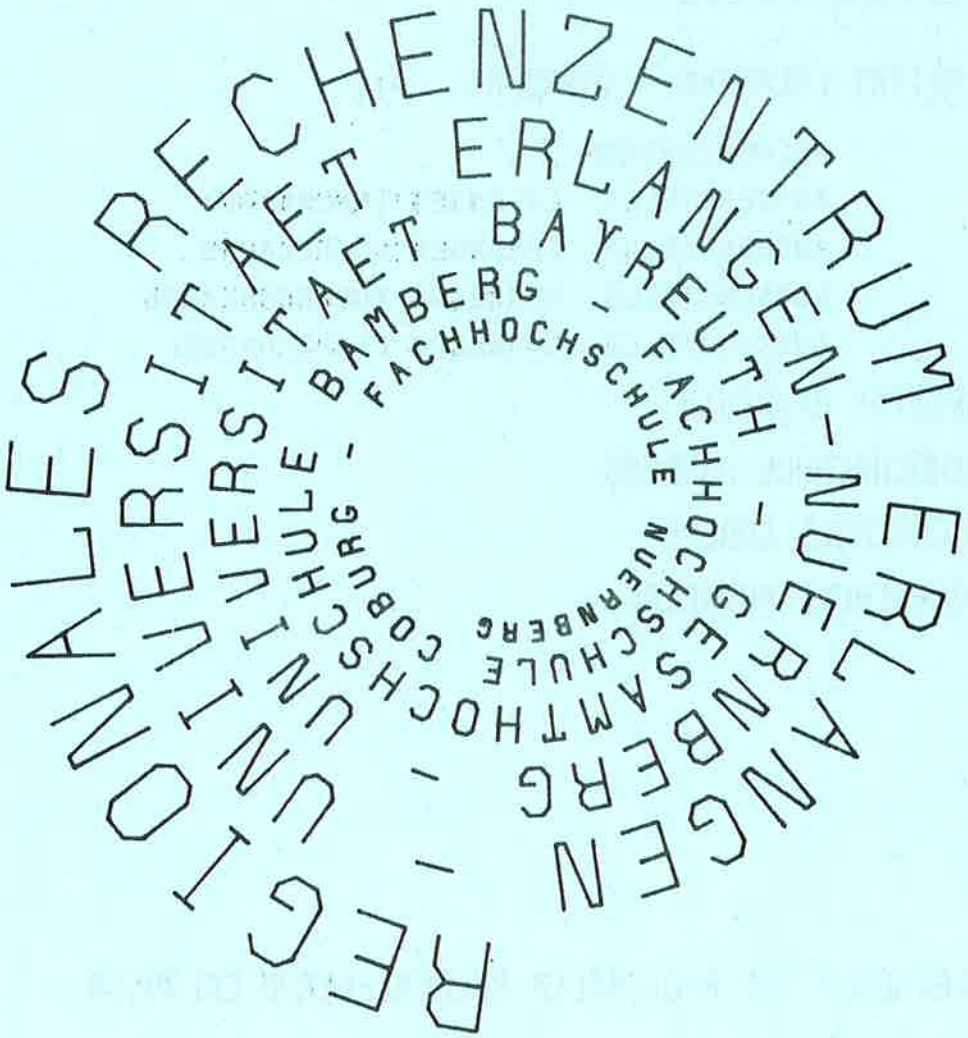
RRZE

BENUTZERINFORMATION



BI 10 -ERLANGEN- 1.AUG.1977

R R Z E
REGIONALES RECHENZENTRUM
MARTENSSTRASSE 1

852o ERLANGEN

TEL: o9131 / 85 7o 31 - 85 7o 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

INHALT:

Anhang

Auf den beiden letzten (gelben) Blättern bringen wir Ihnen ein Stichwortverzeichnis der Benutzer-Inform. Nr. 1 bis 9. Es soll Ihnen die Suche nach Information erleichtern, solange noch kein einheitliches Benutzerhandbuch existiert.

NACH REDAKTIONSSCHLUSS:

Unser Filebestand an der CYBER nähert sich der Kapazitätsgrenze.

Um zu vermeiden, daß zu viele tote Files auf den Platten residieren, werden künftig zu Monatsbeginn alle Files, deren letztes Zugriffsdatum mehr als 2 Monate zurückliegt, automatisch auf Magnetband ausgelagert.

Bitte wenden Sie sich ggf. an die Aufsicht bzw. Beratung.

1 Aktuelle Information

1.1 Feiertagsregelung

Am Montag, 15. 8. 1977 (Mariä Himmelfahrt, ges. gesch. Feiertag) gilt:
Betrieb der Anlagen TR44o und CYBER 172 und Hausöffnungszeiten 14 - 22 Uhr.

1.2 Magnetbandarchiv

Zur Zeit und auch künftig werden Magnetbänder am Regionalen Rechenzentrum je nach Lagerplatz unter verschiedenen Kennzeichen archiviert:
am TR44o beginnen die Kennzeichen mit Txxxx
an CYBER 172 beginnen die Kennzeichen mit Cxxxx
Alle Magnetbänder, die noch nicht in dieser Form gekennzeichnet sind, bitten wir, bis 3o. Sept 1977 über die Aufsicht des Rechenzentrums in das Archiv übernehmen zu lassen. Nach diesem Termin werden Leihbänder (CD 33oo - Kennzeichen RZxxx bzw. Zxxx) gelöscht, Benutzerbänder (CD 33oo - Kennzeichen: Bxxx) den Eigentümern zugestellt. Diese Regelung gilt auch für Leih- oder Benutzerbänder aus dem Archiv des ehemaligen INFRA-TR44o.

Termin!

Termin!

1.3 Expresstationen TR44o und CYBER

am TR44o wird seit Jahresbeginn,
an CYBER 172 ab August 1977
je eine Expresstation mit Kartenleser und Drucker betrieben.
Diese Station steht dem Benutzer zur "Selbstbedienung" zur Verfügung.
Dabei gelten folgende Spielregeln:

1. Die Station darf nur von denjenigen Benutzern bedient werden, die dafür ausgebildet sind. Zu diesem Zwecke finden regelmäßig Einweisungen für Interessenten statt. Die Berechtigung wird dann in einer Liste dokumentiert.

2. Um andere Benutzer nicht zu behindern, gelten folgende Maximalwerte:
Maximale Jobgröße: 2ooo Karten
Maximale Ausgabe: 3o Seiten
Keine Peripherieanforderungen

Zur Zeit können an den Stationen Jobs mit Rechenzeiten bis zu 1 Stunde eingelesen werden. Diese Schranke muß bei steigender Rechnernutzung evtl. herabgesetzt werden.

1.4 LK-Sortierer vom Typ MAUL

Den Benutzern vom Regionalen Rechenzentrum Erlangen steht ab sofort ein LK-Sortierer zur Verfügung. Der Zugang erfolgt über die Operateure am TR44o. Er ist z.B. einsetzbar für das Vorsortieren größerer LK-Datenbestände.

INHALT:

Anhang

Auf den beiden letzten (gelben) Blättern bringen wir Ihnen ein Stichwortverzeichnis der Benutzer-Inform. Nr. 1 bis 9. Es soll Ihnen die Suche nach Information erleichtern, solange noch kein einheitliches Benutzerhandbuch existiert.

NACH REDAKTIONSSCHLUSS:

Unser Filebestand an der CYBER nähert sich der Kapazitätsgrenze.

Um zu vermeiden, daß zu viele tote Files auf den Platten residieren, werden künftig zu Monatsbeginn alle Files, deren letztes Zugriffsdatum mehr als 2 Monate zurückliegt, automatisch auf Magnetband ausgelagert.

Bitte wenden Sie sich ggf. an die Aufsicht bzw. Beratung.

1 Aktuelle Information

1.1 Feiertagsregelung

Am Montag, 15. 8. 1977 (Mariä Himmelfahrt, ges. gesch. Feiertag) gilt:
Betrieb der Anlagen TR44o und CYBER 172 und Hausöffnungszeiten 14 - 22 Uhr.

1.2 Magnetbandarchiv

Zur Zeit und auch künftig werden Magnetbänder am Regionalen Rechenzentrum je nach Lagerplatz unter verschiedenen Kennzeichen archiviert:
am TR44o beginnen die Kennzeichen mit Txxxx
an CYBER 172 beginnen die Kennzeichen mit Cxxxx
Alle Magnetbänder, die noch nicht in dieser Form gekennzeichnet sind, bitten wir, bis 3o. Sept 1977 über die Aufsicht des Rechenzentrums in das Archiv übernehmen zu lassen. Nach diesem Termin werden Leihbänder (CD 33oo - Kennzeichen RZxxx bzw. Zxxx) gelöscht, Benutzerbänder (CD 33oo - Kennzeichen: Bxxx) den Eigentümern zugestellt. Diese Regelung gilt auch für Leih- oder Benutzerbänder aus dem Archiv des ehemaligen INFRA-TR44o.

Termin!

Termin!

1.3 Expresstationen TR44o und CYBER

am TR44o wird seit Jahresbeginn,
an CYBER 172 ab August 1977
je eine Expresstation mit Kartenleser und Drucker betrieben.
Diese Station steht dem Benutzer zur "Selbstbedienung" zur Verfügung.
Dabei gelten folgende Spielregeln:

1. Die Station darf nur von denjenigen Benutzern bedient werden, die dafür ausgebildet sind. Zu diesem Zwecke finden regelmäßig Einweisungen für Interessenten statt. Die Berechtigung wird dann in einer Liste dokumentiert.

2. Um andere Benutzer nicht zu behindern, gelten folgende Maximalwerte:
Maximale Jobgröße: 2ooo Karten
Maximale Ausgabe: 3o Seiten
Keine Peripherieanforderungen

Zur Zeit können an den Stationen Jobs mit Rechenzeiten bis zu 1 Stunde eingelesen werden. Diese Schranke muß bei steigender Rechnernutzung evtl. herabgesetzt werden.

1.4 LK-Sortierer vom Typ MAUL

Den Benutzern vom Regionalen Rechenzentrum Erlangen steht ab sofort ein LK-Sortierer zur Verfügung. Der Zugang erfolgt über die Operateure am TR44o. Er ist z.B. einsetzbar für das Vorsortieren größerer LK-Datenbestände.

1.5 Schriften, die in der Aufsicht zu erwerben sind

Schriften für die CYBER:

FORTRAN Extended	DM 7,--
NOS Ref. Man. Vol. 1	DM 5,--
NOS Ref. Man. Vol. 2	DM 7,--
Timesharing Users Ref. Man.	DM 5,--
ALGOL Vers. 4 Ref. Man.	DM 9,--
COMPASS Vers. 3 Ref. Man.	DM 9,--
Einführung in die Benutzung der CYBER (Kurzfassung von NOS Ref. Man. Vol. 1)	DM 2,--
Applications Programmers Instant	DM 2,--
FORTRAN Extended Instant	DM 1,--
Loader Instant	DM 1,5o

Schriften für den TR 44o:

FORTRAN-Sprachbeschreibung	DM 5,--
Kommandotaschenbuch	DM 2,--

Vorlesungsskript:

Programmierung in FORTRAN an den Rechnern TR 44o und CYBER 17o	DM 3,--

1.6 Maschinenzustand

	mittlerer Fehlerabstand in Std.		Ausfallzeit in Std.	
	TR 44o	CYBER	TR 44o	CYBER
Mai	25	41	17	16
Juni	22	38	39	11
Juli	8o	52	5	33

1.7 Neue Druckerkette an der CYBER 172

Seit dem 2. August 1977 sind an allen Druckern der CYBER neue Druckketten eingebaut, die mit dem standardisierten ASCII-Zeichensatz bestückt sind.
Die folgende Liste zeigt die noch bestehenden Unterschiede in einzelnen Sonderzeichen.

Lochertaste (KC2)	¢	!	|	082	¬	"	#	@	$	?	´	&	_
Drucker CYBER Alt	[	]	∨	≥	¬	≠	≡	≤	$	↓	↑	∧	→
Drucker CYBER Neu	[	]	!	\	^	"	#	@	$	?	´	&	_
Drucker TR440 DC1	]	!	|	¤	¬	[	Ä	Ö	Ü	?	´	&	_
Drucker TR440 DC2	!	!	|	¤	¬	"	#	@	$	?	´	&	_

2. Neues zum Stand der Software

2.1 Programmbibliothek TR 44o-&STARG

2.1.1 Neu implementierte Objekte in der Bibliothek

a) Unterprogramme ALSETZ und ALTEST
Mit den Unterprogrammen ALSETZ und ALTEST können in ALGOL- und FORTRAN-Programmen Alarme (z.B. arithmetischer Alarm bei Division durch Null, Speicherschutzalarm bei Überschreitung von Feldgrenzen) abgefangen und programmintern behandelt werden.

b) Programm MUCH
Aufgaben-(Item-)Analysen für Tests, deren Aufgaben in die Mehrfachwahlform ('Multipler Choice') gekleidet sind. Die genauen Programmbeschreibungen stehen im Anhang oder sind in der Aufsicht erhältlich.

2.1.2 9. Lawine des STARG-Programmaustausches

Die 9. Lawine des STARG-Programmaustausches ist gerade eingetroffen. Die zugehörigen Quellprogramme stehen auf der Wechselplatte W14(STARG9). Auf die Quellen kann mit dem Kommando EINSCHLEUSE zugegriffen werden. Die zugehörige Dokumentation hat den Stand 31.5.77. Sie ist in der Beratung verfügbar und kann über das im Anhang beschriebene Kommando STARG erstellt werden.

2.2 Programmbibliothek TR 44o

Neue Programme: Beschreibung im Anhang

INSPSS
Datenaufbereitungsprogramm für SPSS-Rohdaten.

GKWANDLE
Wandeln einer Datei vom großen in den kleinen Zeichensatz.

2.3 SAC-1

SAC-1 ist ein Programmsystem für symbolische algebraische Berechnungen (Symbolic Algebraic Calculations). Es besteht aus einer Menge von FORTRAN-Unterprogrammen, die z.Zt. in 13 Subsystemen zusammengefaßt sind.

Diese Unterprogramme befinden sich in einer Bibliothek unter dem Benutzerkennzeichen &&SAC1; sie stehen nach der Anmeldung dieser Bibliothek zur Verfügung. Neben der Bibliothek befinden sich auf diesem BKZ einige Dateien, die Testprogramme mit Daten für einige der Subsysteme enthalten. Sie können mit dem Kommando

¤TUE,<Testprogrammdatei>

bearbeitet werden.
Nachfolgend ist eine Kurzbeschreibung der Subsysteme wiedergegeben; die ausführliche Dokumentation kann in der Aufsicht eingesehen werden.

SAC-1 Subsystem Summary

SAC-1 consists of a series of subsystems, each subsystem providing the system with some additional or enhanced capability. At present, 13 such subsystems have been completed, documented and released. Following are brief paragraphs summarizing the facilities and capabilities of each of the currently available subsystems.

1. The SAC-1 List Processing System (LP).

A list processing system on which all subsequent subsystems are dependent, since most of the arithmetic and algebraic objects processed by them are represented internally as lists.

2. The SAC-1 Integer Arithmetic System - Version III (IA).

This subsystem performs operations on infinite-precision integers, i.e. arbitrarily large integers represented as lists. Operations provided include the arithmetic operations, greatest common divisor calculation, and input/output.

3. The SAC-1 Polynomial System (PO).

Provides operations on polynomials in any number of variables with infinite-precision integer coefficients. Operations provided include addition, subtraction, multiplication, division, substitution and evaluation, differentiation and input/output. Greatest common divisor calculations are provided by the SAC-1 Polynomial GCD and Resultant System (see (5) below).

4. The SAC-1 Modular Arithmetic System (MA).

Provides the arithmetic operations in the finite field GF (p), where p is any odd single-precision prime number, and numerous operations on univariate or multivariate polynomials with coefficients in GF(p). Also included are programs for the Chinese remainder theorem and interpolation, for generating a list of large single-precision prime numbers, and for factorization of univariate polynomials over GF(p).

5. The SAC-1 Polynomial GCD and Resultant System (GR).

Provides programs based on very fast modular algorithms for computing the greatest common divisor or resultant of multivariate polynomials with infinite-precision integer coefficients. Also provides fast modular-algorithm programs for polynomial multiplication and division, and some other miscellaneous operations on polynomials.

6. The SAC-1 Rational Function System (RF).

Provides operations on rational functions whose numerators and denominators are multivariate polynomials with infinite-precision integer coefficients. Operations provided include addition, subtraction, multiplication and division, differentiation, substitution and input/output. Infinite-precision rational number arithmetic is included as a special case of rational function arithmetic.

7. The SAC-1 Partial Fraction Decomposition and Rational Function Integration System (RI).

Provides the partial fraction decomposition, relative to a square-free factorization, of any univariate rational function. Also, provides the indefinite integral of any univariate rational function in the form of its rational part plus the integrand of its transcendental part.

8. The SAC-1 Polynomial Real Zero System (RZ).

Computes, with guaranteed accuracy, to any specified precision, all the real zeros of any univariate polynomial with infinite-precision integer coefficients, and provides some other related capabilities.

9. The SAC-1 Polynomial Linear Algebra System (LA).

Performs operations on matrices whose elements are multivariate polynomials with infinite-precision integer coefficients. Operations include addition and multiplication, determinant calculation, inversion, nullspace basis calculation, and solution of systems of linear equations. Fast modular algorithms are used.

1o. The SAC-1 Polynomial Factorization System (PF).

Computes the complete factorization into irreducible polynomials (over the ring of the integers) of any univariate polynomial with infinite-precision integer coefficients. On the UNIVAC 111o, this system can factor most polynomials with single-precision coefficients and degrees up to about 25 in less than a minute.

11. The SAC-1 Gaussian Integer and Gaussian Polynomial System (GP).

Provides operations on infinite-precision Gaussian integers (i.e. complex numbers with integer real and imaginary parts) and multivariate polynomials over the Gaussian integers. Operations provided include addition, subtraction, multiplication, division, substitution and evaluation, differentiation and input/output. This system also includes a modular algorithm for calculating greatest common divisors.

12. The SAC-1 Complex Zero System (CZ).

Computes, with guaranteed accuracy, to any specified precision, all the real and complex zeros of any univariate polynomial with infinite-precision Gaussian integer of Gaussian rational coefficients.

13. The SAC-1 Real Algebraic Number System (RA).

Provides routines to perform operations in $Q(a)$ and $Q(a)[x]$, where Q is the field of rational numbers, and a is any real algebraic number. Operations provided include addition, subtraction, multiplication, division, greatest common divisor calculation, and comparison with respect to the order relation of $Q(a)$.

Entnommen aus:
G.E. Collins, S.C. Schaller,
SAC-1 User's Guide
Univ. of Wisconsin-Madison, Computer Science Department
TR-269 (1976)

2.4 Programmbibliothek-CYBER

Neue Programme: Beschreibung im Anhang

PASCAL

PASCAL : PASCAL-Compiler der ETH-Zürich von N. Wirth
PASCALS : Compiler für einen Subset von Standard-PASCAL
XREF : Cross Reference Generator für PASCAL-Programme
DECODE : Umsetzen von relocatablen binary code in COMPASS-Form

GET44o

Übernahme von Daten vom TR 44o zur CYBER

LIBMOD
Prozedur zur Verwaltung von Benutzerbibliotheken

LIBCALL
Prozedur zum Aufruf von Programmen von Benutzerbibliotheken

Anhang:

TR 44o-Kommandos:

ALSETZ
ALTEST
INSPSS

CYBER:

PASCAL
GET44o
LIBMOD/LIBCALL

9

STAENDIGE ARBEITSGRUPPE DER TR440 - RECHENZENTREN

ANWENDERPROGRAMM-BIBLIOTHEK

Mehrsprachl.	ALSETZ MELDET IN EINEM ALGOLPROGRAMM EINE NEUE ALARMADRESSE AN, SO DASS DER ALGOLPROGRAMMIERER BEIM AUFTRETEN EINES ALARMS SELBST ENTSCHEIDEN KANN, WIE DARAUF REAGIERT WERDEN SOLL. BEISPIELSWEISE KANN DANN MIT DER PROZEDUR ALTEST DIE ALARMURSACHE ABGEFRAGT WERDEN.	Bestell-Nr. E2.80.01.01
TAS		Datum 07.74

Das Unterprogramm ALSETZ ist in der Quellensprache TAS geschrieben.

Programmstart:
DEKLARATION:
JE NACH GEWUENSCHTER LEISTUNG ALS:
'PROCEDURE' ALSETZ(X);'CODE';
'INTEGER''PROCEDURE' ALSETZ;'CODE'
'INTEGER''PROCEDURE' ALSETZ(X);'CODE'
(IN FORTRAN DEKLARATION UNNOETIG)

PROGRAMMBESCHREIBUNG
AUFRUF:
I)ALS FUNKTIONSPROZEDUR OHNE PARAMETER
I:=ALSETZ
DIESE AUFRUFART BEWIRKT, DASS KEIN EREIGNISALARM ZUGESTELLT WIRD, SONDERN DIESE GESAMMELT WERDEN, UND DASS BEI EINEM ANDEREN ALARM DER OPERATOR AN DER UNTERBRECHUNGSSTELLE FORTGESETZT WIRD (ACHTUNG: KANN U.U. ZU EINER UNENDLICHEN SCHLEIFE FUEHREN BEI EINEM ALARM);
II) ALS EIGENTLICHE PROZEDUR ODER FUNKTIONSPROZEDUR MIT EINEM PARAMETER, DER
1) EIN INTEGERLABEL SEIN KANN: DANN WIRD NACH EINEM ALARM DER OPERATOR BEI DIESEM INTEGERLABEL FORTGESEZT.
2) EINE VARIABLE MIT DEM WERT = 0 SEIN KANN:
DANN WIRD NACH EINEM ALARM DER OPERATORLAUF HINTER DEM AUFRUF VON ALSETZ FORTGESETZT. DIESE AUFRUFART IST NUR SINNVOLL ALS FUNKTIONSPROZEDUR, DA ALSETZ DANN GLEICHZEITIG ALS ALTEST FUNGIERT MIT ENTSPRECHENDER WERTUEBERGABE.
VGL. ALTEST (E2.80.02.01).
III) IN FTN: CALL ALSETZ (&100)
ARBEITSWEISE:

DIE ALARMADRESSE WIRD MIT DEM SSR 0 20 AUF EINE ADRESSE IM UNTERPROGRAMM ALSETZ GESETZT. DORT WIRD NACH EINEM ALARM DIE ALARMURSACHE ABGEFRAGT, DER ALARM GELOESCHT (SSR 4 8) UND DIE URSPRUENGLICHE ALARMADRESSE DES OPERATORS WIEDERUM ALS ALARM-ADRESSE ANGEMELDET. DIE ALARMURSACHE WIRD AUSSERDEM FUER ALTEST GESPEICHERT, DAS ABER NICHT UNBEDINGT AUFGERUFEN WERDEN MUSS. DANN WIRD DER OPERATOR BEI DEM LABEL FORTGESETZT, DESSEN ADRESSE UND HIERARCHIE VORHER GESPEICHERT WORDEN WAREN.

PROGRAMMBESCHREIBUNG VOM 01.01.75
ROSENDAHL RZ
Bochum

STAENDIGE ARBEITSGRUPPE DER TR440 - RECHENZENTREN

ANWENDERPROGRAMM-BIBLIOTHEK

ALGOL-Code-Mehrsprachl.	ALTEST KANN AUFGERUFEN WERDEN, WENN VORHER IRGENDWANN EIN ALARM MIT HILFE VON ALSETZ ABGEFANGEN WORDEN IST. ALTEST FRAGT DANN DIE GESPEICHERTE ALARMURSACHE AB.	Bestell-Nr. E2.BO.02.01
TAS		Datum 07.74

Die Prozedur ALTEST ist in der Quellensprache TAS geschrieben

Kopf bzw. Vereinbarung der Prozedur:
DEKLARATION: IN ALGOL:
'INTEGER''PROCEDURE' ALTEST(X);'CODE';
IN FORTRAN: ALGOL EXTERNAL ALTEST

INTEGER ALTEST

PROGRAMMBESCHREIBUNG
AUFRUF:
ALS FUNKTIONSPROZEDUR IN ARITHMETISCHEN AUSDRUECKEN, Z.B.:
I:=ALTEST(K); ODER ALS EIGENTLICHE PROZEDUR.
DER PARAMETER MUSS EINE VARIABLE SEIN, DA IN IHR DIE ALARMURSACHE ALS INTEGERZAHL ABGELEGT WIRD.
DABEI BEDEUTEN DIE MOEGLICHEN WERTE:

0 EREIGNISALARM
1 ARITHMETISCHER ALARM
2 TYPENKENNUNGSALARM
3 SPEICHERSCHUTZALARM
4 UEBERLAUF REGISTER U
5 BEFEHLSALARM
6 DREIERPROBENALARM

ALS FUNKTIONSWERT WIRD UEBERGEBEN:

-1 WENN NOCH KEIN ALARM AUFGETRETEN IST (DIE VARIABLE WIRD DANN NICHT VERAENDERT.).
0 WENN KEIN EREIGNISALARM AUFGETRETEN IST (D.H. DIE VARIABLE AUF PARAMETERPOSITION UNGLEICH NULL).

WENN DIE ALARMURSACHE EIN EREIGNISALARM IST, DIE VARIABLE ALSO DEN WERT NULL ERHIELT, SO SPEZIFIZIERT DER FUNKTIONSWERT DIE ART DES EREIGNISALARMS NAEHER. DABEI BEDEUTEN:

FUNKTIONSWERT	ALARMART
0	NICHT SPEZIFIZIERT, Z.B. BEI EINTEFFENDEM XAN. BEI GESTZTEM WAHLSCHALTER 4
1	NETTOZEITUEBERSCHREITUNG DES ABSCHNITTS

2	UEBERSCHREITUNG DER DRUCKSEITENSCHRANKE DES ABLAUFPROTOKOLLS
3	OPERATEURALARM
4	MEHR ALS 1024 MAL SSR-FEHLERAUSGAENGE ANGESPRUNGEN
5	HALTBEFEHL VON DER KONSOLE EIGETROFFEN
6	ENTSCHLUESSLER SOLL GESPRAECH IN GRUND-ZUSTAND UEBERFUEHREN
7	SPERRE FUER GEMEINSCHAFTSGEBIET ZU LANGE GESETZT
8	NETTOZEITUEBERSCHREITUNG DES OPERATOR-LAUFS

DIESELBEN WERTE FUER DIE VARIABLE AUF PARAMETERPOSITION UND DEN FUNKTIONSWERT LIEFERT AUCH ALSETZ, WENN ES MIT EINER VARIABLEN MIT DEM WERT 0 ALS PARAMETER AUFGERUFEN WIRD.
DIE EREIGNISALARME 1 UND 2 DUERFEN NUR EINMAL IN EINEM ABSCHNITT VORKOMMEN. DANN WERDEN NOCH RESERVEZEIT UND RESERVESEITEN FUER DUMPS ETC. BEWILLIGT, BEIM ZWEITEN AUFTRETEN WIRD DANN DER ABSCHNITT SOFORT ABGEBROCHEN.
ARBEITSWEISE:
ES WERDEN NUR BEI BESTIMMTEN VARIABLEN VON ALSETZ DIE ALARMURSACHEN ENTNOMMEN, UND DIESE VARIABLE WIEDER AUF 'NOCH KEIN ALARM' GESETZT.
ACHTUNG: EINGANG IM MONTAGEOBJEKT ALSETZ (SIEHE DORT).

PROGRAMMBESCHREIBUNG VOM 01.01.75
ROSENDAHL RZ
Bochum

REGIONALES RECHENZENTRUM ERLANGEN

RRZE
PROGRAMMBIBLIOTHEK

TR 440 - KOMMANDO

I N S P S S

INSPSS	Start des Prüf- und Sortierprogramms INSPSS

Spezifikation :

STKART	= Steuerkarten für INSPSS-Lauf, Fremdstring oder Dateiname möglich
DATKART	= Datenkarten für INPSS-Lauf, Fremdstring oder Dateiname möglich
PRUEFAUS	= Ausgabe der geprüften oder sortierten Daten in Datei (TYP = SEQ oder RAM, Satzbau = U 80 Ø)
PRUEFEIN	= Dateneingabe aus Datei (TYP = SEQ oder RAM, Satzbau = U 80 Ø)
KARTZAHL	= Anzahl der zu verarbeitenden Datenkarten; siehe Spezifikation SATZZAHL im DATEI-Kommando

Kommando der RRZE - Programmbibliothek	anlagenspezifische Voreinstellung : KARTZAHL=U400 sonst = -

Einschränkung :

Nur wirksam, falls initialisiert durch Kommando SPSSINIT

Wirkung :

Das Prüf- und Sortierprogramm INSPSS wird gestartet. Es prüft oder sortiert eine Datenmenge und legt das Ergebnis in der Datei der Spezifikation PRUEFAUS ab. Ein Prüfprotokoll wird erstellt und als separater Teilauftrag abgesetzt. Da jede Art der Eingabe gelöst werden mußte (Karte, Datei) und ein Sortierbereich zur Vereinfachung gleich miteingerichtet wird, ist beim Plattenbedarf mit 6 * KARTZAHL zu rechnen: PSB = (1oo+15*6*KARTZAHL/1ooo).
Sind die Spezifikationen PRUEFEIN bzw. PRUEFAUS unbesetzt, erscheint die Fehlermeldung: +++++ OBLIGATE SPEZIFIKATIONSPOSITION UNBESETZT, die in diesem Fall ohne Bedeutung ist.
Die Ausgabedatei kann gleich der Eingabedatei sein.
Als Dateinamen sind nicht erlaubt: DATKART, STKART, DATEIN, DATAUS, SORTIER und AUSGABE, da diese intern verwendet werden!

Programm-Beschreibung "INSPSS"

INSPSS ist ein flexibles Karteneinlese-, Sortier- und Prüfprogramm. Es erstellt Plattendateien, die allen formalen Anforderungen des Programmpakets SPSS an die Rohdaten genügen.

INSPSS liest Lochkarten oder Dateien,
ergänzt eine bestehende Datei durch zusätzliche Daten von Lochkarten,
prüft auf numerischen Inhalt,
sortiert sie in aufsteigender Reihenfolge anhand eines Merkmals von maximal 18 Ziffern, deren Stellung auf der Lochkarte (bzw. der Datei) beliebig ist,
prüft die sortierten Sätze - soweit mehrere Karten pro Objekt existieren - auf Vollständigkeit,
entfernt vorkommende Doppelkarten bzw.
fügt fehlende Karten ein und
schreibt die nun formal vollständigen Datensätze auf eine Datei, die als SPSS-Eingabedatei verwendet werden kann.

Die Steuerung von INSPSS erfolgt über Parameterkarten, die ähnlich aufgebaut sind wie SPSS-Parameterkarten. Die Spalten 1 - 15 enthalten jeweils ein Schlüsselwort (Kontrollfeld), das die zu erfüllende Funktion beschreibt. Die Spalten 16 - 80 (Spezifikationsfeld) geben dem Programm zusätzliche Informationen über die Art der zu erfüllenden Aufgabe.

Die folgende Aufstellung faßt alle zulässigen Parameterkarten für INSPSS zusammen. Zur Erfüllung einer konkreten Aufgabe sind aber nicht immer sämtliche dieser Karten erforderlich. Welche dieser Karten benötigt werden, hängt vom jeweils zu lösenden Problem ab. Unterstrichene Worte weisen auf Standardfunktionen hin. Werden solche Standardfunktionen verlangt, so kann das entsprechende (Spezifikations-) Feld entweder leer bleiben oder mit dem (unterstrichenen) Wort gefüllt werden.

Kontrollfeld	Spezifikationsfeld
SPALTE 1	SPALTE 16
1. EINLESEN	KARTE (als Fremdstring von INSPSS) BAND (Band bedeutet Plattendatei)
2. NULLEN	NUMERISCH
3. SORTIEREN	AUFSTEIGEND
4. ID-NR-SPALTEN	XXXX.. 2-stellige Spaltennummern, max. 16 ID und 2 KA, letztes Zeichen: "*"
5. KA-NR-SPALTEN	XX..
6. PRUEFEN	XX Zahl der Kartennummern, zwei-stellig
7. KARTENFOLGE	XXXXXX.. zwei-stellige Kartennummern (max. 99) in gewünschter Reihenfolge
8. AUFFUELLEN	NEUNEN NULLEN LEERSTELLEN
9. AUFFUELLKARTE	XX Kartennummer der Auffüllkarte, zwei-stellig
10. PLUSKARTEN	
11. VIELFUELL	

Beschreibung der Funktionen einzelner Parameterkarten

1. EINLESEN

Soll die Eingabe über Lochkarten erfolgen, so kann das Spezifikationsfeld entweder leer bleiben oder das Kennwort "KARTE" tragen. Soll die Eingabe über Plattendatei erfolgen, so muß das Kennwort "BAND" im Spezifikationsfeld ab Spalte 16 erscheinen.

"KARTE": die Daten werden im INSPSS-Kommando nach der Spezifikation DATKART als Fremdstring übergeben.

"BAND": die Daten sind in einer Plattendatei (TYP = SEQ oder RAM, SATZBAU = U 80 Ø) enthalten.

Magnetbanddateien müssen vor dem INSPSS-Aufruf mit dem VERLAGERE-Kommando auf Platte gebracht werden. Im INSPSS-Kommando wird PRUEFEIN =‹dateiname›gesetzt.

2. NULLEN

Diese Parameterkarte wird erforderlich, wenn die Eingabesätze (Lochkarte oder Datei) Leerstellen enthalten, die während des Einleseprozesses durch Nullen ersetzt werden sollen. Wird ab Spalte 16 das Kennwort "NUMERISCH" geschrieben, so erfolgt eine Prüfung der Eingabesätze auf die Ziffern 0 - 9 und die Vorzeichen - und +. Kommen irgendwelche anderen Zeichen vor ("Doppellochung"), so wird der entsprechende Satz mit Fehlermeldung ausgedruckt und unverändert weiterverarbeitet.

3. SORTIEREN

Die Parameterkarte bewirkt, daß die eingelesenen Datensätze in aufsteigender Reihenfolge sortiert werden. In diesem Fall muß dem Programm durch zwei weitere Steuerkarten mitgeteilt werden, auf welchen Spalten der Lochkarte bzw. an welcher entsprechenden Stelle der Datei das Sortiermerkmal steht. Diese Mitteilung geschieht durch die Karten "ID-NR-SPALTEN", bzw. "KA-NR-SPALTEN". Kommen innerhalb des Sortiermerkmals nicht-numerische Zeichen vor, so wird der Satz unter Fehlermeldung ausgeschieden.

4. ID-NR-SPALTEN

Im Spezifikationsfeld dieser Karte müssen die Nummern aller Spalten aufgeführt werden, in denen die Identifikationsnummer des zu sortierenden Objektes steht. Es sind maximal 16 Spalten zulässig. Die Angabe der Spaltennummern muß 2-stellig ohne Zwischenraum erfolgen und durch einen Stern (*) abgeschlossen werden.

5. KA-NR-SPALTEN

Auf dieser Karte werden die Nummern jener Spalten aufgeführt, in denen die Kartennummer vermerkt ist. Entfällt die Kartennummer, da nur eine Karte pro Objekt existiert, so muß der Stern bereits auf Spalte 16 stehen. Die Kartennummern-Spalten werden wieder 2-stellig ohne Zwischenräume im Spezifikationsfeld vermerkt und durch einen Stern abgeschlossen.

6. PRUEFEN

Gehören mehrere Karten zu einem Objekt, so kann die Vollständigkeit der Karten geprüft werden. Im Spezifikationsfeld dieser Karte muß die Zahl der Karten, die zu einem vollständigen Objektsatz gehören, vermerkt werden (zweistellig!). Wird diese Parameterkarte benutzt, so müssen gleichzeitig die Parameterkarten "ID-NR-SPALTEN" und "KA-NR-SPALTEN" vorhanden sein. Außerdem muß unmittelbar nach der Parameterkarte "PRUEFEN" die weitere Parameterkarte "KARTENFOLGE" liegen. INSPSS liefert ein Protokoll über alle Unregelmäßigkeiten in der Kartenfolge. Kommen mehrere Karten mit gleichem Sortierbegriff (ID-NR, KA-NR) vor, so werden sie von der zweiten Karte an mit einem Vermerk "Doppelkarte" versehen ausgedruckt und gegebenenfalls (s. u. Nr. 8) ausgeschieden. Der erste Satz bleibt in jedem Fall erhalten und wird ausgedruckt, wenn er sich in mindestens einer der 80 Spalten von der folgenden "Doppelkarte" unterscheidet. In diesem Fall wird er durch "***" am Rande des Ausdrucks gekennzeichnet.
Bei fehlenden Karten wird der fehlende Satz mit dem Zusatz "Füllkarte" im Protokoll vermerkt.

7. KARTENFOLGE

Im Spezifikationsfeld dieser Parameterkarte werden alle Kartennummern, die zu einem vollständigen Kartensatz gehören, 2-stellig ohne Zwischenraum aufgeführt. Maximal sind 99 Kartennummern zulässig. In diesem Falle reicht natürlich eine einzige Parameterkarte nicht aus. Die Kartenfolgenummern werden dann im Spezifikationsfeld einer oder mehrerer Folgekarten (d.h. jeweils ab Spalte 16) fortgesetzt. Die Folgekarten dürfen im Kontrollfeld keine Lochung erhalten.

8. AUFFUELLEN

Die unter Punkt 6 und 7 aufgeführten Prüfroutinen bewirken zunächst nur, daß auf dem Drucker ein Protokoll über die doppelt vorhandenen oder fehlenden Karten eines Objektsatzes ausgegeben wird. Die geprüften Kartensätze werden jedoch un-

beachtet ihrer Unvollständigkeit auf die Ausgabedatei geschrieben. Soll eine automatische Auffüllung fehlender Karten bzw. ein Ausschluß von Doppelkarten erfolgen, so wird die Parameterkarte "AUFUELLEN" erforderlich. Im Spezifikationsfeld kann gewählt werden, ob die aufzufüllenden Karten abgesehen von der Identifikationsnummer und der Kartennummer "NEUNEN", "NULLEN" oder "LEERSTELLEN" enthalten soll. Werden "NEUNEN" (Standard) erwartet, so kann das Spezifikationsfeld frei bleiben.

9. AUFFUELLKARTE

In manchen Fällen wäre die Ergänzung fehlender Karten durch Sätze mit einem einheitlichen Füllzeichen (Neunen, Nullen, Leerstellen) unbefriedigend. Der Benutzer kann deshalb für einige oder alle der zu ergänzenden Kartenarten spezielle Kartenbilder vorgeben. Die "AUFFUELLKARTE" enthält in den Spalten 16 und 17 die Nummer der zu ergänzenden Kartenart. Ihr folgt unmittelbar jene Datenkarte, welche beim Fehlen der entsprechenden Kartenart eingeschoben werden soll. Die jeweilige Identifikationsnummer des zu ergänzenden Objektsatzes fügt INSPSS automatisch hinzu.

10. PLUSKARTEN

Unter besonderen Bedingungen ist es möglich, mit INSPSS zwei Dateien zu mischen bzw. eine bestehende Datei zu erweitern. Die Eingabe der Hauptdatei muß dazu über Plattendatei erfolgen. (EINLESEN_____BAND). Mit der Steuerkarte PLUSKARTE wird in diesem Fall bewirkt, daß im Anschluß an die Hauptdatei die mit DATKART = / eingelesenen Lochkarten eingefügt werden. Die Daten werden anschließend wie eine Datei behandelt (z.B. SORTIEREN, KARTENFOLGE usw.). Es versteht sich von selbst, daß sich die Ordnungsbegriffe (ID-Nr., KA-Nr.) beider Datensätze auf den gleichen Stellen befinden müssen.

11. VIELFUELL

INSPSS überprüft intern, wieviele Korrekturschritte beim Aufbau der Datei (Doppelkarten, Füllkarten, falsche Kartenarten-Nummern) erforderlich werden. Überwiegen diese Korrekturen, so bricht INSPSS den Lauf automatisch mit Fehler-

meldung ab.

Da es jedoch gelegentlich vorkommt, daß mit Absicht Dateien erzeugt werden, bei denen die Füllkarten überwiegen, muß der obengenannte automatische Programmabbruch unterdrückt werden. Dies geschieht durch die Parameterkarte VIELFUELL. Die Protokollierung der Füllkarten entfällt.

Allgemeine Beschränkungen:

1. Die Sortiermerkmale müssen numerisch sein, d.h. sie dürfen ausschließlich Ziffern 9 - o enthalten.

2. Das Sortiermerkmal darf maximal 18 Ziffern umfassen. Davon sind maximal 16 Ziffern für die Identifikations-Nummer und maximal 2 Ziffern für die Kartenarten-Nummer vorgesehen.

3. Bei Objekten mit nur einer Kartenart dürfen die Funktionen PRUEFEIN und AUFFUELLEN nicht aufgerufen werden (trivial!).

4. SORTIEREN von mehr als 4o ooo Datenkarten ist mit INSPSS unzulässig.

Beispiel:

```
    /2XBA,BEN=...,KSB=3o,PSB=1oo .
    .LFANMELDE,UNRZPB.GED
1.  .GEDAECHTNIS,GED,EIN
2.  .TEINTRAGE,DATIN,PROT.=-STD-,INF.=/
        ....Daten...
3.  .TDEKLARIERE,DATOUT,U2o
4.  .SPSSINIT
5.  .INSPSS,PRUEFEIN=DATIN,PRUEFAUS=DATOUT,KARTZAHL=U2o,STKART=/
6.  EINLESEN          BAND
7.  NULLEN            NUMERISCH
8.  SORTIEREN
    ID-NR-SPALTEN     o3o4o5o6o7o8
    KA-NR-SPALTEN     o1o2
9.  PRUEFEIN          11
    KARTENFOLGE       o1o2o3o4o5o6o7o8o91o11
1o. AUFFUELLEN
    AUFFUELLKARTE     o3
    o3      88888888888888......888888....
11. .TKOPIERE,DATOUT,PROT=-STD-
    /2XEN .
```

Nach Einlesen des Gedächtnisses der RRZE-Programmbibliothek (1.) werden die zu prüfenden Daten in die Datei DATIN eingetragen und aufgelistet (2.), die Ausgabedatei DATOUT wird angelegt(3.) und INSPSS mit SPSSINIT initialisiert (4.).

5. INSPSS werden die Ein- und Ausgabedatei, die Kartenzahl und die Steuerkarten angegeben.

6. Die Eingabedaten werden von der Datei DATIN gelesen. (Mit DATKART=/ und EINLESEN_KARTEN könnten sie direkt an INSPSS übergeben werden, jedoch ohne Auflistung)

7. Leerstellen sollen durch Nullen ersetzt werden, die so veränderten Eingabesätze sind auf numerischen Inhalt zu überprüfen.

8. Es soll sortiert werden, wobei die Identifikationsnummer in den Spalten o1 und o2 steht. (Nach der Sortierung

sind die Sätze mithin nach Identifikations-Nummern und innerhalb jeder Identifikationsnummer nach Karten-Nummern in aufsteigender Reihenfolge geordnet.)

9. Die Vollständigkeit der Objektsätze soll geprüft werden. Zu jedem Objektsatz gehören 11 verschiedene Kartenarten mit den Nummern o1 bis 11.

1o. Doppelkarten sollen ausgeschieden bzw. fehlende Karten durch Auffüllkarten ersetzt werden. Der Inhalt der Ergänzungskarten besteht - abgesehen von den jeweiligen Identifikations- und Karten-Nummern, nur aus "NEUNEN" (Standard, da die Karte AUFFUELLEN keines der Worte NULLEN oder LEERSTELLEN enthält.)
Eine Ausnahme soll hierbei nur die Kartenart o3 bilden. Hier ist abweichend von der generellen Regelung (Ergänzungskarte mit NEUNEN) ein spezielles Kartenbild für die Ergänzungskarte vorgegeben.

11. DATOUT enthält die für SPSS aufbereiteten Daten, welche hier zur Kontrolle aufgelistet werden.

INSPSS schreibt ein Eingabeprotokoll über die fehlerhaften Karten. In diesem Beispiel enthalten vier der eingelesenen Karten nicht-numerischen Inhalt. Drei dieser Karten werden trotzdem weiter verarbeitet. Nur die vierte Karte, welche ein nicht-numerisches Zeichen im Sortierbegriff ("F" auf Spalte 1) enthält und zweifach abgedruckt wurde, wird ausgeschieden.

Die eingelesenen Sätze werden sortiert und vor der Ausgabe auf DATOUT auf Vollständigkeit geprüft und korrigiert. Auskunft über die nötigen Reparaturen gibt das Ausgabeprotokoll. Im vorliegenden Beispiel wurde eine der beiden Karten mit der Identifikations-Nummer ooooo1 und der Kartennummer o1 als

DOPPELKARTE ausgeschieden; unmittelbar darüber steht die Karte mit gleichem Sortierbegriff, welche nicht ausgeschieden wurden. Gleichfalls ausgeschieden wurde die Karte mit der Kartennummer 12, da nur Kartenarten o1 - 11 zulässig waren. Andererseits mußten 5 Karten hinzugefügt werden ("FUELLKARTEN"), um vollständige Objektsätze zu jeweils 11 Karten zu erhalten.

Die abschließende Input/Output-Statistik faßt die Vorgänge noch einmal zahlenmäßig zusammen: von 2o eingelesenen Karten schieden 3 aus (1 Doppelkarte, 1 Karte mit falscher KA-Nummer, 1 Karte mit Codefehler im Sortierbegriff). Es blieben 17 Karten übrig. 5 Karten wurden zur Vervollständigung der Objektsätze (2 x 11 = 22 Karten) hinzugefügt.

Zu diesem Beispiel folgt ein INSPSS - Protokoll, die Auflistung der Eingabedaten (DATIN) und der Ausgabedaten (DATOUT).

INSPSS - Protokoll

```
EINLESEN        BAND
NULLEN          NUMERISCH
SORTIEREN
ID-NR-SPALTEN   030405060708*
KA-NR-SPALTEN   0102*
PRUEFEN         11
KARTENFOLGE     01020304050607080910 11
AUFFUELLEN
AUFFUELLKARTE   03
03      8888888888888888888888888888888888888888888888888888888888888888888888888888
*****************************************************************************************

 EINGABE-PROTOKOLL:
 040000001ABCDEFGHIJKLMNNNNNNOOOPPPPPP12345600007560000BBBBBB000008888888888000000000 CODE-FEHLER
 050000001.,+_)>:;¬'000000000000?"=!(789654'¬;:>000000000000000000000000000000000000 CODE-FEHLER
 010000020000TESTKARTEN000000 000000000000000000000000000000000000000000000000000000 CODE-FEHLER
 F1000002000222222222233333333707080906660000002220203050860880666542300875040 77 CODE-FEHLER
 F1000002000222222222233333333707080906660000002220203050860880666542300875040 77 GELOESCHT !

*****************************************************************************************
 AUSGABE-PROTOKOLL:
 010000010000012345678901234567800000000000300000000000000000000000000000000000000000                ***
 010000010000012345678901234567891111111111111111111000000000000000000000000000000000                DOPPELKARTE
 060000019999999999999999999999999999999999999999999999999999999999999999999999999999  FUELLKARTE
 020000029999999999999999999999999999999999999999999999999999999999999999999999999999  FUELLKARTE
 030000028888888888888888888888888888888888888888888888888888888888888888888888888888  FUELLKARTE
 040000029999999999999999999999999999999999999999999999999999999999999999999999999999  FUELLKARTE
 100000029999999999999999999999999999999999999999999999999999999999999999999999999999  FUELLKARTE
 120000020000010101010101010100000000000000300000000000000000000000000000000000000000                                  FALSCHE KA-NR.


 INPUT-OUTPUT-STATISTIK:

        20  RECORDS GELESEN; DAVON WURDEN
        17  RECORDS GESCHRIEBEN,
         1  RECORDS WURDEN ALS DOPPELKARTEN
         1  WEGEN FALSCHER KA-NR UND
         1  WEGEN CODE-FEHLER GELOESCHT.
         5  RECORDS WURDEN HINZUGEFUEGT.
```

Eingabedatei:
DATIN

```
DIE ZIELDATEI DATIN(0001,00) WURDE ERSTELLT
000010 0100000100000123456789 0123456789
000020 0100000100000123456789 0123456789 1111111111111111111
000030 0200000100000123456789 0123456789                    7777777777777
000040 0400000 1ABCDEFGHIJKLM NNNNOOOPPPPPP123456   756   BBBBBBB     8888888888
000050 0300000100000123456789 0123456789                    7777777777777
000060 0500000 1.,+-)>:;¬'              ?"=!(789654'¬;:>
000070 0700000100000123456789 0123456789                  81 2 3 4 5 6 78 9 93
000080 0800000100000123456789 0123456789                  81 2 3 4 5 6 78 9 93
000090 0900000100000123456789 012345611111111111 1 1 1 1 1 1 1 1 1 1 1 11111        111
000100 1000000100000123456789 012345611111111111 1 1 1 1 1 1 1 1 1 1 1 11111        111
000110 1100000100000123456789 012345611111111111 1 1 1 1 1 1 1 1 1 1 1 11111        11
000120 0100001020000TESTKARTEN
000130 F100000200002222222223333333 7 7 8 9 6 6      222 2 3 5 86 88 6665423  875 4 77
000140 0500000 2     2 2 2 2 2      2222222211 1 1 1 1 1 1 1 1 1 1 1 11111        11
000150 0600000 2     2 2 2 2 2      2222222211 1 1 1 1 1 1 1 1 1 1 1 11111        11
000160 0700000 2     2 2 2 2 2      2222222211 1 1 1 1 1 1 1 1 1 1 1 11111        11
000170 0800000 2     2 2 2 2 2      2222222211 1 1 1 1 1 1 1 1 1 1 1 11111        11
000180 0900000 2     2 2 2 2 2      2222222211 1 1 1 1 1 1 1 1 1 1 1 11111        11
000190 11    2        7 7 8 9 9 9 9      66666
000200 1200000 2     1 1 1 1 1 1 1 1 1
```

Ausgabedatei:
DATOUT

```
000001 0100000100000123456789 0123456789 0000000000000000000000000000000000000000000000000000000000000000
000002 0200000100000123456789 0123456789 0000000000000000000007777777777777000000000000000
000003 0300000100000123456789 0123456789 0000000000000000000007777777777777000000000000000
000004 0400000 1ABCDEFGHIJKLM NNNNOOOPPPPPP123456000756000BBBBBBB0000088888888880000000
000005 0500000 1.,+-)>:;¬'000000000000?"=!(789654'¬;:>0000000000000000000000000000000000
000006 0600000 1999999999999999999999999999999999999999999999999999999999999999999999999
000007 0700000100000123456789 0123456789 0000000000000081020304050607080909300000000000000
000008 0800000100000123456789 0123456789 0000000000000081020304050607080909300000000000000
000009 0900000100000123456789 012345611111111111010101010101010101010101011111000000000111
000010 1000000100000123456789 012345611111111111010101010101010101010101011111000000000111
000011 1100000100000123456789 012345611111111111010101010101010101010101011111000000000110
000012 0100000200000TESTKARTEN0000000000000000000000000000000000000000000000000000000000000
000013 0200000299999999999999999999999999999999999999999999999999999999999999999999999999999
000014 0300000288888888888888888888888888888888888888888888888888888888888888888888888888888
000015 0400000299999999999999999999999999999999999999999999999999999999999999999999999999999
000016 0500000200000202020202000000222222222110101010101010101010101010111110000000110
000017 0600000200000202020202000000222222222110101010101010101010101010111110000000110
000018 0700000200000202020202000000222222222110101010101010101010101010111110000000110
000019 0800000200000202020202000000222222222110101010101010101010101010111110000000110
000020 0900000200000202020202000000222222222110101010101010101010101010111110000000110
000021 1000000299999999999999999999999 999999999999999999999999999999 999999999999999
000022 1100000200000000770708090909090000066666000000000000000000000000000000000000000
```

Zum INSPSS-Beisp

GKWANDLE	Wandeln einer Datei vom großen in den kleinen Zeichensatz.

Spezifikation :

1 QUELLE: (obligat) Quelldatei, die den umzucodierenden Text enthält.

2 ZIEL: (obligat) Datei, auf die der umcodierte Text geschrieben wird.

3 MODUS: (optional) Art der Wiedergabe nicht darstellbarer Zeichen.
-STD-: werden als Ausrufezeichen dargestellt
* Voreinstellung
nnn: natürliche Zahl, gibt den Dezimalwert des Ersatzsymbols im Zentralcode an.
HEXA: nicht darstellbare Zeichen werden zweizeilig mit dem zweistelligen Hexadezimalwert ausgegeben.

Kommando der RRZE-Programmbibliothek	anlagenspezifische Voreinstellung : MODUS=-STD- SONST= -

Einschränkung :

Wirkung :

Eine Oktadendatei beliebigen Typs mit nicht mehr als 136 Satzelementen wird satzweise gelesen, im DC1 nicht darstellbare Zeichen werden entsprechend der Angabe bei Modus ersetzt und das Ergebnis auf der unter Ziel angegebenen Datei abgelegt. Die Quelldatei bleibt unverändert.

Kleine Buchstaben werden immer auf große abgebildet. Bei der Angabe MODUS=-STD- oder einer dreistelligen Dezimalzahl werden alle im DC1 nicht darstellbaren Zeichen durch das Ausrufezeichen bzw. das angegebene Zeichen ersetzt. Bei der Angabe MODUS=HEXA werden alle im DC 1 nicht darstellbaren und Sonderzeichen, die in DC1 und DC2 nicht übereinstimmen, zweizeilig mit dem zweistelligen Hexadezimalwert ausgegeben. Die Zieldatei erhält dann die doppelte Satzzahl der Quelldatei.

Beispiel:

Auf der LF-Datei GROSSZS steht ein Text im großen Zeichensatz (DC2), der auf einem Drucker im kleinen Zeichensatz ausgegeben werden soll. Eine Datei KLEINZS wird für die Aufnahme des umcodierten Textes kreiert.

```
/2XBA,BEN=... .
/LFANMELDE,GROSSZS
/DATEI,KLEINZS,RAM,U200,M1360
/LFANMELDE,UNRZPB.GED
/GEDAECHTNIS,GED,EIN
/GKWANDLE,QUELLE=GROSSZS,ZIEL=KLEINZS,MODUS=144
/TKOP,KLEINZS,ZIEL=DR(1,0)-DC1
/2XEN.
```

In dem Beispiel werden nicht darstellbare Zeichen durch "+" wiedergegeben.

Fehlermeldungen:

SSR-FEHLERAUSGANG 1024 MAL AUSGESPRUNGEN.
Ursache: Quelle nicht zum Lesen angemeldet.

DATEI NICHT GEFUNDEN,

WRITE, DATEI 2

Ursache Zieldatei nicht kreiert bzw. angemeldet

++++++ FALSCHE ZEICHENANGABE BEI MODUS

Ursache: evtl. wurde als Ersatzsymbol ein Zeichen angegeben, das nicht dem DC1 angehört.

++++++ FEHLER GKWANDLE, SATZ n LAENGER ALS 136 ZEICHEN.

PASCAL: Compiler for PASCAL 6000 - 3.4.
(ASCII-Version 2.1 vom 15.6.76)

Autor: N. Wirth ETH ZÜRICH
Implementation: U. Amman ETH ZÜRICH

PASCALS: Compiler for a Subset of Standard-PASCAL.
(ASCII-Version vom 1.3.76)

Autor: N. Wirth ETH Zürich

XREF: Cross Reference Generator for PASCAL Programs.

Autor: N. Wirth (10.2.76)

DECODE: List binary code in COMPASS-form.

Autor: N. Wirth (10.2.76)

PASCAL,PASCALS und die Hilfsprogramme XREF und DECODE sind an der ETH-Zürich für die Control Data Computer Serie 6000 mit dem Betriebssystem SCOPE 3.4 entwickelt worden. Nach geringfügigen Änderungen an Systemschnittstellen konnte PASCAL am Regionalen Rechenzentrum Erlangen an der CD-CYBER 172 mit dem Betriebssystem NOS 1.0 adaptiert werden. Da am Regionalen Rechenzentrum mit dem Lochkartencode KC2 gearbeitet wird, wurde die ASCII - Version des PASCAL - Compilers installiert. (Unterschiede KC2 - ASCII siehe 2.)

1. Aufruf

1.1 Procedure - File PASCALI

```
GET,PASCALI/UN=UNRZPB.
CALL(PASCALI(FL=Speicherplatz)
```

Nach dem Aufruf von PASCALI stehen die Compiler PASCAL und PASCALS und die Hilfsprogramme XREF und DECODE zur Verfügung. PASCALI führt folgende Kommandofolge aus:

```
COMMENT.-BEGIN PASCALI-
RETURN,PASCAL,PASCLIB,PASCALS,XREF,DECODE.
ATTACH,PASCAL/UN=UNRZPB.
ATTACH,PASCLIB/UN=UNRZPB.
ATTACH,PASCALS/UN=UNRZPB.
ATTACH,XREF=PASCALX/UN=UNRZPB.
ATTACH,DECODE=PASCALD/UN=UNRZPB.
LIBRARY,PASCLIB.
RFL,FL.
REDUCE,-.
COMMENT.-END PASCALI-
```

/LIBRARY,PASCLIB./ ermöglicht die Ausführung eines vom PASCAL-Compiler erzeugten Binärfiles (z.B.: LGO!) ohne ein zusätzliches Kommando:/LDSET,LIB=PASCLIB./.
PASCLIB enthält die Object-Time Routinen, die zur Laufzeit benötigt werden.
/RFL,FL./ setzt den für den PASCAL-Compiler nötigen Speicherplatz, /REDUCE,-./ verhindert die Reduzierung des Speichers. (Der PASCAL-Compiler erweitert seinen Stack).
Nach der PASCAL-Übersetzung sollte mit /REDUCE./ die automatische Speicherplatzverwaltung wieder eingeschaltet werden.
Statt des Procedure-Files PASCALI können die für die Übersetzung und Ausführung nötigen Files einzeln angesprochen werden.

1.2 Aufruf des PASCAL-Compilers und Ausführung eines PASCAL-Programms im Batch

```
P1,CM60000.
USER,UNRZXY.
CHARGE, UNRZXY,UNRZXY.
GET,PASCALI/UN=UNRZPB.
CALL(PASCALI(FL=60000)
PASCAL.
REDUCE.
LGO.
-EOR-
PROGRAM P1(INPUT,OUTPUT);
...
END.
-EOR-
    ...
    Daten für P1
    ...
-EOI-
```

Der PASCAL-Compiler ist in PASCAL geschrieben und beginnt mit dem PROGRAM-Statement:
PROGRAM PASCAL (INPUT,OUTPUT,LGO);
Im Standardfall entfällt der INPUT-File die Programmquelle, die Programmliste wird auf OUTPUT (Drucker) ausgegeben und der erzeugte Code auf dem LGO-File abgelegt. Durch Angabe von Filenamen beim Aufruf des PASCAL-Compilers und/oder des Programms P1 können die vorgegebenen Filenamen geändert werden:

```
GET,P1QUEL.                    (P1QUEL enthält Quellprogramm P1)
GET,P1DAT.                     (P1DAT  enthält Daten für P1)
GET,PASCALI/UN=UNRZPB.
CALL(PASCALI(FL=60000)
PASCAL,P1QUEL,,P1LGO.
```

```
REDUCE.
SAVE,P1LGO.
P1LGO,P1DAT.
-EOI-
```

Der PASCAL-Compiler übersetzt das Quellprogramm von T1QUEL, die Liste wird auf OUTPUT erzeugt und das Binärprogramm auf dem File P1LGO.P1 liest die Eingabedaten von P1DAT, die Ergebnisse erscheinen auf OUTPUT.
Weitere Informationen hierzu und zu den Compiler options, compiler messages und run-time messages mit Post-Mortem Dump siehe PASCAL-User Manual S. 100.

1.3 PASCAL im Dialog

Der PASCAL-Compiler arbeitet nicht interaktiv. PASCAL-Programme müssen zunächst z.B. mit einem Editor auf einen File geschrieben und können von dort übersetzt werden.

Beispiel (nach LOGIN):

```
EDIT,P1
ADD
PROGRAM P1(INPUT,OUTPUT);
...
END
(Ende EDIT)
```

} Siehe Time-Sharing und Text-Editor Manual

```
BATCH                          (Umschalten auf Batch- Subsystem)
($RFL,20000.)                  (System-Antwort)
/GET,PASCALI/UN=UNRZPB         (/kommt von System)
/CALL(PASCALI(FL=60000)
(COMMENT.-END PASCALI-)        (System)
/PASCAL,P1
....Liste von P1.....
(-PASCAL VERSION 2.1...)       (PASCAL-Meldung bei syntaktisch
                                richtigem Programm)
/GET,DAT1                      (Datenfile)
```

```
/REDUCE
/LGO,DAT1                          (P1 liest Daten von DAT1)
...OUTPUT von P1...
(-LOAD FL fwa RUN FL lwa)          (System: Speicherbelegung P1)
/SAVE,P1                           (P1 wird Permanent-File)
```

Syntaxfehler werden wie im Batch gekennzeichnet und ebenso eine "Error summary" ausgegeben. Im Fehlerfall, insbesondere bei Laufzeitfehlern, sollte der DAYFILE angesehen werden:
DAYFILE,OP=I (OP=I: Beginn beim letzten DAYFILE-Kommando)

2. PASCAL - Zeichensatz

Installiert ist die ASCII Version mit 64 Zeichen, die Zeichen : und % werden als gleich angesehen.
Aus der folgenden Tabelle sind die Unterschiede zwischen dem ASCII - und dem KC2 - Zeichensatz ersichtlich. Da die Drucker der CYBER z. Z. noch nicht mit ASCII-Ketten ausgerüstet sind, ist ebenfalls der CDC-Zeichensatz (KC3) aufgeführt. Der Zeichensatz der CYBER-Terminals entspricht dem ASCII-Code.

2.1 Unterschiede in den Zeichensätzen

$Code_{10}$	KC2	ASCII	CDC(KC3)	
49	¢	[	[	KC2 ≠ ASCII
50	!	]	]	
54	|	!	∨	
61	028	\	≥	
62	¬	^	¬	
48	#		=	KC2 = ASCII
52	"		≠	
53	_		↱	
55	&		^	
56	'		↑	
57	?		↓	
60	@		≤	

2.2 Von der PASCAL-Notation abweichende PASCAL-Symbole

PASCAL-Symbol	Darstellung		
	KC 2	ASCII	CDC
[	¢	[	[
]	!	]	]
{	(*	(*	(*
}	*)	*)	*)
↑	¬	^	¬

Alle anderen Symbole entsprechen der PASCAL-Notation im Report von N. Wirth.

3. PASCALS - eine Untermenge von Standard-PASCAL

PASCALS ist ein Compiler / Interpreter mit eigenem Monitor, der das übersetzte Programm lädt und ausführt. PASCALS wurde für Unterrichtszwecke geschrieben, um möglichst vielen Studenten eine intensive und erfolgreiche Programmierausbildung bieten zu können ohne den Computer, was Compilierungszeit und Speicherbelegung betrifft, über Maßen zu beanspruchen. Da PASCALS eine echte Untermenge von Standard-PASCAL ist, können PASCALS-Programme auch vom PASCAL-Compiler übersetzt werden.

Beispiel:

```
PS1,CM20000.
USER,UNRZXY.
CHARGE,UNRZXY,UNRZXY.
GET,PASCALI/UN=UNRZPB.
CALL(PASCALI(FL=20000)
PASCALS.
-EOR-
PROGRAM PS1 (INPUT, OUTPUT);
... PASCALS-Programm PS1
END.
```

```
-EOR-
...Daten für PS1...
-EOI-
```

PASCALS hat die Standardfiles INPUT (Quelle), OUTPUT (Liste), die beim Aufruf umbenannt werden können (siehe PASCAL). Der erzeugte Binärcode wird direkt ausgeführt, das PASCALS-Programm PS1 kann nur die Standardfiles INPUT und OUTPUT haben. (PASCALS im Dialog: wie PASCAL)

4. Die PASCAL-Hilfsprogramme XREF und DECODE

4.1 XREF - Cross Reference Generator for PASCAL-Programs

XREF erzeugt eine Cross-Reference-Liste von PASCAL- und PASCALS-Programmen.

Beispiel:

```
PX1,CM60000.
USER,UNRZXY.
CHARGE,UNRZXY,UNRZXY,
GET,PASCALI/UN=UNRZPB.
CALL(PASCALI(FL=60000)
XREF.
BKSP,INPUT.
PASCAL.
REDUCE.
LGO.
-EOR-
PROGRAM PX1 (INPUT,OUTPUT).
END.
-EOR-
...Daten für PX1...
-EOI-
```

XREF - Standardfiles: INPUT, OUTPUT

Wenn nach XREF noch der PASCAL-Compiler gerufen wird, muß der INPUT-File nun 1 Record zurückgesetzt werden (bei anderen Files nicht nötig).

4.2 DECODE - List binary code in COMPASS-form

Der PASCAL-Compiler erzeugt keinen Zwischencode, sondern direkt den vom CYBER-Loader verständlichen relocatable binary code. DECODE listet diesen Code in COMPASS-Form auf. DECODE ist für PASCALS nicht verwendbar.

Beispiel:

```
PD1,CM60000.
USER,UNRZXY.
CHARGE,UNRZXY,UNRZXY.
GET,PASCALI/UN=UNRZPB.
CALL(PASCALI(FL=60000)
PASCAL.
REDUCE.
DECODE.
LGO.
-EOR-
PROGRAM PD1(INPUT,OUTPUT);
       ...
END.
-EOR-
...Daten für PD1...
-EOI-
```

DECODE - Standardfiles: LGO, OUTPUT.

5. Literatur

Kathleen Jensen and Niklaus Wirth,
PASCAL User Manual and Report

Lecture Notes in Computer Science, 18,
Springer-Verlag Berlin, Heidelberg, New York

Niklaus Wirth,
PASCAL-S: A Subset and its implementation,
Berichte des Instituts für Informatik der
ETH - Zürich, Heft 12, Juni 1975

Urs Amman,
On Code Generation in a PASCAL Compiler,
Institut für Informatik der ETH - Zürich,
Heft 13, April 1976

CONTROL DATA - CYBER 170
NOS Version 1 - Reference Manual
LOADER Version 1 - Reference Manual
NOS Version 1 - Time - Sharing User's Reference Manual
and Text Editor Reference Manual

6. Hinweise zur Umstellung von TR 440 - PASCAL-Programmen

Es entfallen die Implementations- und Spracheinschränkungen des TR 440-PASCAL-Compilers. Im einzelnen muß Folgendes berücksichtigt werden:

1. Ein Programm muß durch das PROGRAM-Statement mit mindestens einem formalen Parameter, und zwar OUTPUT, eingeleitet werden.
2. Das Zeichen EOL ist sinngemäß durch die Operatoren READLN, WRITELN oder EOLN zu ersetzen.
3. Die Funktion LINENUMBER existiert nicht.
4. Die folgenden Symbole sind zu ersetzen:

	PASCAL		TR 440 (KC4)	KC2	ASCII
Kommentarklammern	{	:	12-0 bzw. / *	durch (*	(*
	}	:	11-0 bzw. * /	durch *)	*)
Negation	NOT	:	11-0-6-8	durch NOT	NOT
Ungleich	<>	:	%	durch <>	<>
Pointer	↑	:	12-11	durch ¬	^

5. Der Datentyp SET kann maximal 58 Elemente enthalten (SET OF CHAR geht nicht).

Diese Liste ist sicherlich nicht vollständig und könnte durch PASCAL-Benutzer korrigiert und erweitert werden. Bei der Umstellung einiger TR 440-PASCAL-Programme erwies sich der CYBER-Editor als sehr nützlich.

REGIONALES RECHENZENTRUM ERLANGEN	R R Z E PROGRAMMBIBLIOTHEK C Y B E R	Code: Q 0 Programm: GET 440 28.6.77

Autor: G. Seybold (RRZE)

Sprache : FORTRAN/COMPASS

GET 440: Übernahme von Daten vom TR 440 zur CYBER

Der Austausch von Daten zwischen TR 440 und Cyber kann mit Hilfe von Steuerkarten organisiert werden, wenn Bänder im IBM-Format als Datenträger benutzt werden. Da diese Art der Datendarstellung an keinem der beiden Rechner der Normfall ist, wird diese Steuerkartenprogrammierung häufig als lästig empfunden. Zur Vereinfachung wurde daher das im folgenden beschriebene Programm entwickelt, das es erlaubt, Magnetbänder im TR 440 -Interncode an der Cyber zu lesen und auf Dateien zu kopieren.

Aufruf des Programms:

```
GET,GET440/UN=UNRZPB.
GET440,exdkz,lfn1,...,lfnn.
```

Erforderliches Format des Eingabebandes:

TR 440 - Interncode (ZC1)
Blockung 256 Worte/Block

Anforderungen an Dateien, die kopiert werden sollen:

Oktadendatei

Das Programm erlaubt es nicht, Dateien etwa mit Satzbau W, wie sie von Fortran binary write erzeugt werden, zu konvertieren.

Solche Dateien oder auch gesicherte Bibliotheken dürfen sich jedoch auf dem Band befinden, sofern sie nicht kopiert, sondern nur übersprungen werden sollen.

Empfehlung:

Am einfachsten läßt sich ein den Anforderungen genügendes Band mit dem SICHERE - Kommando aus einer Texthaltungsdatei erzeugen.

Ausgabe des Programms:

Die auf dem Band stehenden Dateien werden der Reihe nach konvertiert auf die angegebenen Files (lfn_1 bis lfn_n) geschrieben. Wenn für eine Datei kein entsprechender lfn - Parameter angegeben wird, so wird die Datei nicht konvertiert, sondern übersprungen und braucht dann auch keine Oktadendatei zu sein.

Beispiel:

Am TR 440:

```
I2XBA,BEN=UNRZVF,BGB=1□.
□LFAN.,BEISPIEL
□SICHERE,BEISPIEL,MB(ARBEIT)
I2XEN□.
```

An der Cyber:

```
G4,CM10000.
USER,UNRZVF.
CHARGE,UNRZVF,UNRZVF.
GET,GET440/UN=UNRZPB.
GET440,ARBEIT,EXAMPLE.
SAVE,EXAMPLE.
```

```
REWIND,EXAMPLE.
FTN,I=EXAMPLE.
-EOI-
```

Betriebsmittelbedarf:

CM 10000.
ca 2 Sekunden CPU-Zeit für 1000 zu konvertierende Karten.

Vorsicht:

Das Programm ist nur spärlich getestet!

Regionales	RRZE	LIBMOD / LIBCALL
Rechenzentrum	Programmbibliothek	Bearb.: H. Cramer
Erlangen	CYBER	Datum: 2.7.77

Benutzereigene Bibliotheken an der CYBER

Die folgenden Prozeduren LIBMOD und LIBCALL sollen dem Benutzer die Verwaltung und den Aufruf seiner übersetzten Programme erleichtern. Mit LIBMOD lassen sich Records (Programme)vom Typ relocatable, absolute und Overlay und records vom Typ text verarbeiten, mit LIBCALL können sie von einer Bibliothek abgerufen werden.

1. LIBMOD

1.1 LIBMOD - Aufruf

```
GET,LIBMOD/UN=UNRZPB.
CALL(LIBMOD(LIB=libname,LGO=lfn1,ADD=lfn2,oo=lfn3)
```

libname	Name der Bibliothek Default: LIB
lfn1	Name des Files, von dem Records vom Typ relocatable (Programm) verarbeitet werden. Default: LGO
lfn2	Name des Files, von dem relocatable, absolute und Overlay Programme und Texte in die Bibliothek eingearbeitet werden, ebenso andere Bibliotheken. Default: ADD
lfn3	Name des INPUT-Files für zusätzliche LIBEDIT-Steuerkarten Default: oo, d.h. Programme auf den Files LGO und ADD ersetzen Programme gleichen Namens auf der Bibliothek, neue Programme werden der Bibliothek hinzugefügt (s. 1.)

1.2 LIBMOD-Kurzbeschreibung

-Die Bibliothek LIB wird angemeldet (GET), falls dies nicht schon vom Benutzer getan wurde. Existiert LIB noch nicht, wird sie angelegt (REPLACE).

-Die Bibliothek LIB (auch leere Bibliothek!) wird ohne Directory auf einen Scratch-File kopiert.

-Von LGO werden die Records vom Typ relocatable (REL) extrahiert, von ADD die Records vom Typ relocatable (REL), absolute (ABS), overlay (OVL) und text (TEXT).

-Mit LIBEDIT werden die von LGO und ADD gelesenen Records in LIB eingebaut. Records gleichen Namens werden auf LIB ersetzt, neue hinzugefügt. Bei Setzen des OO-Parameters können weitere LIBEDIT-Funktionen ausgeführt werden.

-LIBGEN erzeugt eine neue user library, ihr Inhalt wird aufgelistet.

-Die Ausgangsversion LIB erhält den Namen OLDLIB, die modifizierte Kopie den Namen von LIB.

-OLDLIB ist nach dem Aufruf von LIBMOD als lokaler File vorhanden, der zur Sicherheit permanent angelegt werden könnte (Plattenspeicherproblem).

-LIB wird als indirekter Permanentfile angelegt (REPLACE).

-Inhalt und Filebeschreibung von LIB werden gelistet (CATALOG,CATLIST).

Anmerkung: Aus organisatorischen Gründen enthält jede LIBMOD-Bibliothek den Record LIBMOD vom Typ REL.

1.3 Zusätzliche LIBEDIT-Steuerkarten

Bei fehlendem oo-Parameter wird nur die Steuerkarte:
*BEFORE *,REL/*,ABS/* ,OVL/* ,TEXT/*
abgearbeitet. Es können zusätzliche LIBEDIT-Directives mit oo=lfn angegeben werden. Um Programme einer Bibliothek löschen zu können, benötigt man die *DELETE-Directive (Abk.: *D):

*DELETE type/rname1,type/rname2,...,type/rnamen

Die angegebenen Records vom Typ type mit den Namen rname1 bis rnamen werden gelöscht.

*DELETE type1/rname1-type2/rname2

Die Gruppe der Records von rname1 vom Typ type1 bis rname 2 vom Typ type2 werden gelöscht.
Bei gleichem Typ kann die 2. Typ-Angabe fehlen.

Beispiel:

```
CALL(LIBMOD(LIB=TESTLIB,oo=INPUT)
-EOR-
*DELETE REL/M1-S8
```

Die Records M1 bis S8 vom Typ REL werden auf TESTLIB gelöscht.

Weitere LIBEDIT-Directives siehe NOS-Reference Manual Seite 1-C-1. Es ist darauf zu achten, daß diese nicht in Konflikt mit obiger geraten.

1.4 LIBMOD-Records

In der folgenden Tabelle ist aufgeführt, welche Records von LIBMOD verarbeitet werden können und wie auf diese zugegriffen werden kann. Der Zugriff kann auch, bis auf das Zuladen von Unterprogrammen, mit der in 1.5 beschriebenen Prozedur LIBCALL geschehen, ebenso können Records mit GTR aus einer LIBMOD-Bibliothek kopiert werden.

Record	Typ	LIBMOD INPUT-File	Zugriff und Ausführung	Anmerkung
Unter-programm	REL	LGO ADD	1. LDSET,LIB=libname. LGO.	lokale Bibliothek, gültig für einen Aufruf. Hauptprogramm auf LGO.
			2. LIBRARY,libname. LGO.	globale Bibliothek,gültig für ganzen Job. Hauptprogramm auf LGO.
Haupt-pro-gramm	REL	LGO ADD	LIBLOAD,libname,programm. EXECUTE.	Benötigt das Hauptprogramm Unterprogramme,ist ein LDSET-bzw. LIBRARY-Statement erforderlich.
Haupt-pro-gramm	ABS	ADD	SLOAD,libname,programm. EXECUTE.	Achtung: nur sehr große und sehr oft aufgerufene Programme absolutieren. Die Plattenspeicherkosten sind im Vergleich zu den Rechenzeitkosten für das Laden eines relocatable Hauptprogramms sehr hoch!

Record	Typ	LIBMOD INPUT-File	Zugriff und Ausführung	Anmerkung
Main-Overlay (o,o)	ABS	ADD	SLOAD,libname,ovlname. EXECUTE.	Das Main-Overlay (o,o) ruft die primary und secondary overlays von der Bibliothek, falls im Overlay-Call der Bibliotheksname angegeben ist. Sollen mehrere Overlayprogramme auf einer Bibliothek gelagert werden, müssen die Overlays, außer Main (o,o), verschiedene Level-Nummern haben.
Text	TEXT	ADD	GTR,libname,lfn.TEXT/tname.	Der Record tname wird von der Bibliothek auf den lokalen File lfn kopiert. Recordname eines Textes sind die ersten 7 Zeichen des Textrecords. Ist der Text eine Prozedur, kann lfn als Procedure-File gerufen werden: CALL(lfn(....)

2. LIBCALL

Mit LIBCALL können Hauptprogramme, Overlays und Texte von einer LIBMOD-Bibliothek abgerufen werden, Unterprogramme müssen mit LDSET oder LIBRARY geladen werden.

2.1 LIBCALL-Aufruf

```
GET,LIBCALL/UN=UNRZPB.
CALL(LIBCALL(LIB=libname,PROG=rname,REL=rtype)
```

libname	Name der Bibliothek Default: LIB
rname	Name des Records (Programms) Default: PROG
rtype	Typ des Records Default: REL

2.2 LIBCALL-Kurzbeschreibung

-Der Record rname vom Typ rtype wird von LIB auf den lokalen File rname kopiert.

-LIB wird mit LIBRARY,libname zur globalen Bibliothek. So finden relocatable Hauptprogramme von der Bibliothek benötigte Unterprogramme.

-Hauptprogramme und Overlays können nach LIBCALL mit einem file name call aufgerufen werden: rname.

-Prozeduren (Text) können als Procedure-File aufgerufen werden: CALL(rname(....).

3. Beispiele LIBMOD/LIBCALL

Beispiel 1: Generieren einer LIBMOD-Bibliothek und Aufruf eines relocatable Hauptprogramms mit LIBCALL

```
JOB,...
USER,...
CHARGE,...
FTN.                                    M1,S1,S2(REL)   ⟶   LGO
GET,LIBMOD/UN=UNRZPB.
CALL(LIBMOD(LIB=TESTLIB)                LGO: M1,S1,S2(REL) ⟶ TESTLIB
GET,LIBCALL/UN=UNRZPB.
CALL(LIBCALL(LIB=TESTLIB,PROG=M1)       M1(REL) ⟶ local file M1
M1.                                     Aufruf M1
-EOR-
      PROGRAM M1 (INPUT,OUTPUT)
      CALL S1
      CALL S2
      END
      SUBROUTINE S1
      ....
      END
      SUBROUTINE S2
      ....
      END
-EOR-
  ...Daten...
-EOI-
```

LIBMOD übernimmt vom LGO-File die Programme M1,S1 und S2 und baut sie in die leere Bibliothek TESTLIB als relocatable records (REL) ein. TESTLIB wird mit REPLACE als indirekter Permanentfile gesichert. Auf TESTLIB kann auch mit LDSET,LIBRARY und LIBLOAD zugegriffen werden:

```
FTN.
GET,TESTLIB.
LDSET,LIB=TESTLIB.
LGO.
-EOR-
      PROGRAM M1
      ...
      CALL S1
      CALL S2
      ....
      END
-EOR-
  ...Daten...
-EOI-
```

M1 wird auf LGO übersetzt und von TESTLIB werden S1 und S2 zugeladen.

Beispiel 2: Korrektur einer LIBMOD-Bibliothek

Die Subroutine S1 soll korrigiert werden.

```
GET,LIBMOD/UN=UNRZPB.
GET,LIBCALL/UN=UNRZPB.
FTN.
CALL(LIBMOD(LIB=TESTLIB)
CALL(LIBCALL(LIB=TESTLIB,PROG=M1)
M1.
-EOR-
      SUBROUTINE S1
      ....
      END
-EOR-
  ...Daten...
-EOI-
```

S1 wird gelöscht und durch die neue Version auf LGO ersetzt.

Beispiel 3: Aufnahme und Aufruf eines absoluten Programms

```
GET,LIBMOD/UN=UNRZPB.
GET,LIBCALL/UN=UNRZPB.
FTN.                                    A1,AS1,As2(REL)  ──► LGO
LOAD,LGO.
NOGO,ADD.                               A1(ABS) ──► ADD
CALL(LIBMOD(LIB=TESTLIB)                A1,AS1,AS2(REL),A1(ABS)──►TESTLIB
CALL(LIBCALL(LIB=TESTLIB,PROG=A1,REL=ABS)
A1.
-EOR-
      PROGRAM A1
      ...
      CALL AS1
      ...
      CALL AS2
      ...
      END
      SUBROUTINE AS1
      ...
      SUBROUTINE AS2
      ...
-EOI-
```

TESTLIB enthält die Programme M1,S1,S2 (REL), hinzukommen jetzt A1,AS1,AS2 (REL) und A1 (ABS). A1 kann sowohl relocatable als auch absolut aufgerufen werden, im Beispiel absolut. Wenn A1, AS1 und AS2 nicht relocatable in TESTLIB aufgenommen werden sollen, ist ist vor CALL(LIBMOD... RETURN,LGO zu geben.

Beispiel 4: Löschen von Programmen

```
GET,LIBMOD/UN=UNRZPB.
CALL(LIBMOD(LIB=TESTLIB,oo=INPUT)
-EOR-
*DELETE REL/A1,REL/AS1,REL/AS2
-EOI-
```

Die Programme A1, AS1 und AS2 werden gelöscht. Falls A1, AS1 und AS2 auf der Bibliothek hintereinander liegen, könnte folgende Anweisung gegeben werden:

```
*DELETE REL/A1-AS2
```

Die Reihenfolge der Programme ist dem vorhergehenden LIBMOD-Lauf zu entnehmen. Das Löschen von Programmen (Records) kann gleichzeitig mit einer Neuaufnahme von Programmen erfolgen. In jedem Fall wird der Inhalt des LGO- und ADD-Files in die Bibliothek aufgenommen.

Beispiel 5: Aufnahme und Aufruf eines Overlays

```
GET,LIBMOD/UN=UNRZPB.
GET,LIBCALL/UN=UNRZPB.
FTN.                                    OVM,OVP,OVS(REL) ---> LGO
LOAD,LGO.
NOGO.                                   OVM,OVP,OVS(ABS) ---> XYZ
RETURN,LGO.
CALL(LIBMOD(LIB=TESTLIB,ADD=XYZ)
CALL(LIBCALL(LIB=TESTLIB,PROG=OVM,REL=ABS)
OVM.
-EOR-
     OVERLAY(XYZ,o,o)                   Main Overlay
     PROGRAM OVM(INPUT...)
     ...
     CALL OVERLAY(7HTESTLIB,1,o)
     ...
     END
     OVERLAY(,1,o)
     PROGRAM OVP                        Primary Overlay
     ...
     CALL OVERLAY(7HTESTLIB,1,1)
     ...
     END
     OVERLAY(,1,1)                      Secondary Overlay
     PROGRAM OVS
     ...
     END
-EOR-
  ...Daten...
-EOI-
```

LIBMOD nimmt die Overlays OVM,OVP,OVS in absoluter Form auf. LIBCALL kopiert das Main Overlay OVM auf den lokalen File OVM, von wo es gestartet wird. OVM (o,o) findet OVP (1,o) auf TESTLIB, ebenso OVP (1,o) das secondary overlay OVS (1,1) auf TESTLIB.

Bei mehreren Overlay Programmen müssen die Level Nummern verschieden sein, außer beim Main Overlay, das durch den Programmnamen identifiziert wird.

Beispiel 6: Aufnahme und Aufruf eines Prozedurtextes

```
GET,LIBMOD/UN=UNRZPB.
GET,LIBCALL/UN=UNRZPB.
COPYBR,INPUT,ADD.                          P1(TEXT) ⟶ ADD
CALL(LIBMOD(LIB=TESTLIB)
CALL(LIBCALL(LIB=TESTLIB,PROG=P1,REL=TEXT)
CALL(P1)
-EOR-
P1
CATLIST.
KONTO.
COMMENT. - END P1 -
-EOI-
```

LIBMOD nimmt den Textrecord P1 von ADD in TESTLIB auf, LIBCALL kopiert ihn von TESTLIB auf P1, von wo er als Prozedur gerufen wird. P1 kann nicht direkt vom INPUT-File aufgenommen werden: CALL(LIBMOD(LIB=TESTLIB,ADD=INPUT) geht nicht!

Beispiel 7: Aufnahme einer anderen Bibliothek

```
GET,LIBMOD/UN=UNRZPB.
GET,YOURLIB/UN=YOUR.
CALL(LIBMOD(LIB=TESTLIB,ADD=YOURLIB)
-EOI-
```

Alle Records vom Typ REL,ABS,OVL und TEXT der Bibliothek YOURLIB werden in TESTLIB eingefügt. Ebenso können von Nicht-Bibliothek-Files Records dieser Typen aufgenommen werden. Vorsicht: Records gleichen Namens werden durch die neuen ersetzt.

STICHWORTVERZEICHNIS DER BENUTZERINFORMATIONEN DES RRZE

STICHWORTVERZEICHNIS DER BENUTZERINFORMATIONEN DES RRZE

STICHWORTVERZEICHNIS DER BENUTZERINFORMATIONEN DES RRZE

STICHWORTVERZEICHNIS DER BENUTZERINFORMATIONEN DES RRZE