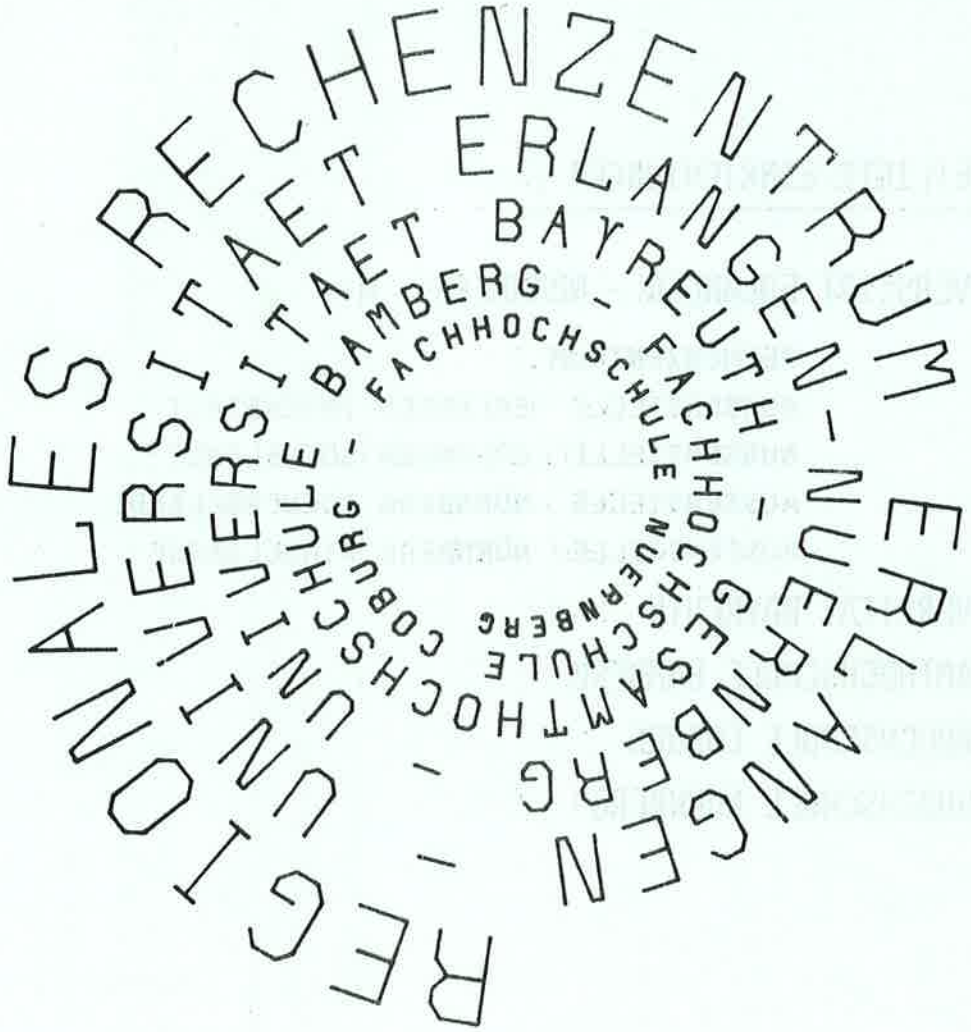
RRZE

BENUTZERINFORMATION



BI 16 -ERLANGEN- 12. FEBRUAR 1979

R R Z E
REGIONALES RECHENZENTRUM
MARTENSSTRASSE 1
8520 ERLANGEN
TEL: 09131 / 85 70 31 - 85 70 32

BETEILIGTE EINRICHTUNGEN :

UNIVERSITÄT ERLANGEN - NÜRNBERG MIT
- RECHENZENTRUM
- AUSSENSTELLE ERLANGEN INNENSTADT
- AUSSENSTELLE ERLANGEN SÜDGELÄNDE
- AUSSENSTELLE NÜRNBERG TUCHERGELÄNDE
- AUSSENSTELLE NÜRNBERG FINDELGASSE

UNIVERSITÄT BAYREUTH
GESAMTHOCHSCHULE BAMBERG
FACHHOCHSCHULE COBURG
FACHHOCHSCHULE NÜRNBERG

HERAUSGEGEBEN VOM REGIONALEN RECHENZENTRUM ERLANGEN

Inhaltsverzeichnis

1 Aktuelle Information

1.1 Lehrveranstaltungen

Im SS79 finden im RRZE die folgenden Lehrveranstaltungen statt:

a) Einführung in die Programmierung(PASCAL)

Anmeldung: Math. Institut, Zimmer 7
Zeit: Freitag, 14-16 Uhr
Raum: Hörsaal des Mathematischen Instituts

b) Methoden der Programmentwicklung

Durchführung größerer Programmiervorhaben mit Fallstudien;
FORTRAN- Kenntnisse sind erforderlich
Termin: nach Vereinbarung
Eine Vorbesprechung findet am 2.5 1979 um 14Uhr im Raum 2.03 RZ-Gebäude statt.

c) Einführung in die Benutzung der Rechenanlagen TR440 und CYBER

19.4.- 25.4.1979
Beginn jeweils 9 hct Raum 2.03

Do., 19.4.	Einführung in die Benutzung des TR440	9-17 Uhr
Fr., 20.4.	Dialogmöglichkeiten am TR440	9-15.30 Uhr
	Einführung in die Programmbibliothek des TR440	15.30-17 Uhr
Mo., 23.4.	Einführung in die Benutzung der CYBER	9-17 Uhr
Di., 24.4.	Dialogmöglichkeiten an der CYBER	9-15.30 Uhr
	Einführung in die Programmbibliothek der CYBER	15.30-17 Uhr
Mi., 25.4.	Einführung in die Benutzung des Grafik-Systems	9-17 Uhr

Bei allen Themen dieser Ausbildungswoche wird Gelegenheit zu praktischen Übungen gegeben.
Bitte benutzen Sie zur Anmeldung das letzte (blaue) Blatt dieser Benutzer-Information.

d) Einführung in SPSS

25.4.-27.4.1979
Beginn 9 hct, Raum 2.03
Voraussetzungen: Kenntnisse der Steuerkarten des TR440, z.B. durch Besuch der Einführung in der vorhergehenden Woche.
Anmeldung in der Aufsicht (Tel. 7039)

e) Einführung in die Programmierung (FORTRAN)

Anmeldung in der Aufsicht des RRZE
Zeit: Mi., 14-16 Uhr
Raum: H4, RZ-Gebäude

f) Einführung in die Programmierung (FORTRAN)

Kurs besonders für Chemie-Ingenieure
Anmeldung in der Aufsicht des RRZE
Zeit: 6.8.-17.8.1979
Raum: H4

g) Rechnergestützte Einführung in die Programmiersprache ALGOL

Anmeldung in der Aufsicht des RRZE
Beginn der Kurse: Mi.,21.2.1979, Di., 8.5.1979, Mi.,25.7.1979
Raum: 2.03, 18 Uhr
Weitere Informationen entnehmen Sie bitte einem besonderen Aushang.

h) Betrieb von Großrechenanlagen

Anmeldung in der Aufsicht des RRZE
Zeit: Di. 16-18 Uhr, Mi. 16-18 Uhr
Raum: H4

i) LISP- Ein Programmiersystem und seine Anwendungen

Zeit: voraussichtlich Mi., 10-12 Uhr
Raum: o2.151, Informatik-Gebäude

k) LISP- Ein Programmiersystem und seine Anwendungen

Seminar zur Vorlesung
Zeit: nach Vereinbarung
Raum: o2.151 Informatik-Gebäude

1.2 Benutzerkolloquium

a) Das Benutzerkolloquium am 14.11.1978

Der Bericht des Rechenzentrumsleiters umfaßte die folgenden Punkte:
-Betriebsstatistik
-Datenfernverarbeitung
-Erweiterungsantrag
-Rechenzeitabrechnung
für die FAU: Kontingent und Schwellwertberechnung
-Benutzerberatung
-10 Jahre Rechenzentrum

Aus der anschließenden Diskussion seien hier nur die Stichworte angegeben, da die meisten Punkte an anderer Stelle in dieser Benutzer- Information behandelt werden:
-Beschriften von Lochkarten an der CYBER
-Matrixdrucker am TR440
-Schriftenverkauf an Außenstationen
-Transport von Magnetbändern zwischen TR und CYBER
-Betrieb in den Weihnachtsferien
-Termin des nächsten Benutzerkolloquiums: 20.2.1979

b) Einladung zum Benutzerkolloquium

Das nächste Benutzerkolloquium findet am Dienstag, dem 20.2.1979 um 16hct im H4 statt. Alle Benutzer des RRZE sind dazu herzlich eingeladen. Bitte besprechen Sie Ihre Fragen möglichst vorher mit einem Benutzervertreter oder mit der Leitung des RRZE.

1.3 Neues zum Stande der Literatur

Dies ist eine Ergänzung zu der Liste in BI 14.

a) TR440 (neu erschienene CGK - Schriften)

440.Do.01	Kommandosprache TR440 Handbuch Nachtrag 11	29DM
440.Do.03	Kommandosprache TR440 Taschenbuch	8DM
440.F2.11	ALGOL-FORTRAN (Polynome etc.) Nachtrag 2	5DM
440.ZZ.06	Druckschriften-Preisliste Nachtrag 1	kostenlos

b) Folgende Schriften sind in der Aufsicht zu erwerben

	Kommandotaschenbuch TR440 zur MV19	3DM
	UPDATE Reference Manual	2.50DM
60497100	COBOL5 Reference Manual Revisionslevel E	10DM

1.4 Karten beschriften

Wir greifen eine Anregung auf, die im letzten Benutzerkolloquium das Problem betraf, daß recht lange Turnaroundzeiten für gestanzte Lochkarten-pakete zu erwarten sind, wenn diese auch gleichzeitig beschriftet werden sollen.

Wir bieten unseren Benutzern hiermit einen neuen Service an: Lochkartenstapel, die an den Anlagen ausgestanzt werden, werden von den Operateuren umgehend beschriftet, wenn
an TR440 das Materialkennzeichen MKZ=003
an CYBER der ID=3 über SETID, filename =3
angegeben wird.

Achtung: Es dürfen nur alphanumerische Lochkarten zum Beschriften abgegeben werden; auf keinen Fall Binärkarten!!

1.5 Das Informationssystem an den Rechenanlagen

Wichtige kurze Informationen (wenige Zeilen) werden an beiden Rechenanlagen dem Benutzer bei jedem Lauf auf die Kopfseite der Liste druckt, oder sie erscheinen z.B. am Bildschirm im Verlauf der Anmeldeprozedur. Sollten einmal längere Mitteilungen nötig sein, so steht dafür an beiden Rechnern ein File INFO zur Verfügung. Ob auf diesem File etwas Wissenswertes steht, wird durch die Kurzinformation (siehe oben) angegeben. Jeder Benutzer kann sich dann diese Files selbst ausdrucken lassen, unter Verwendung folgender Steuerkarten:
TR440: %LFAN.,UNRZDL.INFO %TKOP ., INFO, PROT.=KO
CYBER: Get,INFO/UN=UNRZDL. COPYSBF,INFO.

1.6 Neue Kommandos am TR440

a) Manipulation der LFD-BKZ-Liste
Um ein CYBER-ähnliches Arbeiten am TR440 zu ermöglichen, wurde das Kommando:

%BKZAENDERN,PASSWORT=/pw%/,AKZ=akz,EINTRAGEN=bkze, LOESCHEN=bkzf

implementiert. Die Angabe der Spezifikation PASSWORT ist obligat und bezeichnet das auftragseigene Paßwort; für die drei folgenden Spezifikationen ist jeweils der Wert -STD- zugelassen und bedeutet das auftragseigene AKZ. Damit wird ermöglicht:

- Jeder Benutzer kann jedem beliebigem Benutzer die LFD-Berechtigung für sein auftragseigenes AKZ eintragen, vorausgesetzt, in der BKZ-Liste ist noch ein Eintrag frei (AKZ=bel,EINTRAGEN=-STD-)
- Jeder Benutzer kann jedem beliebigen Benutzer die LFD-Berechtigung für sein auftragseigenes AKZ wieder entziehen (AKZ=bel,LOESCHEN=-STD-)
- Jeder Benutzer kann beliebige BKZ-Einträge in seiner BKZ-Liste löschen (AKZ=-STD-,LOESCHEN=bel)

Die BKZ-Liste eines Benutzers enthält maximal 6 Einträge. Die Reihenfolge der Einträge kann nicht beeinflußt werden, daher Vorsicht mit dem ersten BKZ-Eintrag (auftragseigenes BKZ)! Die Aufsicht trägt jedem Benutzer standardmäßig sein AKZ und das BKZ LFTEMP ein; die restlichen 4 Einträge sind leer. Ein Informieren über die von einem Benutzer an andere AKZ vergebenen BKZ-Berechtigungen ist nur über die Aufsicht möglich.
Das BKZ LFTEMP erfährt keine Sonderbehandlung!

Ist AKZ das auftragseigene oder -STD-, so erhält der Benutzer die Liste seiner BKZ-Berechtigungen ausgedruckt.

Die BKZ-Änderung wirkt sich erst beim nächsten neu zu beginnenden Auftrag aus!

b) Kontostandsabfrage
Durch das Kommando

%KONTO

erhält der Benutzer seinen aktuellen Kontostand und die bis dahin in diesem Auftrag angelaufenen Rechenzeitkosten ausgedruckt.

1.7 Änderungen zur Paßwortangabe am TR440

- Die Beschränkung:

  "Das Auftragspaßwort muß in den Spalten 1-6 auf der ersten Lochkarte hinter der XBA-Karte stehen"

  ist aufgehoben und lautet nun wie folgt:

  "Das Auftragspaßwort besteht aus 6 Zeichen und muß auf der ersten Lochkarte hinter dem XBA-Kommando stehen"

  Damit wird ermöglicht, daß der Benutzer seinem Auftragspaßwort das Zeichen "TEXTENDE" (=%037) voranstellt; bei Ausführung des Kommandos %LISTINPUT wird dann das Paßwort nicht mehr mit aufgelistet.
- Ab 1.3.1979 ist die Angabe des Auftragspaßwortes für alle Benutzer zwingend vorgeschrieben (auch dann, wenn er ein triviales Paßwort bestehend aus 6 Leerzeichen besitzt). Das triviale Paßwort wird in Zukunft bei Antragsverlängerung bzw. Neuantrag nicht mehr zugelassen.
- Bei der Anfrage des Auftragspaßwortes im Dialog wird in Zukunft am FSR-Anschluß eine Schwärzung der Eingabezeile ausgegeben (CYBER-analog: Übereinanderdruck). Am FSR und SIG51-Anschluß wird die eingegebene Zeile anschließend überschrieben.
- Bei Lochstreifeneingabe muß die Reihenfolge der ersten drei Steuerkarten folgendermaßen aussehen:

  ⍰4XBA,BEN=... %.
  ⍰4XUM,COD=SC4%.
  Paßwort

1.8 Pretty-Printer

Im Raum 1.08 des RZ-Gebäudes steht ein Typenrad-Drucker DIABLO 1620. Das Gerät arbeitet mit 30 Zeichen/sec und ist demnächst wahlweise an TR440 und CYBER anschließbar (zur Zeit nur an CYBER).

Es ist standardmäßig mit einem Typenrad German Pica ausgerüstet, und eignet sich gut als Pretty-Printer für Veröffentlichungen und Ähnliches. Dieser Teil der BI wurde bespielsweise am DIABLO gedruckt. Farbbänder sind bei den Operateuren an der CYBER zu haben (Tel. 7037). Fachmann für dieses Gerät ist Herr Büttner, Tel. 7809.

1.9 Tischrechner

Das Rechenzentrum verfügt seit Januar 1979 über einen Basic-Tischrechner HP 9845S. Er läuft gegenwärtig im Probebetrieb. In Kürze wird darüber ausführlich berichtet werden.

1.10 Listenausgabe über Schließfächer

Wir möchten nochmals darauf hinweisen, daß wir Listen nur dann in die Schließfächer ablegen, wenn das explizit gewünscht wird. Bitte geben Sie dann Ihren Wunsch in folgender Form an:

1) am TR440:
ß2XBA, BEN=bennr Fach-xy

2) an der CYBER:
Name.
User,...
Charge,...
HEADER. Fach xy

1.11 Betriebsstatistik

Zu der Tabelle auf der nächsten Seite möchten wir die folgenden Anmerkungen machen:

- In der Ausfallzeit sind auch Stromausfall und Klimastörungen enthalten.
- Der mittlere Fehlerabstand bezieht sich auf die Standzeit der Anlagen selbst,die zwischen zwei Totalzusammenbrüchen liegt.
- Die berechnete CPU-Auslastungder Anlagen konnte aufgrund einer fehlerbehafteten Erfassungsweise nur mit einer Tol eranz von ±3% angegeben werden.

Betriebsstatistik

	Gesamtbetriebszeit GBZ	Rechenzeit RZ	Ausfallzeit AZ	Wartgs.-zeit	Ausfallzeit % d. GBZ	Mittl. Fehlerabstand	Auslastung %
Plan	333 h/Monat	291 h/Monat	0 h/Monat	36 h/Monat	0 %	70 h	66 %
CY Januar	509	426	8	40	2	213	96
TR	464	300	100	30	21	21	86
CY Februar	677	588	24	31	4	196	84
TR	571	418	72	35	18	42	37
CY März	721	637	16	37	2	212	67
TR	625	477	83	40	20	40	84
CY April	628	516	20	65	5	172	65
TR	485	420	22	28	6	105	63
CY Mai	644	566	11	23	2	142	53
TR	567	454	40	34	7	35	44
CY Juni	597	537	19	26	3	77	85
TR	430	329	29	53	7	41	74
CY Juli	570	527	18	23	3	264	97
TR	443	374	6	44	1	37	46
CY August	758	712	1	31	0	356	66
TR	520	334	14	26	3	37	47
CY September	624	568	31	25	5	189	85
TR	416	289	1	24	0	36	72
CY Oktober	556	491	16	16	3	246	86
TR	424	363	11	25	3	26	60
CY November	731	678	36	12	5	170	66
TR	627	503	74	50	12	36	56
CY Dezember	610	571	5	31	1	286	92
TR	330	273	32	25	10	55	91
CY 1978 insgesamt	7625	6547	205	360	2,91	252	78,5
TR	5902	4534	484	414	9	43	58,8

2 Neues von der Software

2.1 Neues vom Betriebssystem für die CYBER

a) Einführung von Release 4

Im Dezember 1978 wurde nach internen Tests die neue Betriebssystemversion "Release 4" zum Testen für die Benutzer angeboten. Von diesem Angebot haben etliche Benutzer Gebrauch gemacht, so daß es kein Sprung ins kalte Wasser mehr war, als am 2.1.1979 der Betrieb mit Release 4 nach der Weihnachtspause begonnen wurde.

b) Änderungen in Release 4

- Zur Erhöhung der Sicherheit gegenüber Mißbrauch von Benutzernummern muß in Zukunft mit jeder Benutzernummer ein Paßwort von mindestens 4 Zeichen Länge verbunden sein. Beim Eintragen einer Benutzernummer durch das Rechenzentrum wird ein Paßwort mit eingetragen, und bei jeder Änderung eines Paßwortes wird durch das System verlangt, daß das neue Paßwort mindestens 4 Zeichen lang ist. Benutzer mit einer "alten" Benutzernummer können unter Umständen noch ohne ein Paßwort der obigen Mindestlänge weiterarbeiten. Das Rechenzentrum wird jedoch in der nächsten Zeit auch diese Benutzer dazu zwingen, ein (mindestens 4 Zeichen langes) Paßwort zu verwenden.
- Es wird dringend empfohlen, Fortran-, Basic-, Cobol- und Compass- Programme neu zu übersetzen. In einigen Fällen ist dies sogar unbedingt notwendig.
- Fortran- Programme mit eigener Feldlängenmanipulation (z.B. dynamische Vergabe von Blank-Common) müssen mit
  FTN (STATIC,...)
  übersetzt werden.
- "-" statt "CALL" an den Terminals kann nicht mehr benutzt werden. Procedure Files müssen auch im Dialog mit "CALL" gestartet werden.
- Das Programmsystem "FORM" hat sich völlig geändert. Es gibt dazu ein neues Reference Manual.
- Der "CYBER Record Manager" hat sich erheblich geändert. Es gibt jetzt für "Basic Access Methods" und "Advanced Access Methods" zwei neue Reference Manuals.
- Zusätzlich zur bisherigen Steuerkartensprache KCL (Kronos Control Language) gibt es eine neue Steuerkartensprache CCL (Cyber Control Language). Langfristig ist die völlige Umstellung von KCL auf CCL geplant. In Release 4 existieren KCL und CCL unabhängig voneinander.
- Die Vorbesetzung von undefinierten Variablen hat sich geändert. Damit ergibt sich in Zukunft eine höhere Wahrscheinlichkeit, daß die Verwendung von undefinierten Integer-Variablen zur Laufzeit durch "Error mode 4" angezeigt wird.
- Die Diagnostik des Fortran-Übersetzers ist besser geworden. Dadurch kann es vorkommen, daß Programme, "die schon immer so gelaufen sind" zu Recht als fehlerhaft zurückgewiesen werden.
- Das interne Format von index-sequentiellen Files hat sich geän-

dert. Eine Übernahme der alten Files oder eine Konvertierung sind zwar möglich, wir empfehlen aber die Neuherstellung solcher Files.
- Beim Arbeiten mit APL an "normalen" Terminals ist das APL-System mit dem Kommando

  APL, TT=713

  aufzurufen.

c) Umstellungsprobleme

- Von ALGOL existiert noch keine neue Version, es wird die alte Version (von Release 3) weiter verwendet.
- Dasselbe wie für ALGOL gilt auch für SIMULA.
- In Fortran-Programmen können die Anweisungen "REWIND" und "ENDFILE" nicht mehr dazu verwendet werden, am Terminal die sofortige Ausgabe von Zeilen zu erzwingen. (Ansonsten funktionieren diese Anweisungen aber noch.) Bei dringendem Bedarf für eine Ersatzlösung wenden Sie sich bitte an die Aufsicht oder Beratung.
- Die Ausführungsgeschwindigkeit von manchen Jobs hat abgenommen. Das wirkt sich auf die Benchmark-Messungen des Rechenzentrums mit etwa 10 % aus, Benutzerprogramme scheinen jedoch weniger davon betroffen zu sein.

d) Fehler

Bitte wenden Sie sich an die Aufsicht oder an die Beratung, wenn Sie Fehler in Release 4 bemerken oder vermuten. Seit der Inbetriebnahme sind die folgenden Fehler bekannt geworden, die zur Zeit auch bearbeitet werden:
- Das Ändern von File-Zuordnungen beim Starten von Programmen durch Angabe der Filenamen auf der LGO-Karte führt zu Schwierigkeiten, falls
  1) auf der LGO-Karte mehr Files angegeben werden als im (Fortran-) PROGRAM-Statement, oder
  2) im PROGRAM-Statement nicht die Reihenfolge: Angabe der Filenamen, dann Angabe der Zuordnungen zu Fortran- Unit- Numbers eingehalten wurde.
  Also:
  PROGRAM X(F1,F2,TAPE1=F1,TAPE2=F2)
  läuft, während
  PROGRAM Y(F1,TAPE1=F1,F2,TAPE2=F2)
  zu Schwierigkeiten führt.
- Der FTN-Post-Mortem-Dump bringt gelegentlich oktale Dumps von sich selbst. Es bsteht der Verdacht, daß dieser Fehler beim Mischen von alten und neuen Binärdecks verursacht wird.
- Die Möglichkeit, Control-Bytes zur Steuerung der Ein/Ausgabe an TELEX von FORTRAN aus auszugeben, funktioniert nicht mehr. Diese speziellen Zeichenfolgen (z.B. :A (Doppelpunkt gefolgt von A)) werden nicht zur E/A-Steuerung verwendet, sondern als Klartext am Terminal ausgeben.
- Beim Mischen von Haupt- und Unterprogrammen in COBOL5 kann es zu einem Compilerfehler kommen. Abhilfe: Unter Verwendung der

entsprechenden Steuerkartenparameter Haupt- und Unterprogramme getrennt übersetzen.
- Das neue System zeigt eine fatale Neigung, sich "aufzuhängen". Dadurch sind gelegentlich Betriebsstörungen durch "Deadstart" bedingt, die insbesondere auch bei Telex-Betrieb einen Wiederbeginn ohne RECOVER-Möglichkeit erfordern können. Diesen Fehlern gilt selbstverständlich zur Zeit unsere größte Anstrengung.

e) Zeitplan für die nächsten Änderungen

Es ist zu hoffen, daß im Februar 1979 die gravierendsten Fehler aus dem Betriebssystem beseitigt werden können. Im März 1979 soll dann die Umstellung der Datenfernübertragungssoftware auf "CCP 3.1" erfolgen. Für die Benutzer werden sich daraus hoffentlich keine wesentlichen Änderungen ergeben. Eine weitere größere Änderung ist dann frühestens Ende des Jahres 1979 zu erwarten.

2.2 PASCAL - Compiler am TR440

Am TR440 steht eine verbesserte Version des PASCAL - Compilers zur Verfügung.

Der Compiler und das Laufzeitsystem für PASCAL - Programme liegen in einer Bibliothek und werden mit dem Kommando
%BIBANMELDE,BIBLIOTHEK=PASCAL,TRAEGER=LFD
bereitgestellt.

Die Besonderheiten der TR440 Implementierung sind der Beschreibung im Anhang dieser BI zu entnehmen.

2.3 PFORT (CYBER und TR440)

An der CYBER ist eine neue Version des PFORT-Verifiers implementiert. In dieser Version wurde eine neue Option eingebaut:
K Kompaktes Listing ohne Seitenvorschübe
NK Listing wie bisher mit Seitenvorschüben.
Die Voreinstellung ist K.
Diese Option wird wie die anderen Options über eine Kommentarkarte in der Quelle gesteuert, z.B.

```
C* NK,NP,NR
```

Die Implementierung dieser Version an der TR440 wird in Kürze abgeschlossen sein. Eine ausführliche Beschreibung ist in Vorbereitung.

2.4 Neue SPSS- und BMDP-Version am TR440

a) Neues SPSS Version

Die im Oktober installierte SPSS-Version 6.02 mit dem SPSS-Operator 602.13 hat sich erfreulicherweise bewährt.

Die wesentlichen Verbesserungen gegenüber SPSS-Operator 6.08 sind:

- Die Fehler in den Einleseroutinen der Daten werden größtenteils behoben. Schwierigkeiten treten noch bei der Abarbeitung zu langer Formatangaben auf. Die Gesamtzahl der Formatelemente sollte nicht größer als 120 und die in einer Klammer nicht größer als 20 sein (Abhilfe: nur die benötigten Variablen einlesen oder, wenn nötig, Variable mit ADD VARIABLES in einem folgenden Lauf hinzufügen).
- Das SPSS-Kommando hat die neue Spezifikation
  LISTING = datei
  erhalten, die es ermöglicht, das Druckerprotokoll in eine Datei auszugeben.
- Bei fehlender OUTPUT-Spezifikation kreiert SPSS eine Datei, falls eine BCD-Ausgabe erfolgt.

Näheres erfahren Sie in den SPSSNEWS (siehe Anhang), die durch
%ERZEUGE,DOKUMENT.SPSSNEWS(Gerät)
ausdruckbar sind. Die Kommandobeschreibung DOKUMENT.SPSS und die Liste der bekannten Fehler DOKUMENT.SPSSFEHLER sind leider noch auf dem Stand der alten Version.

Die Auslieferung der SPSS-Version 7 hat sich verzögert, ein Installationstermin kann noch nicht genannt werden.

b) Neue BMDP-Version

Am TR440 steht die neuere BMDP-Version vom April 1977 zur Verfügung. Diese Version wurde - wie auch die vorhergehende - von Herrn Koll vom Regionalen Hochschulrechenzentrum der Universität Kaiserslautern von der IBM-Version für den TR440 aufbereitet.

Am RRZE wird BMDP über die Kommandoprozedur BIMED aufgerufen. Um den Umgang mit BMDP zu erleichtern, wurde BIMED um einige Spezifikationen erweitert und das Kommando BIMEDT in BIMED integriert. Mit der jetzt eingebauten dynamischen Speicherverwaltung entfallen die Kosten für die Montage, die bei der Vergrößerung des Arbeitsspeichers nötig war.

In dieser Version sind folgende Programme hinzugekommen :
BMDP2F, BMDP3F, BMDPAM, BMDP9R.
Die BIMED-Kommandobeschreibung mit Beispielen (siehe Anhang) erhalten Sie durch:
%ERZEUGE,DOKUMENT. BMDP
auf dem Drucker DR(2,0)-DC2 (Groß/Kleinschrift) im RRZE, bzw. durch

%ERZEUGE,DOKUMENT.BMDP(Gerät)
auf dem als "Gerät" angegebenen Drucker; ebenso ist sie in Band 67 der RRZE-Dokomentationen zu finden.

2.5 Neue Bibliothekskommandos (Gemeinschaftsbibliotheken) am TR440

Das in der Betriebssystemversion MV 19 eingeführte Kommando BERZEUGE legt Privatbibliotheken an, auf die nur von auftragseigenen LFD-Benutzerkennzeichen zugegriffen werden kann. Bis zur Behebung dieses Fehlers durch die Herstellerfirma CGK stellt das RRZE eine Kommandoprozedur in der Programmbibliothek zur Verfügung mit der Gemeinschaftsbibliotheken initiert werden können:
%ERZEUGE,KOMMANDO.BINIT
%BINIT,TRAEGER,(GV) (GV= 9999.99)
TRAEGER und GV entsprechen den Spezifikationen im BERZEUGE-Kommando, GV jedoch nur in der Form (g.v.). Anschließend kann mit BERZEUGE unter Angabe der gleichen GV-Nummer eine Gemeinschaftsbibliothek erstellt werden.
BINIT darf nur vor dem Neuerrichten und nicht vor der Erweiterung von Bibliotheken gerufen werden, da bereits existierende Bibliotheken gelöscht werden.

Die Firma CGK weist darauf hin, daß eine Mischbearbeitung von Bibliotheken durch die Operatoren PS&BIBTRANS (alte Kommandos), PS&BIB (neue Kommandos) und PS&KOP (kopiere etc.) wegen Inkompatibilitäten zu Fehlern führt.

2.6 Programmbibliothek TR440

Aufgrundvon Inkompatibilitäten mit der neuen Systemversion MV19 steht absofort das Gedächtnis vom 7.2.1978 nicht mehr zur Verfügung. Die Benutzer werden nochmals auf das in der BI 14 und der BI 13 beschriebene Kommando ERZEUGE verwiesen.

3 Neuer FORTRAN Standard

3.1 Einleitung

Im April 1978 wurde die Normung eines neuen FORTRAN Standards abgeschlossen. Der offizielle Titel lautet:

> American National Standard Programming Language
> FORTRAN X3.9-1978,

üblicherweise spricht man jedoch immer von FORTRAN 77.

Für den Einsatz von FORTRAN 77 an den beiden Rechenanlagen des Regionalen Rechenzentrums Erlangen gibt es von Seiten der Hersteller folgende Aussagen:

Die Firma CDC arbeitet an einem FORTRAN 77 Compiler. Dieser wird ab Mitte 1979 an der Cyber verfügbar sein. Gleichzeitig wird ein Konvertierungsprogramm bereitgestellt, das die Konvertierung von FORTRAN 4 Programmen in FORTRAN 77 Programme erleichtert.

Die Firma CGK hat sich vertraglich verpflichtet unter bestimmten Voraussetzungen ebenfalls einen FORTRAN 77 Compiler zu implementieren. Derzeit steht aber der Zeitpunkt, zu dem auch an der TR440 ein FORTRAN 77 Compiler verfügbar sein wird, noch nicht fest.

In dem folgenden Text werden einige der neuen Möglichkeiten von FORTRAN 77 beschrieben.

3.2 Änderungen gegenüber FORTRAN 4

Die wesentlichen Änderungen von FORTRAN 77 gegenüber FORTRAN 4 umfassen ein neuer Datentyp CHARACTER, die PARAMETER-Anweisung, neue Kontrollstrukturen und neue Möglichkeiten der Ein- und Ausgabe. Zu allen Punkten werden hier Erläuterungen gegeben, dabei wird eine Vollständigkeit nicht angestrebt.

a) Datentypen und Datenstrukturen

Bei den Datentypen und den Datenstrukturen sind zu erwähnen der neue Datentyp CHARACTER, die PARAMETER-Anweisung und die neuen Möglichkeiten der Bildung von Feldern.

1) CHARACTER

Der neue Datentyp CHARACTER dient zur Zeichenverarbeitung und ersetzt das bisherige Hollerith-Daten-Konzept. Mit der Einführung des Datentyps CHARACTER ist nun auch eine portable Zeichenketten-(String-)Verarbeitung möglich. Jede CHARACTER-Variable erhält als zusätzliches Kennzeichen die Anzahl der in ihr darstellbaren Zeichen, also die Länge des in der Variablen abgespeicherten Strings. Der Default-Wert ist 1. Die folgenden Deklarationen sollen dies erläutern:

```
CHARACTER*80 KARTE
CHARACTER SPALTE(80)
CHARACTER*6 NAME, INITLS*2, B(2,4)
```

KARTE ist eine CHARACTER-Variable der Länge 80, also das Abbild einer Lochkarte.
SPALTE ist ein eindimensionales Feld von 80 Elementen der Länge 1, also das Abbild der Spalten einer Lochkarte.
NAME ist eine Variable der Länge 6, INTLS eine der Länge 2 und B ein 2*4 Feld mit Elementen der Länge 6.

CHARACTER-Konstanten sind in Apostroph eingeschlossene Zeichenketten, das Apostroph innerhalb einer Zeichenkette wird durch ein Doppelapostroph dargestellt, z.B.

```
'RRZE'
'+*-/'
'DON''T'
```

In einer CHARACTER-Konstanten dürfen alle in einer Maschine darstellbaren Zeichen vorkommen.

Mit CHARACTER-Konstanten und CHARACTER-Variablen können folgende Operationen durchgeführt werden:

- Zuweisung
- Konkatenation
- Auswahl von Teilstrings
- Vergleiche gemäß der Interndarstellung

Es gibt u.a. 3 Intrinsic Functions zur Behandlung von CHARACTER-Variablen:

- ICHAR : Umwandlung CHARACTER --> INTEGER
- CHAR : Umwandlung INTEGER --> CHARACTER
- INDEX : Prüfung auf Teilstring und Angabe der Anfangsposition

Darüber hinaus gibt es 4 Intrinsic Functions zum lexikografischen (genormten) Vergleich zweier Strings. Es sind dies die Funktionen:

LGE : lexikografisch größer oder gleich
LGT : lexikografisch größer
LLE : lexikografisch kleiner oder gleich
LLT : lexikografisch kleiner

2) Die PARAMETER-Anweisung

Mit der PARAMETER-Anweisung kann man benannte Konstanten deklarieren. Damit erhält man Programme, die wesentlich besser änderbar sind als bisherige FORTRAN 4 Programme, da bei Änderung einer Konstanten diese Änderung nur noch an einer Stelle durchgeführt werden muß. Außerdem haben Programme mit benannten Konstanten einen wesentlich höheren Dokumentationswert als Programme ohne solche Konstanten.

Das folgende Programm zeigt die Verwendungsmöglichkeiten der PARAMETER-Anweisung sowie die neue (optionale) PROGRAM-Anweisung:

```
      PROGRAM XAMPL P
      CHARACTER BLANK
      PARAMETER (
     1   LENGTH=80,
     2   N = 9,
     3   N ROWS = N*N -2*N
     4   N COLS = 4*N + 1,
     5   N ENTRY = NROWS * NCOLS,
     6   BLANK = ' ')
      IMPLICIT CHARACTER*(LENGTH) (C)
      DIMENSION CARRAY (NROWS,NCOLS)
      COMPLEX I, ONEI
      LOGICAL TRUE, FALSE
      CHARACTER MSG *(*)
      PARAMETER (
     A   I=(0.0,1.0),
     B   ONE I = 1 + I,
     C   ERRLIM = 1.0E-5,
     D   TRUE = .TRUE.,
     E   FALSE = .FALSE.,
     F   MSG = 'INPUT ERROR')
      LOGICAL CNVRGD
      DATA CARRAY/NENTRY * BLANK/
      ...
      ...
      IF(X.LT.0) WRITE (6,A) MSG
      IF (REL DIF (X, OLD X).LT.ERRLIM) CNVRGD = TRUE
```

3) Felder

Felder können bis zu 7 Dimensionen haben und die untere Feldgrenze kann eine beliebige ganze Zahl sein, der Default-Wert ist 1.

Beispiele:

```
      DIMENSION A(2,3,3,2)
      REAL B(-5:5,6,0:9)
      CHARACTER C(0:9,6)*5
```

b) Kontrollstrukturen

Bei den Kontrollstrukturen sind wesentliche Erweiterungen mit einer neuen zweiseitigen IF-Anweisung und bei der DO-Anweisung eingeführt worden.

1) Die IF-Anweisung

Es wurde eine neue blockorientierte IF-THEN-ELSE-ENDIF Konstruktion eingebaut, die etwa der IF-Anweisung in ALGOL 60 ähnlich ist. Mit dieser Konstruktion ist es erstmals möglich, auch in FORTRAN lesbare strukturierte Programme zu schreiben. Das folgende, selbsterklärende Beispiel soll dies erläutern:

```
      LOGICAL FUNCTION PRIM(N)
      IMPLICIT INTEGER (A-Z)
      IF (N.LE.1) THEN
         PRIM=.FALSE.
      ELSE IF (N.EQ.2) THEN
         PRIM=.TRUE.
      ELSE IF (MOD(N,2).EQ.0) THEN
         PRIM=.FALSE.
      ELSE
         DO 18 DIVIDR = 3, INT(SQRT(REAL(N))), 2
           IF (MOD(N,DIVISR).EQ.0) THEN
             PRIM=.FALSE.
             RETURN
           ENDIF
18       CONTINUE
         PRIM=.TRUE.
      ENDIF
      END
```

2) Die DO-Anweisung

Die DO-Anweisung und die Ausführung von DO-Schleifen haben sehr viele gravierende Änderungen erfahren. Bis auf die sinnvolle Einschränkung, daß kein "extended range" mehr erlaubt ist, sind jedoch DO-Schleifen von FORTRAN 4 auch in FORTRAN 77 zulässig. Zu beachten ist dabei jedoch, daß eventuell der Schleifenkörper nicht ausgeführt wird.

Die Schleifenvariable darf eine Variable vom Typ INTEGER, REAL oder DOUBLE PRECISION sein. Der Anfangswert, der Endwert und die Schrittweite dürfen beliebige Ausdrücke vom Typ INTEGER, REAL oder DOUBLE PRECISION sein, insbesondere auch negativ oder Null.

Die Anzahl der Schleifendurchläufe errechnet sich z.B. bei der

DO-Anweisung

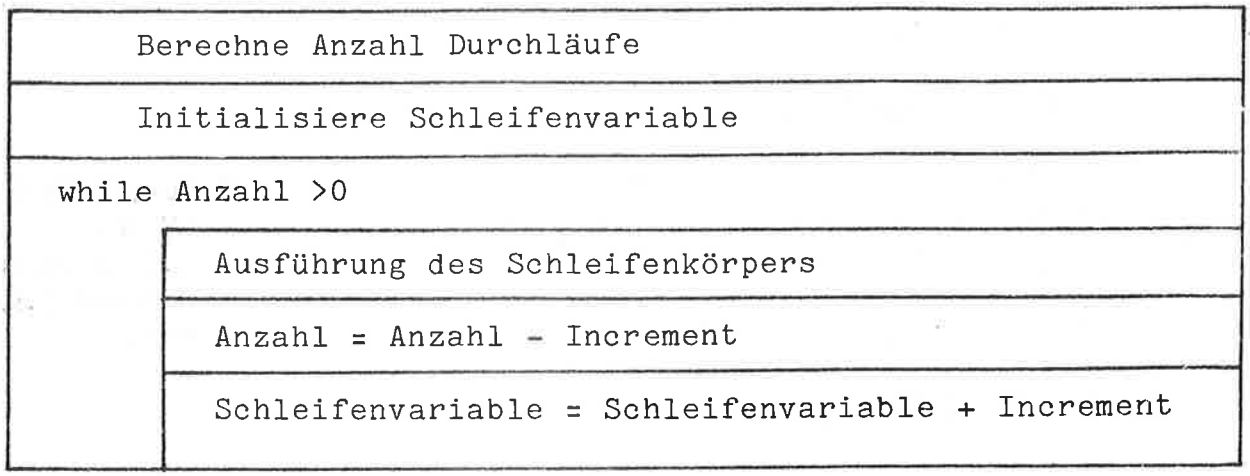
```
      DO 100, v=e1,e2,e3
```

nach der Formel

```
      MAX(INT((m2-m1+m3)/m3),0)
```

wobei m1, m2 bzw. m3 die Werte der Ausdrücke e1, e2 bzw. e3 sind. Es ist somit möglich, daß eine DO-Schleife nullmal ausgeführt wird.

Das folgende Struktogramm zeigt die Ausführung einer DO-Schleife.



Demnach hat nach Beendigung der Schleife die Schleifenvariable den Wert, den sie beim nächsten Durchlauf hätte, in dem Beispiel

```
      DO 100, J = 1, 8, 2
      K=J
100   CONTINUE
```

ist somit nach Beendigung der Schleife K=7 und J=9.

c) Ein- und Ausgabe

Die Ein- und Ausgabe ist eines der Gebiete, bei denen eine Standardisierung sehr schwierig ist, da hier naturgemäß Bezüge zu den Betriebssystemen der einzelnen Anlagen bestehen, die durch eine maschinenunabhängige und portable Programmiersprache nur unzulänglich beschrieben werden können. In FORTRAN 77 wurden viele Operationen eingebaut und damit standardisiert, die bisher auch schon in sehr unterschiedlicher Art in den einzelnen FORTRAN 4 Dialekten möglich waren.

Folgende Operationen und Möglichkeiten wurden in FORTRAN 77 nun standardisiert:

- Direkter (Random) Zugriff auf Dateien
- Default-Wert für Format
- Default-Wert für die logische Gerätenummer
- Öffnen und Schließen von Dateien
- Benutzung von CHARACTER-Variablen als Dateien
- Reaktion auf Fehler
- Reaktion auf Dateiende

1) Spezifikatoren

In der READ- bzw. WRITE-Anweisung sind Spezifikatoren zugelassen, und zwar:

UNIT	für die logische Gerätenummer,
FMT	für das Format ,
REC	für die Satznummer bei Random-Zugriff,
IOSTAT	zur Ablage des Errorcodes,
ERR	für den Error-Label und
END	für den End-Label.

Bei Verwendung dieser Spezifikatoren dürfen die Spezifikationen zu einer READ- bzw. WRITE-Anweisung in beliebiger Reihenfolge notiert werden. Damit FORTRAN 4 Anweisungen gültig bleiben, dürfen UNIT und FMT weggelasen werden, wenn dies die beiden ersten Spezifikationen in der Anweisung sind. Die folgenden Anweisungen haben alle die gleiche Bedeutung:

```
      READ(10,66,REC=I+2,IOSTAT=K,ERR=98,END=99) A
      READ(UNIT=10,FMT=66,IOSTAT=K,ERR=98,
     +END=99,REC=I+2) A
      READ(REC=I+2, IOSTAT=K,ERR=98,END=99,
     +UNIT=10,FMT=66) A
```

2) Formate und Formatcodes

Formate in READ- bzw. WRITE-Anweisungen dürfen in FORTRAN 77 in folgenden Formen angegenen werden:

- als Marke einer FORMAT-Anweisung,
- als INTEGER-Variable, der mittels einer ASSIGN-Anweisung eine Marke zugewiesen wurde oder
- als CHARACTER-Ausdruck bzw. CHARACTER-Konstante.

Die folgenden WRITE-Anweisungen haben somit identische Wirkung:

```
      WRITE(10,7)'MITTELWERT: ',(X+Y)/2
    7 FORMAT(A,F7.2)

      INTEGER FMT
      ASSIGN 7 TO FMT
      WRITE(10,FMT)'MITTELWERT: ',(X+Y)/2
    7 FORMAT(A,F7.2)

      CHARACTER FORM*8
      FORM='(A,F7.2)'
      WRITE(FMT=FORM,UNIT=10)'MITTELWERT: ',(X+Y)/2

      WRITE(10,'(A,F7.2)')'MITTELWERT: ',(X+Y)/2
```

Zur Beschreibung der Ein- und Ausgabedaten wurden folgende Formatcodes hinzugefügt:

Iw.d	mindestens d Ziffern
Ew.dEe	genau e Ziffern im Exponenten
Aw	CHARACTER Daten
A	CHARACTER Daten, die Feldweite w wird durch die Länge der Daten bestimmt
Tc	Tabulator auf Spalte c
TLc	Tabulator links um c Stellen
TRc	Tabulator rechts um c Stellen
S,SP,SS	PLUS-Zeichen als Vorzeichen ist optional, immer gedruckt, immer unterdrückt.
BN	Leerstellen werden ignoriert (bei Eingabe)
BZ	Leerstellen werden wie Nullen behandelt (bei Eingabe)
:	Folgende H-Format-Texte werden nur ausgegeben, wenn noch Variablen ausgegeben werden sollen.

3) Interne Dateien

Eine CHARACTER-Variable bzw. ein Feld von CHARACTER-Variablen können als interne Datei verwendet werden. Damit können die ENCODE- DECODE-Möglichkeiten der Cyber realisiert werden. Das folgende Beispiel zeigt eine interessante Verwendung dieser Möglichkeiten. Dabei steht auf jeder eingelesenen Lochkarte als erstes Zeichen der Formatcode mit dem die folgende Konstante eingelesen werden soll. Das Programm sieht so aus:

```
CHARACTER BUFFER*80
READ(UNIT=6,'(A80)') BUFFER
IF (BUFFER(1:1).EQ.'I') THEN
   READ(UNIT=BUFFER,FMT='(T2,I3)') I
 ELSE IF (BUFFER(1:1).EQ.'F') THEN
   READ(UNIT=BUFFER,FMT='(T2,F3.1)') X
 ELSE
   CALL INP ERR
END IF
```

Die Eingabedaten haben folgende Form:

```
I123
F3.9
I-75
F0.5
F-.3
```

4) Die OPEN-Anweisung

Mit der OPEN-Anweisung wird ein Bezug hergestellt zwischen einer logischen Gerätenummer und einer Datei. Gleichzeitig können mit der OPEN-Anweisung Eigenschaften der Datei verifiziert werden.

Mit der Anweisung

```
OPEN(10,NAME='JOAN',ACCESS=DIRECT)
```

wird der Bezug zwischen der Datei mit dem Namen JOAN und der logischen Gerätenummer 10 hergestellt, gleichzeitig wird spezifiziert, daß auf die Datei direkt zugegriffen werden soll. Die Anweisung

```
OPEN(NAME=JOAN,STATUS='OLD',UNIT=10,ERR=99)
```

hat den gleichen Effekt. Zusätzlich wird geprüft, ob JOAN ein alte, d.h. schon existierende Datei ist, falls nein erfolgt ein Sprung auf die Anweisung mit der Marke 99.

In der OPEN-Anweisung sind die folgenden Spezifikatoren zugelassen:

UNIT	logische Gerätenummer
FILE	Dateiname
STATUS	'OLD'
	'SCRATCH'
	'NEW'
	'UNKNOWN'
ACCESS	'SEQUENTIAL'
	'DIRECT'
FORM	'FORMATTED'
	'UNFORMATTED'
RECL	Satzlänge in Dateien mit Random-Zugriff
BLANK	'NULL' Blankunterdrückung bei Eingabe
	'ZERO' Blanks als Null interpretiert
ERR	Fehlermarke
IOSTAT	INTEGER-Variable zur Kennzeichnung des Dateiendes

5) Die INQUIRE-Anweisung

Mit der INQUIRE-Anweisung kann der Status einer Datei während eines Programmlaufes abgefragt werden. Dabei wird die Datei entweder über den UNIT-Spezifikator oder über den FILE-Spezifikator ausgewählt. Die weiteren Spezifikatoren dienen zur Bezeichnung der gewünschten Statusabfragen. Die folgende Tabelle zeigt die möglichen Statusabfragen:

Spezifikator	Typ	Wirkung
EXIST	LOGICAL	zeigt an, ob die Datei existiert
OPENED	LOGICAL	zeigt an, ob die Datei eröffnet wurde
NUMBER	INTEGER	gibt bei eröffneter Datei die zugehörige logische Gerätenummer an
NAME	CHARACTER	gibt den Dateinamen an, falls es sich um eine benannte Datei handelt
ACCESS	CHARACTER	gibt bei eröffneter Datei den Zugriffstyp an
SEQUENTIAL	CHARACTER	gibt an, ob auf die Datei sequentiell zugegriffen wird
DIRECT	CHARACTER	gibt an, ob auf die Datei random zugegriffen wird
FORM	CHARACTER	"FORMATTED´ oder ´UNFORMATTED´
FORMATTED	CHARACTER	gibt ab, ob die Datei formatiert bearbeitet wird
UNFORMATTED	CHARACTER	gibt an, ob die Datei unformatiert bearbeitet werden kann
BLANK	CHARACTER	gibt an, wie Blanks verarbeitet werden
ERR	LABEL	Fehlerausgang
RECL	INTEGER	gibt die Satzlänge an
NEXTREC	INTEGER	gibt die Nummer des nächsten Satzes
IOSTAT	INTEGER	zeigt die Fehlerart an, falls ein Fehler aufgetreten ist

d) Prozeduren

Die wesentlichen Änderungen bei Prozeduren betreffen die folgenden Punkte:
- INTRINSIC FUNCTIONS
- die ENTRY-Anweisung
- die SAVE-Anweisung

1) INTRINSIC FUNCTIONS

Alle Funktionen, die in FORTRAN 4 in die Klassen INTRINSIC und BASIC EXTERNAL eingestuft wurden, sind in FORTRAN 77 in der Klasse der INTRINSIC FUNCTIONS zusammengefaßt. Es gibt also nicht mehr die Unterscheidung zwischen BASIC EXTERNAL und INTRINSIC FUNCTIONS. Innerhalb der INTRINSIC FUNCTIONS wurden wiederum Klassen gebildet und diesen Klassen wurde ein allgemeiner Name zugeordnet. Dieser allgemeine Name (generic name) kann beim Aufruf einer jeden Funktion dieser Klasse verwendet werden. Die Funktion selbst wird dann durch den Typ der Argumente bestimmt.

Beispiel: Eine Klasse bilden alle Funktionen, die einen absoluten Wert berechnen, also die Funktionen IABS, ABS, DABS und CABS. Der allgemeine Name dieser Klasse ist ABS.
Mit den Deklarationen

```
      INTEGER I
      REAL R
      DOUBLE PRECISION D
      COMPLEX C
```

sind die folgenden Funktionsaufrufe gleichwertig:

ABS(I) und IABS(I)
ABS(R) und ABS(R) (hier gibt es nur einen Namen)
ABS(D) und DABS(D)
ABS(C) und CABS(C)

2) Die ENTRY-Anweisung

Mit der ENTRY-Anweisung ist es möglich in einer Prozedur mehrere Eingänge zu definieren. Diese Möglichkeit war bisher in den meisten FORTRAN 4 Dialekten auch schon gegeben, standardisiert war diese Möglichkeit jedoch noch nicht.

3) Die SAVE-Anweisung

Die SAVE-Anweisung ist eine Deklarationsanweisung. Mit ihr ist es möglich, in einer Prozedur lokale Variablen so zu deklarieren, daß der Wert dieser Variablen zwischen zwei Prozeduraufrufen erhalten bleibt. Das folgende Beispiel soll dies erläutern:

```
      SUBROUTINE RANDOM(X)
      INTEGER SEED
      SAVE SEED
      X= ... SEED ...
      SEED= ... SEED ...
     RETURN
C
      ENTRY SET(N)
      SEED=N
      END
```

Regionales Rechenzentrum Erlangen	**RRZE** Programmbibliothek TR44Ø	PASCAL Bearb.: M. Abel Datum: Januar 1979

PASCAL-Compiler für den TR44Ø

Aufruf:

```
□BIBANMELDE,BIBLIOTHEK=PASCAL
           ,TRAEGER=LFD
□UEBERSETZE,...
           ,SPRACHE=PASCAL
           ,...
```

Achtung: Der Aufruf des Compilers hat sich gegenüber der Version vom März 1978 geändert.

Literatur:

K. Jensen, N. Wirth
PASCAL User Manual and Report
Lecture Notes in Computer Science No. 18

PASCAL-Übersetzer MV 36

I. Die Sprache PASCAL

A. Spracherweiterungen

A.1 Vorübersetzte Prozeduren

Es ist möglich, vorübersetzte Programmteile anzuschließen. Die Deklaration eines solchen als Prozedur oder Funktion besteht aus <procedure heading> mit vollständiger Parameterliste und dem Wortsymbol extern als <block>.
Die Deklaration muß im <procedure and function declaration part> des Hauptprogramms erfolgen, sonst meldet der Übersetzer einen Fehler.

Um auch Unterprogramme aus anderen Sprachen anzuschließen, kann nach extern noch die Anschlußkonvention spezifiziert werden. Dabei werden folgende <identifier> als Sprachen erwartet:

PASCAL | FORTRAN | ALGOL

Als optionaler Wert wird PASCAL eingesetzt.
Bei ALGOL bzw. FORTRAN lassen sich von PASCAL nur folgende Datentypen als Parameter übergeben:

- integer, real, boolean
- char als integer (ord(ch))
- selbstdeklarierte Skalare als Ordnungszahl
- eindimensionale Felder, deren Komponenten den obigen Datentyp haben.

A.2 Der Datentyp file

Die Deklaration von file-Typen ist fast uneingeschränkt erlaubt. Eine Variable bzw. die Komponente einer Variablen vom Typ file wird auf eine Datei des BS3 abgebildet. Aus diesem Grunde sind folgende Einschränkungen getroffen:

- Komponenten eines file-Typs dürfen nicht vom Typ file sein (kein file of file)
- in einer dynamisch generierten Variablen darf keine Komponente vom Datentyp file sein.

Durch die Satzstruktur der Dateien, die das BS3 ermöglicht, sind folgende Definitionen für files getroffen:

- beim Datentyp file wird jede Komponente in einen Satz abgelegt
- beim Datentyp packed file werden beliebig viele Komponenten in einem Satz abgelegt; der Satzabschluß muß vom Benutzer selbst durch Aufruf der Standard-Prozedur PUTLN erzeugt werden.

Der Standardtyp text (siehe D.2) wird auf eine Texthaltungsdatei des BS3 abgebildet, d.h. sie ist mit den Texthaltungskommandos der Kommandosprache zu bearbeiten. Für alle files wird beim Betreten der Prozedur oder des Hauptprogramms ein reset-Aufruf erzeugt, außer für printer und output, die zum Schreiben eingestellt werden (siehe D.4 file-Prozeduren).

B. Semantik

B.1 <program-statement>

Vor jedem PASCAL-Programm muß das <program-statement> stehen, dabei kann der <identifier> fehlen und wird dann durch STDHP ersetzt.
Als <program parameter> sind nur <file identifier> zugelassen. Diese <file identifier> müssen als Variable im Hauptprogramm deklariert werden (außer den Standardfiles (siehe D.3)).

Ein * hinter dem <file identifier> zeigt an, daß auf den file nur lesend zugegriffen werden darf. Ein schreibender Zugriff auf diesen file führt zu einem Laufzeitfehler.

Den <file identifier> werden vom Übersetzer Zahlen zugeordnet (Kanalnummerzuordnung), die für die Zuordnung von Dateien benutzt werden können (siehe II.B.).

Wird auf die Standardfiles (siehe D.3) zugegriffen, so müssen diese als <program parameter> auftreten. Werden sie nicht aufgeführt, so sind die Variablen dem Übersetzer nicht bekannt, und die Benutzung führt zu einer Fehlermeldung des Übersetzers.

B.2 Vorübersetzte Prozeduren

Man kann PASCAL-Prozeduren oder Funktionen als Unterprogramme vorübersetzen. Dazu müssen folgende Formvorschriften erfüllt sein:

- das <program statement> muß fehlen
- im Hauptprogramm sind nur <constant definition part> <type definition part> neben <procedure and function declaration part> zugelassen
- der <block> des Hauptprogramms muß leer sein, d.h. nach dem letzten Semikolon folgt sofort ein Punkt
- auf die Standardfiles darf nicht zugegriffen werden.

Auf alle Prozeduren und Funktionen, die in dem <procedure and function declaration part> des Hauptprogramms deklariert werden, kann von anderen Programmen über extern zugegriffen werden.

Erstellt man mehrere vorübersetzte Objekte, die zusammen benutzt werden sollen, so müssen die Namen eindeutig sein.

Für die Verträglichkeit der Parametertypen beim Aufruf von vorübersetzten Prozeduren muß der Benutzer selbst Sorge tragen, da keine Tests während der Laufzeit gemacht werden.

C. Einschränkungen

1. Die Wortsymbole extern und value sind reserviert.
2. Der Basistyp einer Menge (set) darf entweder
   a) ein deklarierter Skalartyp mit höchstens 624 Elementen oder einer Teilmenge daraus
   oder

b) ein Subrange, dessen untere Grenze ≥0 und dessen obere Grenze ≤ 623 ist, sein.

3. Ein <string> darf maximal 78 Zeichen lang sein.
4. Es ist nicht möglich, einen Typ file of file zu konstruieren; jedoch sind record bzw. array mit file-Komponenten erlaubt. Auch sind file-Komponenten, die durch new-Aufrufe erzeugt werden, verboten.

   Innerhalb einer rekursiven Prozedur bzw. Funktion sollte keine file-Komponente auftreten, da für jede file-Komponente statisch ein Name vergeben wird. Die Zahl dieser Namen darf 40 nicht überschreiten, sonst meldet der Übersetzer einen Fehler.

D. Standardnamen und ihre Anwendung

D.1 Konstanten

maxint = 34359738368;	größte integer-Zahl, die sich aus dem Runden einer real-Zahl erhalten läßt.
alfaleng = 6;	siehe alfa
eol = '*021'	neue Zeile bei der Ausgabe

D.2 Types

```
text = packed file of char;
alfa = packed array [1..alfaleng] of char;
```

Für Ergebnisse von function ist der Typ alfa zugelassen.

D.3 Variable

```
var  input, output: text;
```

Bei der Benutzung der Standardfiles müssen diese als <program parameter> auftreten. Wird keine Zuordnung im STARTE-Kommando (siehe II.B.) vorgenommen, so werden folgende Standardzuordnungen gemacht, abhängig in welchem Modus das Programm gestartet wird:

a) Abschnittmodus

   Der unter der Spezifikation DATEN des STARTE-Kommandos zugeordnete Fremdstring wird als input zugegriffen. Output ist das Ablaufprotokoll.

b) Gesprächsmodus

Ist die Spezifikation DATEN des STARTE-Kommandos besetzt, so wird dorther über input gelesen. Sonst wird von der Konsole beim Zugriff auf input eine Eingabe angefordert bzw. noch vorhandene Eingabe weiterverarbeitet. Durch die Eingabe von #064 wird das Prädikat eof(input) auf true gesetzt.
Output ist das Konsolprotokoll.

printer: text;

Diesem Ausgabefile wird das Ablaufprotokoll zugeordnet. Im Gesprächsmodus muß dazu das Ablaufprotokoll eingeschaltet sein, sonst wird eine leere Leistung erbracht.

D.4 Prozeduren und Funktionen

Prozeduren

mark(i) In die integer Variable i wird der momentane heap-Zeiger gespeichert (new-Speicher).

release(i) Der heap- Zeiger wird auf den Wert aus der integer Variablen i gesetzt. Die folgenden new-Aufrufe überspeichern eventuell vorher angelegte Daten. Außerdem wird überflüssiger Speicher freigegeben.

Anmerkung: Der Wert ist keine integer Größe!

new(p:k) Zeigt p auf ein array oder ist die letzte Komponente des records ein array, so gibt k die obere Grenze (der ersten Dimension) an. Es wird für dieses array soviel Speicher angefordert wie k als obere Grenze erfordert.
Es erfolgt keine Prüfung, ob die statische Grenze oder die dynamische Grenze k eingehalten werden.

new(p,t_1...,t_n)

Es wird nur der Speicher für die durch t_i gesteuerten Varianten angelegt. Gleichzeitig werden die tagfield identifier mit den angegebenen Werten besetzt. Da nur der minimale Speicher angefordert wird, sollten die tagfields nicht mit anderen Werten besetzt werden, um ein Überschreiben anderer Daten zu verhindern.

new(p,t_1,...,t_n:k)

Ist die letzte Komponente der durch die Konstanten t_1,...,t_n angesteuerten Variante ein array, so wird für dieses array k als oberste Grenze der ersten Dimension eingesetzt. Weiterhin gilt das oben Gesagte.

dispose(p) bzw.
dispose(p,t_1,...,t_n) erbringen eine leere Leistung.

date (a) Der Variablen a vom Typ alfa wird das Tagesdatum in der Form jjmmtt zugewiesen.

time (a) Der Variablen a vom Typ alfa wird die Uhrzeit in der Form hhmmss zugewiesen.

ddate (b) Der Variablen b vom Typ packed array [1..8] of char wird das Tagesdatum in der Form tt.mm.jj zugewiesen.

file Prozeduren

reset(f) wie im Report definiert

reset(f,n) Ist dem file f eine Random-Datei zugeordnet (siehe II.B.), so wird zum Lesen auf den Satz n positioniert. Ist der Satz n nicht vorhanden, so wird auf den folgenden Satz positioniert.
n = 0 entspricht reset(f).
Ist f ein packed file, so ist das Prädikat eoln true, die Komponente f↑ ist aber undefiniert.

rewrite(f) wie im Report definiert

rewrite(f,n)

Ist dem file f eine Random-Datei zugeordnet (siehe II.B), so wird zum Schreiben auf Satz n positioniert. Ist ein Satz n vorhanden, so wird dieser durch put/putln überschrieben.
n = 0 entspricht rewrite(f).
n = -1 heißt hinter den letzten Satz positionieren zum fortlaufenden Schreiben.

getln(f) bzw.
putln(f) f muß Typ packed file sein.
Es wird auf den Beginn des nächsten Satzes fortgeschaltet. Beim Lesen wird auf die erste Komponente des Satzes zugegriffen. Beim Schreiben wird ein Satzabschluß erzeugt und auf den nächsten Satz positioniert. Beim Lesen wird aber auch automatisch auf den nächsten Satz fortgeschaltet, falls die Information eines Satzes abgearbeitet ist. Das Satzende wird gemeldet (eoln = true), und die zugegriffene Komponente ist undefiniert.

page(f) f muß eine Ausgabedatei sein. Es wird ein Steuerzeichen für einen Seitenvorschub abgelegt.

Funktionen

card (s) liefert die Mächtigkeit der Menge s (s kann auch ein Ausdruck mit dem Ergebnistyp set sein) als integer Wert an.

clock (i) liefert in Abhängigkeit von i die verschiedenen Maschinenzeiten als integer Werte an.

i = 3 : Laufzeit des Abschnitts
4 : Maschinenzeit (Abschnitt/Operator unabhängig)
5 : Laufzeit des Operators
6 : Transportindex des Abschnitts
7 : Transportindex des Operators
8 : Abschnittsrechenzeitschranke
9 : Operatorrechenzeitschranke
sonst : Der Wert 0 wird angeliefert.

file Funktionen

eof(f) zeigt das file-Ende an. Beim schreibenden Zugriff auf einen file muß eof(f) = true sein, sonst führt ein Zugriff auf einen Laufzeitfehler.

eoln(f) zeigt an, ob beim packed file das Satzende beim Lesen erreicht wurde. In dem Fall ist die Komponente undefiniert. In den anderen Fällen hat eoln(f) keine Bedeutung.

linenumber(f)

liefert die aktuelle Satznummer als vom Typ integer ab, ist allerdings kein Wert vom Typ integer.

II. Benutzung des PASCAL-Übersetzers auf der TR440

A. Übersetzung

Der Übersetzer wird durch das UEBERSETZE-Kommando mit SPRACHE=PASCAL gestartet. Von den weiteren Spezifikationen werden ausgewertet:

QUELLE:	wie beschrieben
NUMERIERUNG:	es wird -STD- erwartet
MO:	wie beschrieben
VARIANTE:	es werden D, GS, KV ausgewertet
PROTOKOLL:	es werden -,-STD-, A,R,O,KO ausgewertet
DYNKON:	-STD- bedeutet, daß während der Übersetzung das statische Einhalten der Grenzen von Subranges geprüft wird. Es werden aber keine dynamischen Kontrollen eingebaut.
TRACE:	-STD- wird ausgewertet. Es werden dann die Prozeduraufrufe im Ablaufprotokoll festgehalten.
BEREICH:	-,a wird ausgewertet. Bei der Angabe a-b wird b ignoriert.

B. Rechnen

Nach der üblichen Montage (siehe Kommando MONTIERE) wird das Programm mit STARTE gestartet. Dabei werden den Spezifikationen DATEN und DATEI entsprechende Werte zugeordnet.

DATEN: Daten für den file input

DATEI: Zuordnung von Dateien zu files über die Kanalnummer oder über (file-identifier) wie bei PL/I

Beispiel:

```
        program (input, output, f1, f2);
                      .
                      .
                      .
        kanalnummerzuordnung
        f2                    4
        f1                    3
        output                2
        input                 1
      #STARTE, DATEI = (INPUT)-EINGABE'3-TEST   bewirkt,
```

daß über input auf eine Datei EINGABE zugegriffen wird, output hat die normale Zuordnung, und der file f1 spricht die Datei TEST an. Ist keine Datei F3 vorhanden, so wird eine in der Standarddatenbasis kreiert.

B.2 Laufzeitfehleranalyse

Eine Laufzeitfehleranalyse ist zur Zeit nur bedingt möglich, ein sprachabhängiger Dump wird noch nicht ausgegeben. Wird die Spezifikation DUMP des STARTE-Kommandos auf DUMP=A-NICHTS gesetzt, so wird eine Rückverfolgung versucht. Die Adressen lassen sich dann auswerten, wenn das Objekt mit PROT.=A übersetzt wurde, Die relativen Startadressen der Prozeduren sind sedezimal ausgegeben, so daß die vom Rückverfolger ausgegebenen Adressen dort eingeordnet werden können.

C. Symbolerweiterungen

Neben den Grundsymbolen des Reports werden folgende Zeichen erkannt:

a) Kommentarklammern:

```
/*      */     dabei darf */ nicht im Kommentar vorkommen
(*      *)     dabei darf*) nicht im Kommentar vorkommen
{        }     dabei darf } nicht im Kommentar vorkommen
```

b) Ersatzdarstellungen:

```
(. oder (/  für  [
.) oder /)  für  ]
.,          für  ;
@ , | oder  -> für ↑
```

c) Operatoren:

```
Mengenvereinigung :   or,  v, +
Mengendurchschnitt:   and, ^, *
```

III. TR440 PASCAL Interpretation

A. Standarddatentypen

A.1 integer

Festkommazahl einfacher Wortlänge mit TK 1

Es muß gelten $-2^{46} < i < 2^{46}$

Im Bereich -10^{13} .. $+10^{13}$ lassen sich die Zahlen drucken, andernfalls werden maximal zwölf * ausgegeben. Im Bereich -maxint..maxint können sie als Konstante vereinbart werden.

A.2 real

Gleitpunktzahl einfacher Wortlänge mit TK 0

Es muß gelten

$$7.458341 \cdot 10^{-155} \leq |X| \leq 8.37988 \cdot 10^{152} .$$

A.3 char

Ein Zeichen in der Darstellung des ZC1 rechtsbündig im Wort mit TK 1. Läßt sich das Zeichen nicht ablochen, so kann es mit Hilfe der Ersatzdarstellung abgelocht werden:

'*$z_1z_2z_3$' wobei $z_1z_2z_3$ die dreistellige dezimale Ordnungszahl im ZC1 ist.

A.4 boolean

false = 0 und true = 1 jeweils rechtsbündig im Wort mit TK 1

A.5 selbstdeklarierte Skalare

werden als Festkommazahl rechtsbündig mit TK 1 abgelegt. Die Zahlen sind aus dem Bereich 0..n-1, wenn n die Anzahl der Elemente des skalaren Typs ist.

A.6 Pointer (Zeiger)

werden als Festkommazahlen halber Wortlänge dargestellt, die jeweils im linken Halbwort abgelegt werden.

A.7 set-Werte

werden als Bitketten dargestellt, die von links nach rechts in den benötigten Wörtern mit TK 2 abgelegt werden. Die Bitkette ist das Bild der charakteristischen Funktion der Menge:

$$F(x) = \begin{cases} L & \text{falls } x \in M, \\ 0 & \text{falls } x \notin M, \end{cases} \qquad \text{ord}(x) \in [0..n]$$

A.8 record/array

Die einzelnen Komponenten werden jeweils linear hintereinander abgelegt. Im record wird allerdings nicht unbedingt die angegebene Reihenfolge der Felder beibehalten. Ein mehrdimensionales array wird immer zu einer Kette von eindimensionalen Feldern aufgebrochen; die Ablage erfolgt somit spaltenweise.
Beispiel:

```
array [1..n,1..m] of integer   wird aufgefaßt als
array [1..n] of array [1..m] of integer
```

Wird vom Programmierer die gepackte (packed) Ablage gefordert, so wird versucht, mehrere Komponenten, die zu ihrer Darstellung weniger als ein Wort Platz benötigen, von links nach rechts in einem Wort abzulegen. Dabei wird keine Bit-Position bevorzugt.

Beispiel:

für die verschiedene Strukturierung eines TR440 Wortes

```
wort = packed record
         case integer of
         1:(i: integer);
         2:(r: real);
         3:(a: alfa);
         4:(s: set of 0..47);
         5:(b: packed array [1..48] of boolean)
       end;
```

A.9 file

Zu jedem file wird ein Beschreibungsblock von 16 GW Länge aufgebaut, der sowohl die file-Prädikate als auch die Information für das Betriebssystem enthält. Anschließend wird der Platz für die Komponente (file buffer) bzw. bei gepackten files der Platz für einen Satz reserviert.
Der genaue Aufbau ist in einer internen Dokumentation beschrieben.

B. Die Standardprozedur write

Ist bei der Benutzung der write-Prozedur kein Feldlängenparameter angegeben, so werden folgende Standardwerte eingesetzt:

Typ des Wertes	Feldlänge
integer	10
real	20 (ø±D.DDDDDDDDDDDE±DDD)
boolean	5 (øTRUE oder FALSE)
char	1
<string>	Länge des Strings
alfa	6 (alfaleng)

Bei real-Werten wird bei kürzerer Feldlänge als 20 nur die Mantisse verkürzt (minimale Feldlänge 10). Wird bei boolean eine kürzere Feldlänge als 5 angegeben, so wird nur T oder F rechtsbündig ausgegeben. Paßt ein Wert nicht in das angegebene Format, so wird das Standardformat bzw. das für die Größe des Wertes nötige Format genommen. Ist die Feldlänge ≤ 0, so wird eine leere Leistung erbracht.

Ist der Wert des char ein Steuerzeichen (ord(ch) < 64), so wird der Satz abgeschlossen und auf den nächsten Satz positioniert (Wirkung wie putln/writeln). Ist dem file eine Ausgabedatei zugeordnet oder das Ablaufprotokoll, so wird das Zeichen als erste Oktade abgelegt, sonst unterbleibt die Ausgabe.

Unabhängig vom Typ des Wertes kann ein Wort sedezimal eingeschlossen in Apostroph ausgegeben werden. Als Feldlänge ist 'H' (char) anzugeben; wird die zweite Feldlänge gesetzt, so wird dieser Wert als Format benutzt. In dem Fall werden die Apostrophe weggelassen und führende Nullen durch ø (blank) ersetzt.

C. Prozeduranschluß und Parameterversorgung

C.1 Allgemeine Information

Beim Aufruf einer externen Prozedur bzw. Funktion werden folgende Werte in den Registern übergeben:

$\langle RA \rangle_{25..48}$	::=	Adresse des Parameterbeschreibungsblocks
$\langle RQ \rangle_{25..48}$	::=	vor dem Aufruf eingestellte Indexbasis
$\langle RH \rangle_{25..48}$	::=	dynamisches Niveau des statischen Vorgängers (für TAS-Unterprogramme ohne Bedeutung)
<BB>	::=	Rückkehradresse

C.2 Verwendung von Indexregistern

Jede Aktivierung einer PASCAL-Prozedur besitzt eine eigene Indexbasis. Die Verwendung der Indexregister 0..17 ist festgelegt und wird auch für die Rückverfolgung im Fehlerfall benötigt.

0..4	werden freigehalten (Programmiersystemkonvention)
5,6,7	werden für Teilwort- oder Multiplezugriffe benutzt (sind aber frei, werden evtl. zerstört)
8	Rückkehradresse im Normalfall
9	Beginn des freien Speichers vor Aufruf der Prozedur
10	Stackpointer (wird zum Adressieren von Hilfsvariablen benötigt)
11	Adresse des Parameterbeschreibungsblocks
13	Beginn des freien Speichers nach Aufruf der Prozedur
14	letzte Indexbasis } werden beim
15	letzter Unterprogrammordnungszähler } BCI getauscht
16	aktuelle Basisadresse
17	Adresse des statischen Vorgängers
18 ff	Rückkehradresse für SU/SUE-Unterprogrammsprünge

Der Unterprogrammordnungszähler wird durch Umstellen der Indexbasis auf U_o= 17 eingestellt.

C.3 Parameterbeschreibungsblock

	24	4		8
TK2	FA	SS		PZ
TK2	ADR1	ZV1		
. .	. .	. .		
TK2	ADRn	ZVn		

FA : Fehlerausgang (siehe z.B. altes TAS-Handbuch D.8; D.9)

SS : Sprachschlüssel

PZ : Parameterzahl

ADRi : Relativadresse des i-ten Parameters bezüglich der aktuellen Basisadresse

ZVi : Zusatzversorgung zur Beschreibung des i-ten Parameters (siehe C.6)

C.4 Aufruf einer PASCAL-Prozedur

Beim Betreten einer PASCAL-Prozedur oder -Funktion werden die Registerinhalte in die Verwaltungsindexregister 8-17 bezüglich der für diese Prozedur neu einzustellenden Indexbasis abgelegt. Außerdem wird Speicher für lokale Variable, Hilfsspeicher und Parameterplätze für aufzurufende Prozeduren angefordert.

In der MAKRO-Bibliothek PASCAL.PC&MACLIB stehen für die nötigen Befehlsfolgen Makros zur Verfügung:

PROCEXTERN

Dieses Makro ist einmal in einem Montageobjekt aufzurufen. Es deckt nötige Externbezüge ab und benennt die Indexzellen in folgender Form

```
INDEX 5(XC.,XL.,XR.,1,XGBFS.,XSTP.,XLBFS.,2,XHIER.)
```

Auf diese Namen wird in den anderen Makros Bezug genommen.

PROCENTRY(PARLC=0, FSP=22, HSP=0)

Dieses Makro enthält die Eingangsbefehle. Dabei sind die Makro-Parameter folgendermaßen zu besetzen:

PARLC gibt die Anzahl der durch Parameter belegten Adreßstellen an. Bei der Angabe PARLC=VAR wird die Anzahl aus dem Parameterbeschreibungsblock berechnet.

FSP gibt die Zahl der Adreßstellen für die lokalen Variablen an. Die Verwaltungsindexzellen müssen dazu gezählt werden, d.h. FSP ist mindestens 18.

HSP gibt die Zahl der Adreßstellen für den Hilfsspeicherbereich an. XSTP zeigt auf das letzte Ganzwort vor diesem Bereich (siehe EZ-Befehl).

Die Makro-Parameter sind so eingestellt, daß eine Prozedur ohne Parameter und ohne lokale Variable und ohne Hilfsspeicher aufgerufen wird.

PROCEXIT(FW=NEIN)

Dieses Makro umfaßt die Prozedurausgangsbefehle. FW ≥ 0 gibt die Adresse bezüglich der aktuellen Basisadresse an, aus der ein Funktionswert in <RA> geladen wird. Wird FW nicht besetzt, so wird ein Prozedurausgang erzeugt, wobei die Befehlsfolge <RA> nicht zerstört.

Die in den Makros zusammengefaßten Befehlsfolgen sind kein Muß; sie sollen nur das Schreiben von externen Prozeduren in TAS erleichtern.

C.5 Zugriffe auf die Parameter

Die Parameter werden im Speicher in der Reihenfolge ihrer Spezifikation abgelegt. Dabei werden folgende Werte übergeben:

call-by-value	der Wert steht im Speicher (bei multiples werden alle Werte in ihrer Reihenfolge abgelegt)
call-by-reference	die absolute Adresse steht linksbündig im Ganzwort

call-by-procedure in einem Ganzwort stehen links die Startadresse und rechts das dynamische Niveau des statischen Vorgängers.

Die Parameter liegen jeweils vor dem Speicherbereich der aktuellen Prozedur und lassen sich über die Indexzelle XHIER adressieren.

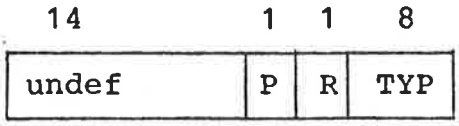
Beispiel: M XHIER, MAB B -k,

Ist die Parameterbelegung nicht bekannt, so kann man die Adresse aus dem Beschreibungsblock übernehmen:

Beispiel: M 11, TCB k*2:k-ter Parameter; MRX B XHIER,
: mod2 und BB enthalten die absolute Adresse;

Diese Zugriffe sind nur dann möglich, wenn der Standardaufruf erfolgt.

C.6 Aufbau der Zusatzversorgung



R : call-by-reference (bit15 = L)
P : call-by-procedure (bit14 = L)
Typ ≠ 0 gibt den Typ des Funktionswertes an, d.h.
Typ = 0 heißt Prozedur.

Sind bit15 = 0 und bit14 = 0, so belegt der Parameter typabhängig verschiedene Zahl von Ganzworten, sonst wird je ein Ganzwort belegt.

TYP: kurze Typspezifikation des Parameters mit Hilfe von char-Werten beschrieben:
R = real
B = boolean
C = char
I = integer bzw. deklarierte Skalare
P = Pointer
S = set

M = record
F = file
N = eindimensionales Feld mit real-Komponenten
O = eindimensionales Feld mit boolean-Komponenten
L = eindimensionales Feld mit integer-Komponenten
D = eindimensionales Feld mit char-Komponenten
A = array (keines der Typen D, L, N, O)

Genauere Angaben über den Typ der übergebenen Werte lassen sich nicht geben, da diese Information ziemlich umfangreich werden kann.

Die Anzahlen der von den Parametern belegten Speicherplätze lassen sich aus den Adressen des Beschreibungsblocks berechnen.

D. Spezielle maschinenabhängige Sprachmittel (TR 440)

D.1 Zugriff auf die Typenkennung

Dem PASCAL-Übersetzer ist der skalare Typ

(TKO,TK1,TK2,TK3)

bekannt. Da dieser Typ keinen Namen hat, kann keine Variable und kein strukturierter Typ von diesem Typ definiert werden. Die Definition eines "subranges" von diesen Typ führt zu einer Fehlermeldung. Es können in einem PASCAL-Programm also nur Werte dieses Typs auftreten, die dann besonders behandelt werden.

D.1.1 Setzen der Typenkennung

Durch eine Zuweisung `v := TKi;` wird die Typenkennung des durch v bezeichneten Ganzwortes auf den Wert TKi aus TKO,TK1,TK2,TK3 gesetzt. Dabei kann v einen beliebigen Typ haben, es muß nur ein Ganzwort adressiert werden. Der Wert von v wird bei dem Setzen der Typenkennung nicht beeinflußt.
Durch die Zuweisung eines Wertes an v wird allerdings die Typenkennung des Wertes übernommen (siehe III.A).

D.1.2 Abfragen der Typenkennung

Durch die Abfrage `v = TKi` bzw. `v <> TKi` wird getestet ob die Typenkennung des durch v bezeichneten Ganzwortes gleich oder ungleich TKi ist. Entsprechend dem Test wird der boolsche Wert false oder true erzeugt. Für v gelten die gleichen Voraussetzungen wie unter 1.1 schon gesagt.

D.2 Angabe von Konstanten in oktaler bzw. sedezimaler Schreibweise

ES können Werte vom Typ real oder integer auch in der oben erwähnten Schreibweise noch folgender Syntax angegeben werden:

```
<unsigned constant>  ::= <octalconstant>|<sedecimal-const>
<octalconstant>  ::= <octaldigits> B <tk_wert>
<octaldigits>  ::= <octaldigit> | <octaldigits> <octaldigit>
<octaldigit>  ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
<sedecimal-const> ::= ' <sede_digits> 'X <tk_wert>
<sede_digits> ::= <sede_digit> | <sede_digits> <sede_digit>
<sede_digit>  ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F
```

<tk_wert> ::= <empty> | 0 | 1 | 2 | 3

Dabei gilt, daß eine <octalconstant> maximal aus 16 und eine <sedecimal-const> maximal aus 12 Ziffern bestehen darf. Wird die maximale Ziffernzahl nicht angegeben, so wird der Wert rechtsbündig mit führenden Nullen abgelegt. Ist der <tk_wert> leer, so wird eine 1 eingesetzt, ist der<tk_wert> gleich 0, so wird ein real-Wert angenommen ansonsten wird ein integer-Wert erzeugt.

zu I.B.2:
Bei vorübersetzten Prozeduren ist im <block> des Hauptprogramm auch der <variable declaration part> zugelassen. Diese Variable sind dann global zu den vorübersetzten Prozedurpaket, daß in dem Hauptprogramm ohne Statementteil zusammengefaßt ist. In dem <variable declaration part> dürfen keine file's deklariert werden. Außerdem müssen die Datenbereiche aller zusammengehörigen "Module" und dem Hauptprogramm nicht größer als ca. 30 K sein, da diese Bereiche in der Großseite 0 liegen müssen.

Wird bei Laufzeitfehlern eine Prozedur eines Moduls in der aktiven Kette der Prozeduraufrufe angetroffen, so werden die globalen Variablen des Moduls vor den Variablen dieser Prozedur einmal ausgegeben.

zu I.D.4:
Prozeduren

halt — Der Prozeduraufruf bewirkt den Abbruch des Programms mit der Fehlermeldung:
+++FEHLER: HALT-AUFRUF IN ZEILE <nr>
wobei <nr> die Quellzeilennummer des halt-Aufrufes angegeben wird, der den Abbruch bewirkt.
Anschließend wird wie bei einem Laufzeitfehler verfahren.

zu II.B.2:
Laufzeitfehler
Ist beim UEBERSETZE-Kommando VARIANTE=D oder KV oder GS gesetzt gewesen, so wird bei einem Fehler während der Laufzeit neben der Fehleranalyse die Bestimmung des Fehlerortes gemacht, wenn DUMP(STARTE)=A-<dumpstring> gesetzt wird. Es wird dann die dynamische Aufrufverschachtelung mit Quellzeilennummern ausgegeben. Abhängig von dumpstring werden dann die gewünschten Variablenwerte sprachabhängig ausgegeben.

Es gilt:

<dumpstring> ::= <dumpanweisung> | <dumpanweisung> (<dumpeinschränkung>)

<dumpanweisung> ::= NICHTS | NEST

<dumpeinschränkung> ::= Liste von Namen, die von der <dumpanweisung> ausgeschlossen sind.

Für weitere Information sei auf das TR 440-Kommandotaschenbuch hingewiesen, was dort für ALGOL_Dump gesagt wird gilt für PASCAL mit Einschränkungen analog.

Es gilt:

<dumpstring> ::= <dumpanweisung> | <dumpanweisung> (<dumpeinschränkung>)

<dumpanweisung> ::= NICHTS | NEST

<dumpeinschränkung> ::= Liste von Namen, die von der <dumpanweisung> ausgeschlossen sind.

Für weitere Information sei auf das TR 440-Kommandotaschenbuch hingewiesen, was dort für ALGOL_Dump gesagt wird gilt für PASCAL mit Einschränkungen analog.

Regionales Rechenzentrum Erlangen	R R Z E Programmbibliothek TR440	B M D P Bearb.: H.Cramer Datum : Jan. 79

BMDP: Biomedical Computer Programs (P-Serie, Version April 1977)

Diese BMDP-Version wurde von Herrn Koll vom Regionalen Hochschulrechenzentrum der Universität Kaiserslautern von der IBM-Version für den TR440 aufbereitet. Am Regionalen Rechenzentrum Erlangen steht BMDP über die Kommandoprozedur BIMED zur Verfügung.
Die Quellprogramme sowie die Testprogramme und Testdaten mit den dazugehörigen Ergebnissen können beim Sachbearbeiter eingesehen werden.

Diese Beschreibung erhalten Sie mit:

```
%ERZEUGE,DOKUMENT.BMDP(gerät,anzahl)
*Voreinstellungen: gerät=DR(2,0)-DC2, anzahl=1
```

Inhaltsverzeichnis

1 Aufruf der Programme

```
%ERZEUGE,SYSTEM.BMDP
%BIMED,PROGRAM,CONTROL,WORKSPACE,INPUT,SAVE,SUB,CODE,OUTPUT
```

Bedeutung der BIMED-Spezifikationen:

PROGRAM =Name des BMDP-Programms

*Voreinstellung: NOPROG (kein Programmaufruf)

name Namen der vorhandenen BMDP-Programme:

BMDP1D,BMDP2D,BMDP3D,BMDP4D,BMDP5D,BMDP6D,
BMDP7D,BMDP8D,BMDP9D,
BMDP1F,BMDP2F,BMDP3F,
BMDP1M,BMDP2M,BMDP3M,BMDP4M,BMDP6M,BMDP7M,
BMDPAM,
BMDP1R,BMDP2R,BMDP3R,BMDP4R,BMDP5R,BMDP6R,
BMDP9R,
BMDP1S,BMDP3S,
BMDP1V,BMDP2V

Namen der doppelt genauen Versionen:

P4MD zu BMDP4M, P7MD zu BMDP7M

In dieser Version nicht vorhandene Programme:

BMDP1L,BMDPAR,BMDP3V

- - - - - -

CONTROL =Angabe der BMDP-Steuerkarten (und Daten) (80 Zeichen/Karte!).

*Voreinstellung: keine Steuerkarten (und Daten)

/f oder /f%/ Fremdstring
datei Texthaltungsdatei

Dokumentation: BMDP77-Manual Seite 56ff

- - - - - -

WORKSPACE =Arbeitsspeicher für das Programm

*Voreinstellung: 2 (2046 Ganzworte)

n natürliche Zahl n mit $2 \leq n \leq 64$
Es stehen n*1024-2 Ganzworte Arbeitsspeicher zur Verfügung.

Dokumentation: BMDP77-Manual Seite 841ff

- - - - - -

INPUT =BMDP-Eingabedatei, die Rohdaten enthält oder mit SAVE(BMDP-File) erzeugt wurde.

*Voreinstellung: keine Eingabedatei

l-datei Datei mit der logischen Gerätenummer l,
mit $10 \leq l \leq 99$. l muss im INPUT-Paragraphen der BMDP-Steuerkarten angegeben werden:
UNIT IS l.

Dokumentation: BMDP77-Manual Seite 75ff

In Verbindung mit SUB=TRANSF und der CODE-Spezifikation können beliebig formatierte und binäre Daten eingelesen werden.

Dokumentation: BMDP77-Manual Seite 110

- - - - - -

SAVE =BMDP-SAVE-Datei(BMDP-File)

*Voreinstellung: es wird keine SAVE-Datei erzeugt.

l-datei Datei mit der logischen Gerätenummer l,
mit $10 \leq l \leq 99$.
l muss im SAVE-Paragraphen der BMDP-Steuerkarten angegeben werden:
UNIT IS l.

Kreation und Erweiterung einer SAVE-Datei:
siehe Beispiele 3 und 4.

Dokumentation: BMDP77-Manual Seite 123ff

- - - - - -

SUB =Name der zu ändernden FORTRAN-Subroutine

*Voreinstellung: keine Subroutineänderung

zulässige Angaben:

subroutine TRANSF: Subroutine TRANSF (nicht BMDP1S)
TRANSS: Subroutine TRANSF von BMDP1S
FUN : Subroutine FUN von BMDP3R
CON : Subroutine CON von BMDP3R

Dokumentation: BMDP77-Manual Seite 106ff

- - - - - -

CODE =Angabe des eigenen FORTRAN-Codes für SUB

*Voreinstellung: kein eigener Code

/f oder /f%/ Fremdstring
datei Texthaltungsdatei

Dokumentation: BMDP77-Manual Seite 106ff

- - - - - -

OUTPUT =Ausgabedatei für eigenes Ausgabeformat

*Voreinstellung: keine Ausgabedatei

l-datei Datei mit der logischen Gerätenummer l, mit $10 \leq l \leq 99$. In Verbindung mit SUB=TRANSF und der CODE-Spezifikation können Daten und Ergebnisse (z.B. ein SAVE-File) formatiert und binär ausgegeben werden.

Dokumentation: BMDP77-Manual Seite 135ff

2 Hinweise

2.1 Fehlerbehandlung in BIMED

Treten bei der Anwendung des Kommandos oder in der intern abzuarbeitenden Komanndofolge Fehler auf, so wird das Kommando abgebrochen. Wird bei PROGRAM ein falscher Programmname bzw. bei SUB eine falsche Subroutine angegeben, erfolgt ein Sprung an das Prozedurende von BIMED mit der Meldung:
+++++ OPERATOR NICHT VORHANDEN: "operatorname" bzw.
WARNG.: MARKE NICHT GEFUNDEN, SPRUNG ZUM PROZEDURENDE

2.2 Benutzereigener FORTRAN-Code

Die Kommandoprozedur BIMEDT, wie in der vorhergehenden Version und wie im BMDP77-Manual beschrieben, ist ersetzt durch die Spezifikationen SUB und CODE im BIMED-Kommando (siehe Beispiel 6).

2.3 Dynamische Arbeitsspeichererweiterung

Mit der Spezifikation WORKSPACE ist die Möglichkeit gegeben, den Arbeitsspeicher eines Programmes dynamisch zu erweitern, ohne das Programm neu montieren zu müssen. Eine Neumontage erfolgt nur dann, wenn benutzereigener FORTRAN-Code (Spezifikationen SUB und CODE) angegeben wird.

2.4 BMDP-Programmaufruf im Gespräch

Steuerkarten und Rohdaten(Spezifikation CONTROL) müssen von einer Datei (80 Zeichen/Satz) eingegeben werden. Der BMDP-Ausdruck kann mit
%*DNUMMER=6U9 auf das Terminal und mit
%*Dnummer=6U4 auf das Terminal mit gleichzeitiger Ausgabe ins Druckerprotokoll umgeschaltet werden.

2.5 Standardeingabe- und Standardausgabedatei

Die Nummern der Standardeingabedatei(5) und der Standardausgabedatei(6) können mit %*DNUMMER umbenannt werden.

3 Betriebsmittelbedarf

Der Betriebsmittelbedarf ist stark problemabhängig, im folgenden einige Richtwerte:

KSB: ca.(40+WORKSPACE)K
PSB: $\geq$100K, bei temporären Dateien zusätzlicher Bedarf
TSB: ca. 50-100K
RZS: Rein problemabhängig, bei der Angabe von SUB und CODE wird für die Montage zusätzlich eine Rechenzeit von 30 Sekunden benötigt.

Es wird empfohlen, zunächst einen grosszügigen Bedarf anzufordern, der dann in folgenden Aufträgen genau angepasst werden kann (Kosten!).

4 Literatur

BMDP-77
Biomedical Computer Programs
P-Series
W.J.Dixon
University of California Press
Berkeley - Los Angeles - London - 1977

Das BMDP77-Manual ist in den Benutzerräumen, der Beratung und an den RJE-Stationen vorhanden.
Beachten Sie die Fussnoten bezüglich der Unterschiede in den BMDP-Versionen!

5 Beispiele

5.1 Beispiel 1: BMDP-Standardauftrag

```
/2XBA,BEN=akz,KSB=50,PSB=200,TSB=100,RZS=1%.      Auftragskarte
paswor                                            Passwort
%ERZEUGE,SYSTEM.BMDP                              BIMED-Bereitstellung
%BIMED,PROGRAM=BMDP1D,WORKSPACE=5,CONTROL=/       BMDP-Aufruf
/PROBLEM     TITLE IS 'TEST BMDP1D'.              ..
/INPUT       VARIABLES ARE 12.                    :
             CASES ARE 110.                       :
             FORMAT IS '(12F2.0)'.                : BMDP-Steuerkarten
/VARIABLE    NAMES ARE ....                       :
             MINIMUM IS ...                       :
             MAXIMUM IS ...                       :
/END                                              ..BMDP-Ende
.... Rohdaten: 110 Karten mit je 12 Werten        ..
im Format F2.0 ...                                : Daten
/FINISH                                           ..Datenende
/2XEN%.                                           Auftragsende
```

5.2 Beispiel 2: Rohdateneingabe von einer Datei

```
/2XBA,BEN=akz,KSB=50,PSB=200,TSB=100,RZS=1%.
paswor
%ERZEUGE,SYSTEM.BMDP
%LFANMELDE,LESEN=ROHDAT
%BIMED,PROGRAM=BMDP1D,INPUT=10-ROHDAT,CONTROL=/
/PROBLEM     TITLE IS ...
/INPUT       UNIT IS 10.
             VARIABLES ARE ...
             FORMAT IS ...
/END
/FINISH
/2XEN%.
```

Die LFD-Datei ROHDAT enthält die von BMDP1D gelesenen Rohdaten. Die Nummern der UNIT- und INPUT-Angabe müssen übereinstimmen.
Liegen Rohdaten in uneinheitlichem Format oder binär vor, können sie mit eigenem Format eingelesen werden. In Verbindung mit SUB=TRANSF und der CODE-Spezifikation werden die nötigen FORTRAN-Anweisungen übergeben (siehe Beispiel 6 und BMDP77-Manual Seite 110).

5.3 Beispiel 3: Kreation eines BMDP-Files (SAVE)

```
%2XBA,BEN=akz,KSB=50,PSB=200,TSB=100,RZS=1%.
paswor
%ERZEUGE,SYSTEM.BMDP
%DATEI,NAME=SAVDAT,TYP=SEQ,SATZZAHL=U1,SATZBAU=U1W,TRAEGER=P
%BIMED,PROGRAM=BMDP1D,SAVE=20-SAVDAT,CONTROL=/
/PROBLEM    TITLE IS ...
/INPUT      VARIABLES ARE ...
            ......
/SAVE       CODE IS DAT1.
            UNIT IS 20.
            NEW.
/END
......Rohdaten...
/FINISH
%KOPIERE,DATEI=SAVDAT,QUELLTRAEGER=-STD-,ZIELTRAEGER=LFD,
MODUS=BLK´RES´AKT
%2XEN%.
```

Der BMDP-File wird auf der temporären Datei SAVDAT angelegt, die anschliessend in die LFD kopiert wird. Die Nummern der SAVE- und UNIT-Angabe müssen übereinstimmen, die Angabe NEW muss gesetzt sein.
Da die genaue Berechnung der Satzzahl und des Satzbaus eines BMDP-Files schwierig ist, wird folgende Arbeitsweise empfohlen:

1. TRAEGER=LFD oder Wechselplatte

Es wird eine temporäre Datei mit SATZZAHL=U1 und SATZBAU=U1W kreiert, die anschliessend in die LFD bzw. auf eine Wechselplatte kopiert wird. (MODUS=BLK´RES´AKT)

2. TRAEGER=Magnetband

Der BMDP-File wird mit SATZZAHL=U1 und SATZBAU=U1W direkt auf dem Magnetband kreiert; das Kopieren entfällt.

In beiden Fällen wird der BMDP-File auf die richtige Grösse expandiert.

5.4 Beispiel 4: Erweiterung eines BMDP-Files

```
§2XBA,BEN=akz,KSB=50,PSB=200,TSB=100,RZS=1§.
paswor
§ERZEUGE,SYSTEM.BMDP
§KOPIERE,DATEI=SAVDAT,QUELLTRAEGER=LFD,ZIELTRAEGER=-STD-,
MODUS=BLK´RES´AKT
§BIMED,PROGRAM=BMDP1D,SAVE=20-SAVDAT,CONTROL=/
/PROBLEM    TITLE IS ...
/INPUT      ....
/SAVE       CODE IS DAT2.
            UNIT IS 20.
/END
....Rohdaten...
/FINISH
§KOPIERE,DATEI=SAVDAT,QUELLTRAEGER=-STD-,ZIELTRAEGER=LFD,
MODUS=BLK´RES´AKT
§2XEN§.
```

Die Datei SAVDAT, die den BMDP-File mit CODE=DAT1 aus Beispiel 3 enthält, wird aus der LFD in die Standarddatenbasis(-STD-) kopiert, von BMDP1D um den BMDP-File mit CODE=DAT2 erweitert und dann in die LFD zurück kopiert. Im SAVE-Paragraphen darf die Angabe NEW nicht gesetzt sein.

5.5 Beispiel 5: Dateneingabe von einem BMDP-File

```
§2XBA,BEN=akz,KSB=50,PSB=200,TSB=100,RZS=1§.
paswor
§LFANMELDE,LESEN=SAVDAT
§ERZEUGE,SYSTEM.BMDP
§BIMED,PROGRAM=BMDP2D,INPUT=21-SAVDAT,CONTROL=/
/PROBLEM    TITLE IS ...
/INPUT      UNIT IS 21.
            CODE IS DAT1.
/VARIABLE   GROUPING IS ...
/END
/FINISH
§2XEN§.
```

Der BMDP-File mit CODE=DAT1 (Beispiel 3 und 4) auf der Datei SAVDAT wird von Programm BMDP2D als Eingabe verwendet. Im INPUT-Paragraphen darf keine FORMAT-Angabe, muss eine CODE-Angabe gemacht sein. Damit wird dem BMDP-Programm mitgeteilt, dass die Eingabedatei einen BMDP-File (Binärdaten) und nicht Rohdaten enthält.

5.6 Beispiel 6: Benutzereigene FORTRAN-Transformationen

```
§2XBA,BEN=akz,KSB=50,PSB=200,TSB=100,RZS=1§.
paswor
§ERZEUGE,SYSTEM.BMDP
§BIMED,PROGRAM=BMDP1D,SUB=TRANSF,CODE=/
       X(10)=X(4)/X(3)
§/,CONTROL=/
/PROBLEM     TITLE IS ...
/INPUT       VARIABLES ARE ....
             ......
/VARIABLE    NAMES ARE V1,V2,HEIGHT,WEIGHT,V5,
             V6,V7,V8,V9,RATIO.
/END
.....Rohdaten...
/FINISH
§2XEN§.
```

Mit SUB=TRANSF wird die für benutzereigene FORTRAN-Transformationen benötigte Subroutine TRANSF um die hinter CODE=/ folgenden FORTRAN-Statements erweitert. TRANSF wird übersetzt, BMDP1D neu montiert und ausgeführt. Der angegebene Code berechnet aus den Variablen 4 (WEIGHT) und 3 (HEIGHT) die neue Variable 10 (RATIO), die im VARIABLE-Paragraph aufgeführt sein muss.

5.7 Beispiel 7: Ausgabe eines BMDP-Files in eigenem Format

```
§2XBA,BEN=akz,KSB=50,PSB=200,TSB=100,RZS=1§.
paswor
§ERZEUGE,SYSTEM.BMDP
§LFANMELDE,LESEN=SAVDAT
§DATEI,NAME=BCDOUT,TYP=RAM,SATZZAHL=U1,SATZBAU=U80Ø,TRAEGER=P
§BIMED,PROGRAM=BMDP1D,SUB=TRANSF,CODE=/
       WRITE(20,1000) (X(I),I=1,5)
1000   FORMAT(5F6.2)
§/,INPUT=10-SAVDAT,OUTPUT=20-BCDOUT,CONTROL=/
/PROBLEM     TITLE IS ...
/INPUT       UNIT IS 10.
             CODE IS DAT2.
/END
/FINISH
§KOPIERE,DATEI=BCDOUT,QUELLTRAEGER=-STD-,ZIELTRAEGER=LFD,
MODUS=BLK´RES´AKT
§2XEN§.
```

Es wird die Texthaltungsdatei BCDOUT kreiert, BMDP1D liest von der Datei SAVDAT den in Beispiel 3 erzeugten BMDP-File mit CODE=DAT2 und schreibt die Werte der 5 Variablen nach BCDOUT. BCDOUT wird mit KOPIERE permanent angelegt und kann mit anderen Programmen (z.B. SPSS) weiterverarbeitet werden. Die Zahl der Sätze auf BCDOUT entspricht der Zahl der Fälle des BMDP-Files.

Regionales Rechenzentrum Erlangen	R R Z E Programmbibliothek TR440	S P S S N E W S Bearb.: H.Cramer Datum : Okt. 78

Informationen zu SPSSH Release 6.02/Operator 602.13

Noch nicht implementiert sind

- SAVE ARCHIVE
- OSIRIS-Anschluß (wird auch vorerst nicht implementiert werden)
- die in der deutschen SPSS-Beschreibung aufgeführten GRZ-Erweiterungen für interaktiven Betrieb (auch nicht das von Version 4 bekannte SWITCH)
- INPUT FORMAT BINARY (bis auf nV mit 1 < n < 32)
- Bearbeitung von Alleinzugriffsgeräten durch das Kommando §SPSS, d.h. insbesondere, daß Dateien auf Magnetbändern vorher eingeschleust oder kreiert werden müssen.

Noch nicht behobene Fehler,

die unbemerkt zu falschen Ergebnissen führen können, sind die Fehler:

28, 42, 43, 44, 47, 53, 63, 64, 65, 67, 69.

Die Fehlertexte erhalten Sie durch das Kommando

```
§ERZEUGE,DOKUMENT.SPSSFEHLER(geraet)
```

als Teilauftrag ausgedruckt.

Schwierigkeiten und Fehler

melden Sie uns bitte umgehend. Besonders möchten wir darauf hinweisen, daß die Routinen für formatierte Eingabe völlig neu in TAS geschrieben wurden.

Die Fehlerbehandlung

der Format-Konvertierungsroutine FCVT wurde verbessert.

Bei der Ausführung von ´READ INPUT DATA´ oder ´READ MATRIX´ können Fehler auftreten, von denen hier die Nummern 1776, 1779 und 1788 näher erläutert werden.

Darüber hinaus gibt FCVT (statt wie bisher Dumps) interne Fehlermeldungen aus in der Form:

```
+++++ INTERNER FEHLER : FCVT´2-stellige Zahl´
      KONTROLLPUNKT   : )
      CALLNUMBER      : ) ´6-stellige Zahl´
      RETURN          : )
```

Fehler 1776

Die Fehlernummer 1776 mit dem Text ´A DATA SET CANNOT BE OPENED´ bedeutet i.a., daß eine Datei
- nicht vorhanden (d.h. weder kreiert noch angemeldet oder eingeschleust) ist oder
- in einer anderen Datenbasis als angegeben liegt oder
- im SPSS-Kommando keiner bzw. einer falschen Spezifikation zugeordnet wurde.

Wenn keiner dieser Fehler vorliegt, kann beim Versuch, die Datei zu eröffnen, Kernspeichermangel aufgetreten sein. Beim ersten Zugriff auf eine Datei (also noch nicht bei der Kreation) wird im Kernspeicher ein Schreib/Lese-Puffer von 1K Länge eingerichtet. Da dann auch der Trommelspeicherbedarf um 1K steigt, kann Fehler 1776 auch auf Trommelspeichermangel hindeuten.

Fehler 1779

mit dem Text ´AN INPUT RECORD WAS TOO SHORT FOR THE INPUT FORMAT´ bedeutet i.a., daß ein Datensatz kürzer als erwartet ist. Insbesondere wenn die Daten in eine Datei eingetragen wurden, kann es notwendig sein, die Sätze auf die erforderliche Satzlänge mit Leerzeichen zu füllen.

Error 1779 wurde bisher auch immer dann ausgegeben, wenn bei der Konvertierung eines Eingabeelementes ein Fehler auftrat. Da es sich in diesem Fall um Fehler im SPSS-System handelt, wird jetzt eine interne Fehlermeldung ausgegeben.

Fehler 1788

mit dem Text ´CODE FOR FORMAT INTERPRETATION TOO LONG. CHOOSE A SHORTER INPUT FORMAT WITH APPROPIATE REPLICATIONS´ zeigt an, daß der Befehlsspeicher zur Ausführung des Formats nicht ausreicht. Abhilfemöglichkeiten:

- Nur einen Teil der Variablen einlesen und weitere später mit ADD VARIABLES oder ADD DATA LIST zufügen. oder

- Auf DATA LIST verzichten und das Eingabeformat durch geeignete Klammerung mit Wiederholungsfaktoren kompakter schreiben.

Interne FCVT-Fehler

sollten i.a. nicht auftreten, auf jeden Fall aber dem RRZE gemeldet werden. Erwähnt seien hier nur die Folgenden

FCVT07 bzw. FCVT19

werden ausgegeben, wenn die Verschachtelungstiefe von Klammern größer als 5 wird.

FCVT97

bedeutet, daß eine nicht implementierte Leistung (BINARY) verlangt wurde.

Verbesserungen von SPSS(602.13) gegenüber SPSS(6.8)

In dem Operator SPSS(602.13) für Release 6.02 sind die Fehler 6, 8, 10, 27, 32, 83, 84, 85, 93, 103, 111, 119, 128 und 128 behoben.

Insbesondere kann GET ARCHIVE mit mehr als 500 inaktiven Variablen verwendet werden. Es sollte jetzt auch möglich sein, durch Verwendung von GET ARCHIVE statt GET FILE PSB einzusparen, wenn nur die benötigten Variablen eingelesen werden.

Außerdem sind alle bisher bekannt gewordenen Schwierigkeiten im Zusammenhang mit READ MATRIX behoben.

Kompatibilität mit SPSS-4-Systemdateien

Mit SPSS Version 4 erzeugte Systemdateien können nur unter der Kanalnummer 30 eingelesen werden d.h., daß von den einzulesenden Archive-Dateien nur eine aus SPSS4 stammen darf.

Neuer Kommandoperator B&SPSS(2.3)

Das Kommando §SPSS hat als Leistungserweiterung eine neue Spezifikation LISTING erhalten. LISTING erlaubt die Angabe einer Datei, in die die Ergebnisausgabe geschrieben wird, die sonst ins Druckerprotokoll erfolgt.

Beispiel:

```
§SPSS,...,LISTING=AUS
```

Die Ergebnisse stehen in der Datei AUS. Sie können mit DRUCKE ausgegeben werden oder weiterverarbeitet werden.
Wenn AUS nicht existiert, wird die Datei mit Typ SEQ und Satzbau M133A kreiert. Soll die Ausgabe als Texthaltungsdatei verarbeitet werden, muß die Datei vorher kreiert werden z.B.

```
§DATEI,AUS,RAM,U1000,M1330
```

1000 Sätze reichen für etwa 20 Seiten SPSS-Ausgabe.

Die Kommandobeschreibung wird demnächst um die Spezifikation LISTING ergänzt.

Korrektur zur Spezifikation OUTPUT

In Version 6.08 trat ein Fehler auf, wenn unter OUTPUT im Kommando SPSS keine Datei angegeben und Rohdatenausgabe angefordert wurde. In dieser Version wird OUTPUT = BCDAUSGABE voreingestellt. Das SPSS-Kommando kreiert dann immer, falls keine andere Datei angegeben wird, eine WO-Datei BCDAUSGABE und loescht sie nach dem SPSS-Lauf, falls sie leer ist.