC
DLSET	6301	Menge existiert nicht	DLC
	6302	KEINE MULTI-VALUE-ENTITY	DLC
	6303	UNGUELTIGES ID	DLC
	6304	UNGUELTIGER NAME	DLC
DLELEM	6401	Element nicht vorhanden	DLE
	6402	MENGE NICHT VØRHANDEN	DLE
	6403	KEINE MULTI-VALUE-ENTITY	DLE
	6404	LEERE MENGE	DLE
	6405	KEINE ELEMENTBEZIEHUNG	DLE
PUTRIE	**6501**	KEIN NAME FREI	FDR
DLRELE	**6601**	NICHT VØRHANDEN -/namea	DLR
	6602	NICHT VØRHANDEN -/nameb	DLR
	6603	NICHT VØRHANDEN -/namec	DLR,DLM
	6604	KEINE RELATIØN	DLR,DLM
STRELE	**6701**	MENGE KANN NICHT RELATIØN SEIN	STR,STM,STF
	6702	A kann keinen Seiteneintrag mehr aufnehmen	STR,STM,STF
	6703	TRIADENSPEICHER VØLL	STR,STM,STF
SETØP	**6801**	NICHT VØRHANDEN/namea	SØU,SØD,SØI
	6802	NICHT VØRHANDEN/nameb	SØU,SØD,SØI
	6803	ILLEGALE DIFFERENZBILDUNG	SØD
	6804	UEBERSPEICHERN VØN/name (falls nicht erlaubt)	SØU,SØD,SØI
	6805	RELATIØN KANN NICHT MENGE SEIN (namea)	SØU,SØD,SØI

Programm	F.-Nr.	Meldung bzw. Beschreibung	Befehl(e)
	6806	RELATIØN KANN NICHT MENGE SEIN (nameb)	SØU,SØD,SØI
STSET	**6901**	ILLEGALER NAME	STC
	6902	ILLEGALE ELEMENTANZAHL	STC
FDELEM	**7001**	ELEMENT EXISTIERT NICHT	FDE
	7002	MENGE EXISTIERT NICHT	FDE
	7003	KEINE MENGE	FDE
LIELEM	**7101**	MENGE EXISTIERT NICHT	LIE
	7102	KEINE MENGE	LIE
	7103	LEERE MENGE	LIE
	7104	NUMMER NICHT IM ELEMENTBEREICH	LIE
	7105	ELEMENTEINTRAG WURDE GELØESCHT	LIE
RDENT	**7201**	ENTITY EXISTIERT NICHT	RDE,CPA,CPD
RDNAME	**7301**	KEINE ID-INFØRMATIØN	RDN
	7302	ILLEGALES ID	RDN,gen.
	7303	ID UNGUELTIG	RDN,gen.
	7304	KEIN SYNØNYM MIT DIESER NUMMER VØRHANDEN	RDS
	7305	ILLEGALE NAMEN-ADRESSE	RDN,gen.
	7306	KEIN NAME VØRHANDEN	RDN,gen.
RDSYN	**7401**	ENTITY EXISTIERT NICHT	RDS,LIS
FDRELE	**7501**	NICHT VØRHANDEN -/namec	FDR
	7502	NICHT VØRHANDEN -/nameb	FDR
	7503	NICHT VØRHANDEN -/namea	FDR
	7504	TRIADE NICHT VØRHANDEN	FDR
	7505	TRIADE UNGUELTIG	FDR
	7506	TRIADE GELOESCHT	FDR
	7507	RETRIEVAL ABGEBRØCHEN	FDR
	7508	KEINE RELATION	FDR
FDNEXT	**7601**	FALSCHE SEITENNUMMER	-
	7602	FALSCHER INDEX	-
	7603	BØTTØM SCHØN ERREICHT	-
FDRTØP	**7701**	FALSCHE SEITENNUMMER	-
	7702	FALSCHER INDEX	-

Programm	F.-Nr.	Meldung bzw. Beschreibung	Befehl(e)
LIRING	**7801**	FALSCHE SEITENNUMMER	FDT,FDN,LIR
	7802	FALSCHER INDEX	FDT,FDN,LIR
	7803	LEERE ZELLE	FDT,LIR
	7804	BØTTØM SCHØN ERREICHT	FDN
SYSEXP	**8301**	SYSTEMKAPAZITAET UEBERSCHRITTEN	SYE
	8302	SEITENLAENGE UEBERSCHRITTEN	SYE
	8303	SYSTEM-FILE ZU KLEIN	SYE
SYSLST	**8401**	FALSCHER SEITENTYP	SYL
	8402	LEERE SEITE	SYL
STFACT	**9001**	ILLEGALER NAME	STF
	9002	ILLEGALE KØMPØNENTEN-ANZAHL	STF
DLFACT	**9101**	NUR 300 KØMPØNENTEN GELØESCHT	DLF
	9102	illegales Fact, nicht vollständig gelöscht	DLF
STMANY	**9201**	ILLEGALER NAME	STM
	9202	ILLEGALE KØMPØNENTEN-ANZAHL	STM
DLMANY	**9301**	ILLEGALER NAME	DLM
	9302	NICHT VØRHANDEN/nameavnameb	DLM
	9303	ILLEGALE KØMPØNENTEN-ANZAHL	DLM

3.3.3 Fehlerstops

Bei einer Reihe von Fehlern wird die Bearbeitung des DATAS-Programms (Assembler oder vom Benutzer geschrieben) abgebrochen, und die Datenstruktur in den Zustand vor Beginn des Programm-Laufes versetzt. Diese Fehler lassen sich in zwei Gruppen aufteilen:

1. Fehlerhafte Eröffnung der Bearbeitung der Struktur (in der folgenden Liste durch * bezeichnet);
2. Systemfehler, eventuell durch eine zerstörte oder teilweise zerstörte Datenstruktur verursacht (in der folgenden Liste durch + bezeichnet, bzw. durch ++, wenn sie automatisch einen Dump der zuletzt bearbeiteten Seite erzeugen).

Fehler der zweiten Gruppe dürften ohne äußere Einwirkung auf den System-File nicht auftreten, solange dieser nur mittels des Assemblers bzw. korrekt aufgerufener DATAS-Unterprogramme bearbeitet wird. Die Fehler beider Gruppen weisen der Fehlervariablen keinen neuen Wert zu, sondern übernehmen den zuletzt aufgetretenen Wert, um so eventuell eine Rückverfolgung der Fehlerursache zu erleichtern. Die folgende Liste ist wieder nach Programmen geordnet.

Programm	Meldung
DIALØG	* FALSCHE BUFFERLAENGE
	* SYSTEM NICHT INITIALISIERT
DSHIFT	++ FEHLER IM ALGØRITHMUS
DELENT	++ FEHLER BEI NAME-CØNFLICT
	++ FEHLER BEI SYNØNYM-LØESCHUNG
SYSINT	* SYSTEMKAPAZITÄT UEBERSCHRITTEN
	* SEITENLAENGE UEBERSCHRITTEN
	* SYSTEM-FILE ZU KLEIN
PAGE	+ FALSCHER SEITENTYP
	+ FALSCHE SEITENNUMMER

3.4 Beispiel

Es folgt ein Listing eines Assembler-Programms, das das im Abschnitt 1.6 besprochene Beispiel einer Assoziation behandelt. Weitere Beispiele können in /1/ und /2/ gefunden werden; sie beziehen sich jedoch auf ältere Versionen des Systems und nutzen daher die durch die inzwischen erfolgten Erweiterungen des Systems gegebenen neuen Möglichkeiten nicht aus.

```
START DIALOG (10.18)

  SYI                                    FDR,RELATION,AUTO
                                        -------- RELATION
  STR,RELATION,AUTO,TUER                -------- AUTO
                                        -------- TUER
  STM,RELATION,BUS,,2,TUER,DACH         ******** TRIADE(N) GEFUNDEN

  FDR,RELATION                           FDR,RELATION,AUTO,DACH
******** TRIADE(N) GEFUNDEN             **7504** TRIADE NICHT VORHANDEN
-------- AUTO
-------- BUS
```

```
  FDR,RELATION,BUS
******** TRIADE(N) GEFUNDEN
-------- TUER
-------- DACH

  FDR,RELATION,BUS,,1
-------- RELATION
-------- BUS
-------- *OTUER
******** TRIADE(N) GEFUNDEN

  LIE,*OTUER
-------- TUER
-------- DACH
 ANZAHL DER ELEMENTE =     2
```

```
  FDR,RELATION,BUS,,,1
******** TRIADE(N) GEFUNDEN
-------- TUER
-------- DACH
 TRIADENSEITE =     1   INDEX =   706 (= 1302B)

  LIR
-------- TUER
-------- DACH

  SYD
GEAENDERT: &STDDB.NONAME (0001.00)
STOP
ENDE DIALOG (10.18)  0.78
```

Literatur

/1/ J. Encarnacao, G. Weck — Eine Implementierung von DATAS - Datenstrukturen in Assoziativer Speicherung, Institut für Angewandte Mathematik und Informatik der Universität des Saarlandes, Bericht Nr. A74-1, Saarbrücken, Januar 1974

/2/ G. Weck — SAD - Ein Modell einer Strukturunabhängigen Assoziativen Datenstruktur, Dissertation am Institut für Angewandte Mathematik und Informatik der Universität des Saarlandes, Saarbrücken, Dezember 1974

Anhang

Auf den folgenden Seiten wird eine Liste der zur DATAS-Bibliothek auf der Telefunken TR440 gehörenden Montageobjekte (Unterprogramme) und des DATAS-Interpreters angegeben. Die Liste enthält Angaben über Umfang, Quellsprache, Erstellungsdatum und eventuelle zusätzliche Eingänge der Montageobjekte sowie eine Beschreibung der Gebiete des Operators.

Auftrag 0428 BROELIST ** ** Universität Saarbrücken ** MV160105

MAINTENANCE

02.10.75 --- MV 00.1.024

OPERATOREN

NAME	GEN.-VERS.-NR.	ERST.-DATUM	NAME	GEN.-VERS.-NR.	ERST.-DATUM
DIALOG	(0010.24)	30.09.75			

MONTAGEOBJEKTE

NAME	GEN.-VERS.-NR.	ERST.-DATUM	NAME	GEN.-VERS.-NR.	ERST.-DATUM
AALLOC	(0010.00)	18.08.75	ALCNAM	(0010.14)	29.08.75
ALOOK	(0010.00)	18.08.75	ALPHAD	(0010.02)	19.08.75
ALPHAL	(0009.01)	05.08.75	ALPHAN	(0010.03)	20.08.75
ASK	(0009.07)	11.08.75	BKTIE	(0009.01)	05.08.75
BRETRI	(0009.01)	05.08.75	BRSTRT	(0009.01)	05.08.75
CHNAME	(0009.01)	05.08.75	CHSNGL	(0009.01)	05.08.75
CRETRI	(0009.01)	05.08.75	DECREL	(0010.23)	24.09.75
DECSET	(0009.01)	05.08.75	DECSYN	(0010.16)	01.09.75
DELENT	(0010.11)	28.08.75	DELETE	(0010.00)	18.08.75
DIALOG	(0010.20)	17.09.75	DINCOD	(0009.03)	07.08.75
DLDATA	(0010.22)	24.09.75	DLELEM	(0010.00)	18.08.75
DLFACT	(0009.07)	11.08.75	DLMANY	(0009.01)	05.08.75
DLNAME	(0009.01)	05.08.75	DLRELE	(0010.17)	03.09.75
DLSET	(0010.00)	18.08.75	DLSNGL	(0009.01)	05.08.75
DOUTCD	(0010.01)	19.08.75	DSHIFT	(0010.04)	25.08.75
DSINIT	(0009.01)	05.08.75	DUMP	(0009.01)	05.08.75
DUMPE	(0009.01)	05.08.75	DUMPN	(0009.01)	05.08.75
ERROR	(0010.19)	17.09.75	EXDATA	(0010.00)	18.08.75
EXTSYS	(0009.01)	05.08.75	F1INIT	(0009.01)	05.08.75
F2INIT	(0009.01)	05.08.75	F2REL	(0009.01)	05.08.75
FDELEM	(0010.00)	18.08.75	FDNEXT	(0010.00)	18.08.75
FDRELE	(0010.03)	20.08.75	FDRTOP	(0010.00)	18.08.75
GARCOL	(0009.03)	07.08.75	GOBOTM	(0009.01)	05.08.75
HASHNM	(0010.11)	28.08.75	HSINIT	(0009.01)	05.08.75
IHASH	(0009.01)	05.08.75	KEEPFR	(0009.01)	05.08.75
LIELEM	(0010.08)	26.08.75	LIRING	(0010.03)	20.08.75
LOGAUT	(.)		NOKOPF	(0001.00)	20.01.75
NSINIT	(0010.00)	18.08.75	NUM	(0010.04)	25.08.75
OCDUMP	(0009.01)	05.08.75	PAGE	(0010.04)	25.08.75
PEORDR	(0009.01)	05.08.75	PUTRIE	(0010.00)	18.08.75
RDENT	(0010.12)	28.08.75	RDNAME	(0010.00)	18.08.75
RDSYN	(0009.01)	05.08.75	REQCL1	(0009.01)	05.08.75
RETRIV	(0009.01)	05.08.75	RTRNID	(0010.20)	17.09.75
SETOP	(0010.21)	18.09.75	STDATA	(0010.00)	18.08.75
STELEM	(0010.16)	01.09.75	STENT	(0010.00)	18.08.75
STFACT	(0010.04)	25.08.75	STMANY	(0009.01)	05.08.75
STORE	(0010.00)	18.08.75	STRELE	(0010.16)	01.09.75
STSET	(0009.01)	05.08.75	STSNGL	(0009.01)	05.08.75
SYSEXP	(0010.16)	01.09.75	SYSGCL	(0010.00)	18.08.75

Auftrag 0428 BRDELIST ** ** Universität Saarbrücken ** MV160105

```
**********                MAINTENANCE                **********

  02.10.75                   ---                    MV 00.1.024

**********              MONTAGEOBJEKTE               **********

                GEN.-     ERST.-                        GEN.-     ERST.-
  NAME          VERS.-NR. DATUM      NAME             VERS.-NR. DATUM

  SYSINT        (0010.24) 30.09.75   SYSLST           (0010.06) 25.08.75
  SYSOUT        (0010.13) 28.08.75

**********                MAINTENANCE                **********

  02.10.75                   ---                    MV 00.1.024

**********               OPERATOREN                  **********

 GEBIET|OGNM  |PGNM  |LNG/K|&L1 AB|SSP|AA/S|VK  |LK  |I|V|RGNR|DA/S

 DIALOG       (0010.24) 30.09.75   |&L1:  23 K  |OKB:  27 GW |

 D   1 |DQ1   |      |   16|    0| J |   0|KSP|-   |-|-|   -|   -
 L   1 |LQ1   |-     |   12|    -| N |  16|KSP|TSP |J|J|   -|   -
 I   1 |-     |      |    7|    0| - |  21|-  |-   |-|-|   1|   5
```

Auftrag 0428 BRDELIST ** ** Universität Saarbrücken ** MV160105

**********		MAINTENANCE					**********	
02.10.75		---					MV 00.1.024	
**********		MONTAGEOBJEKTE					**********	
AALLOC	(0010.00)	18.08.75	FTN					
	DATEI(GW):	&M1:	21	&M2:	119	SNR:	1	
ALCNAM	(0010.14)	29.08.75	FTN					
	DATEI(GW):	&M1:	31	&M2:	307	SNR:	2	
ALOOK	(0010.00)	18.08.75	FTN					
	DATEI(GW):	&M1:	21	&M2:	130	SNR:	3	
ALPHAD	(0010.02)	19.08.75	FTN					
	DATEI(GW):	&M1:	49	&M2:	519	SNR:	4	
ALPHAL	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	45	&M2:	431	SNR:	5	
ALPHAN	(0010.03)	20.08.75	FTN					
	DATEI(GW):	&M1:	23	&M2:	262	SNR:	6	
ASK	(0009.07)	11.08.75	FTN					
	DATEI(GW):	&M1:	49	&M2:	470	SNR:	7	
BKTIE	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	21	&M2:	141	SNR:	8	
BRETRI	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	25	&M2:	219	SNR:	9	
BRSTRT	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	21	&M2:	84	SNR:	10	
CHNAME	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	33	&M2:	247	SNR:	11	
CHSNGL	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	27	&M2:	93	SNR:	12	
CRETRI	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	21	&M2:	159	SNR:	13	
DECREL	(0010.23)	24.09.75	FTN					
	DATEI(GW):	&M1:	25	&M2:	100	SNR:	14	
DECSET	(0009.01)	05.08.75	FTN					
	DATEI(GW):	&M1:	23	&M2:	73	SNR:	15	
DECSYN	(0010.16)	01.09.75	FTN					
	DATEI(GW):	&M1:	33	&M2:	235	SNR:	16	
DELENT	(0010.11)	28.08.75	FTN					

Auftrag 0428 BROELIST ** ** Universität Saarbrücken ** MV160105

**********	MAINTENANCE					**********
02.10.75	---					MV 00.1.024
**********	MONTAGEOBJEKTE					**********
	DATEI(GW):	&M1:	41	&M2:	718	SNR: 17
	EINGAENGE:	EDLENT				
DELETE	(0010.00)	18.08.75		FTN		
	DATEI(GW):	&M1:	33	&M2:	250	SNR: 18
DIALOG	(0010.20)	17.09.75		FTN		
	DATEI(GW):	&M1:	149	&M2:	5850	SNR: 19
DINCOD	(0009.03)	07.08.75		FTN		
	DATEI(GW):	&M1:	27	&M2:	329	SNR: 20
DLDATA	(0010.22)	24.09.75		FTN		
	DATEI(GW):	&M1:	35	&M2:	357	SNR: 21
	EINGAENGE:	EDLDAT				
DLELEM	(0010.00)	18.08.75		FTN		
	DATEI(GW):	&M1:	37	&M2:	415	SNR: 22
	EINGAENGE:	EDLELE				
DLFACT	(0009.07)	11.08.75		FTN		
	DATEI(GW):	&M1:	41	&M2:	332	SNR: 23
DLMANY	(0009.01)	05.08.75		FTN		
	DATEI(GW):	&M1:	33	&M2:	338	SNR: 24
DLNAME	(0009.01)	05.08.75		FTN		
	DATEI(GW):	&M1:	23	&M2:	69	SNR: 25
DLRELE	(0010.17)	03.09.75		FTN		
	DATEI(GW):	&M1:	41	&M2:	625	SNR: 26
	EINGAENGE:	EDLREL				
DLSET	(0010.00)	18.08.75		FTN		
	DATEI(GW):	&M1:	43	&M2:	466	SNR: 27
	EINGAENGE:	EDLSET				
DLSNGL	(0009.01)	05.08.75		FTN		
	DATEI(GW):	&M1:	23	&M2:	56	SNR: 28
DOUTCD	(0010.01)	19.08.75		FTN		
	DATEI(GW):	&M1:	29	&M2:	205	SNR: 29
DSHIFT	(0010.04)	25.08.75		FTN		
	DATEI(GW):	&M1:	29	&M2:	470	SNR: 30
DSINIT	(0009.01)	05.08.75		FTN		
	DATEI(GW):	&M1:	21	&M2:	83	SNR: 31
DUMP	(0009.01)	05.08.75		FTN		

Auftrag 0428 BRUELIST ** ** Universität Saarbrücken ** MV160105

**********		MAINTENANCE				**********	
02.10.75		---				MV 00.1.024	
**********		MONTAGEOBJEKTE				**********	
	DATEI(GW):	&M1:	39	&M2:	188	SNR:	32
DUMPE	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	39	&M2:	188	SNR:	33
DUMPN	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	39	&M2:	188	SNR:	34
ERROR	(0010.19)	17.09.75		FTN			
	DATEI(GW):	&M1:	51	&M2:	314	SNR:	35
EXDATA	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	29	&M2:	152	SNR:	36
EXTSYS	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	110	SNR:	37
F1INIT	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	21	&M2:	79	SNR:	38
F2INIT	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	21	&M2:	76	SNR:	39
F2REL	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	21	&M2:	96	SNR:	40
FDELEM	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	440	SNR:	41
	EINGAENGE:	EFDELE					
FDNEXT	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	33	&M2:	299	SNR:	42
FDRELE	(0010.03)	20.08.75		FTN			
	DATEI(GW):	&M1:	49	&M2:	1111	SNR:	43
	EINGAENGE:	EFDREL		ELIREL			
FDRTOP	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	33	&M2:	246	SNR:	44
GARCOL	(0009.03)	07.08.75		FTN			
	DATEI(GW):	&M1:	29	&M2:	1047	SNR:	45
GOBOTM	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	21	&M2:	77	SNR:	46
HASHNM	(0010.11)	28.08.75		FTN			
	DATEI(GW):	&M1:	31	&M2:	293	SNR:	47
	EINGAENGE:	EHASHN					

Auftrag 0428 BRDELIST ** ** Universität Saarbrücken ** MV160105

**********		MAINTENANCE				**********	
02.10.75		---				MV 00.1.024	
**********		MONTAGEOBJEKTE				**********	
HSINIT	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	21	&M2:	52	SNR:	48
IHASH	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	25	&M2:	112	SNR:	49
KEEPER	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	21	&M2:	151	SNR:	50
LIELEM	(0010.08)	26.08.75		FTN			
	DATEI(GW):	&M1:	39	&M2:	537	SNR:	51
	EINGAENGE:	ELIELE					
LIRING	(0010.03)	20.08.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	421	SNR:	52
LOGAUT	(.)			TAS			
	DATEI(GW):	&M1:	11	&M2:	46	SNR:	53
NOKOPF	(0001.00)	20.01.75		TAS			
	DATEI(GW):	&M1:	15	&M2:	55	SNR:	54
	EINGAENGE:	KOPF					
NSINIT	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	21	&M2:	79	SNR:	55
NUM	(0010.04)	25.08.75		FTN			
	DATEI(GW):	&M1:	29	&M2:	210	SNR:	56
QCDUMP	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	53	&M2:	366	SNR:	57
PAGE	(0010.04)	25.08.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	429	SNR:	58
PEORDR	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	19	&M2:	112	SNR:	59
PUTRIE	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	43	&M2:	691	SNR:	60
RDENT	(0010.12)	28.08.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	294	SNR:	61
	EINGAENGE:	ERDENT					
RDNAME	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	463	SNR:	62
RDSYN	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	33	&M2:	213	SNR:	63

Auftrag 0428 BROELIST ** ** Universität Saarbrücken ** MV160105

**********		MAINTENANCE				**********	
02.10.75		---				MV 00.1.024	
**********		MONTAGEOBJEKTE				**********	
REQCL1	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	29	&M2:	152	SNR:	64
RETRIV	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	33	&M2:	270	SNR:	65
RTRNID	(0010.20)	17.09.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	353	SNR:	66
SETOP	(0010.21)	18.09.75		FTN			
	DATEI(GW):	&M1:	45	&M2:	1053	SNR:	67
	EINGAENGE:	ESETOP					
STDATA	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	47	&M2:	644	SNR:	68
	EINGAENGE:	ESTDAT					
STELEM	(0010.16)	01.09.75		FTN			
	DATEI(GW):	&M1:	41	&M2:	460	SNR:	69
	EINGAENGE:	ESTELE					
STENT	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	43	&M2:	918	SNR:	70
STFACT	(0010.04)	25.08.75		FTN			
	DATEI(GW):	&M1:	31	&M2:	255	SNR:	71
STMANY	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	29	&M2:	222	SNR:	72
STORE	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	39	&M2:	486	SNR:	73
STRELE	(0010.16)	01.09.75		FTN			
	DATEI(GW):	&M1:	41	&M2:	962	SNR:	74
	EINGAENGE:	ESTREL					
STSET	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	29	&M2:	189	SNR:	75
STSNGL	(0009.01)	05.08.75		FTN			
	DATEI(GW):	&M1:	25	&M2:	76	SNR:	76
SYSEXP	(0010.16)	01.09.75		FTN			
	DATEI(GW):	&M1:	49	&M2:	862	SNR:	77
SYSGCL	(0010.00)	18.08.75		FTN			
	DATEI(GW):	&M1:	27	&M2:	104	SNR:	78

********** MAINTENANCE **********

02.10.75 --- MV 00.1.024

********** MONTAGEOBJEKTE **********

SYSINT	(0010.24)	30.09.75		FTN			
	DATEI(GW):	&M1:	45	&M2:	589	SNR:	79
SYSLST	(0010.06)	25.08.75		FTN			
	DATEI(GW):	&M1:	35	&M2:	383	SNR:	80
SYSOUT	(0010.13)	28.08.75		FTN			
	DATEI(GW):	&M1:	29	&M2:	240	SNR:	81