KfK 4587
Juli 1989

SERVUS
Ein System von Dateien und Prozeduren zur Speicherung und Präsentation von Rechenergebnissen, die Funktionen (einer Variablen) sind

— Version V, April 1989 —

K. Thurnay
Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

KfK 4587

SERVUS

Ein System von Dateien und Prozeduren zur Speicherung und Präsentation von Rechenergebnissen, die Funktionen (einer Variablen) sind

- Version V, April 1989 -

K. Thurnay

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Zusammenfassung

Das Funktions-Handhabungssystem SERVUS dient dazu, die - bei einer Rechnung anfallenden - Funktionen für spätere Verwendung in geeigneten Dateien zu speichern. Das System ermöglicht es, die gespeicherten Funktionen in einem nachfolgenden Schritt weiterzuverarbeiten oder zu präsentieren, als Zahlenkolonnen, als Kurvenscharen oder als Funktionsoberflächen.

Es wird erklärt,
wie man eine SERVUS-Datei anlegt,
wie man Funktionen in diese Datei speichert,
wie man Funktionen aus dieser Datei liest,
wie man sich von der Datei einen Katalog beschafft,
wie man eine Kopie der Datei anlegt,
wie man Funktionen einer Datei manipuliert,
wie man Funktionen der Datei ausdrucken läßt und
wie man Funktionen der Datei als Kurven oder als 3D-Oberflächen zeichnen läßt.

The Function-Manipulating-System SERVUS.

Abstract.

SERVUS is a service-utility to store functions resulting from a numerical calculation in specially fitted datasets for further use. SERVUS enables the user, either to manipulate the stored functions, or to display them in a second step. The functions can be presented as columns of numbers, as a family of curves or as surfaces of functions.

The paper describes the steps, needed
for allocating a SERVUS-dataset,
for storing a function in this set,
for reading a function from this dataset,
for cataloguing the functions, stored in the set,
for copying a SERVUS-dataset,
for manipulating stored functions,
for printing stored functions and
for drawing two or three-dimensional pictures of them.

Inhaltsverzeichnis

1.0 Das Funktions-Handhabungssystem SERVUS. 1
1.1 Einführung. 1
1.2 Benötigte Hilfsmittel. 2

2.0 Über SERVUS-Dateien. 5
2.1 Aufbau. 5
2.2 Einschränkungen. 5
2.3 Die Schreib-Normen der SERVUS-Dateien. 6

3.0 Eine SERVUS-Datei wird angelegt. 9

4.0 Eine SERVUS-Datei wird mit einer Endzeile versehen. 11

5.0 Ein- und Ausgabe der Funktionen bei SERVUS-Dateien. 13
5.1 Öffnung der Dateien. 13
5.2 Schreiben in eine Datei. 14
5.3 Schreiben im Dialogverfahren. 16
5.4 Lesen aus einer Datei. 17
5.5 Benötigte Sonder-Steuerkarten. 18
5.6 Beispiele. 19

6.0 Eine SERVUS-Datei wird gesichtet und kopiert. 23
6.1 Das Verfahren. 23
6.2 Sichten bzw. Kopieren im Stapel-Betrieb. 24
6.3 Sichten/Kopieren im Dialog. 26

7.0 Änderungen der Funktionen einer SERVUS-Datei. 27
7.1 Einige Funktionen der Datei werden renormiert oder gestrichen. 27
7.2 Die Reihenfolge der Funktionen in der Datei wird abgeändert. 28
7.3 Mehrere Funktionen der Datei werden zu einer neuen Funktion zusammengefügt. 30
7.4 Einige Funktionen der Datei werden invertiert. 31
7.5 Eine Funktionenschar der Datei wird transponiert. 33
7.6 Funktions-Kenngrößen werden abgeändert. 35
7.7 Umwandlung der Funktionen in ihren Logarithmen , Prüfung des monotonen Wachstums. 37

8.0 Funktionen aus einer SERVUS-Datei werden ausgedruckt. 39

9.0 Funktionen aus einer SERVUS-Datei werden als eine Kurvenschar gezeichnet. . 41
9.1 Über das Abbildungsverfahren. 41
9.2 Stapel-Verfahren für die Kurven-Darstellung. 41
9.3 Kurven-Darstellung im Dialog. 47

10.0 Funktionen aus einer SERVUS-Datei werden als eine Funktions-Oberfläche gezeichnet. 51
10.1 Das Verfahren. 51
10.2 Stapel-Verfahren für die Oberflächen-Darstellung. 51
10.3 Oberflächen-Darstellung im Dialog. 56

11.0 Eine Text-Datei wird zum GS-Bild umgewandelt. 59
11.1 Das Verfahren. 59
11.2 Spielregeln der Prozedur TEXIMA : 59
11.3 Die FORTRAN-TEXIMA-Prozedur 61

12.0 Die SERVUS-Kommandoprozeduren. 63
12.1 Über Dialoge im Kommando- und in FORTRAN-Mode. 63
12.2 Haupt-Prozeduren 64
12.2.1 Die Kommandoprozedur IDA 64
12.2.2 Die Kommandoprozedur ENDE 65
12.2.3 Die Kommandoprozedur FUNKIN 65
12.2.4 Die Kommandoprozedur KOPIER 66
12.2.5 Die Kommandoprozedur DATAS 69
12.2.6 Die Kommandoprozedur LOESCH 70
12.2.7 Die Kommandoprozedur DRUDA 72
12.2.8 Die Kommandoprozedur FIGUR 74
12.2.9 Die Kommandoprozedur DDFIG 75
12.2.10 Die Kommandoprozedur TEXIMA 77
12.2.11 Die Kommandoprozedur SVLOAD. 78
12.3 G.S.-Prozeduren 79
12.3.1 Die G.S.-Prozedur BRAUS 79
12.3.2 Die G.S.-Prozedur IMMANI 85
12.3.3 Die G.S.-Prozedur PICOR 88
12.3.4 Die G.S.-Prozedur PIXI 89
12.3.5 Die G.S.-Prozedur PODEL 90
12.3.6 Die G.S.-Prozedur PONUM 92
12.4 Hilfsprozeduren 94
12.4.1 Die Hilfsprozedur ADA 94
12.4.2 Die Hilfsprozedur ALOGS 95
12.4.3 Die Hilfsprozedur BEKI 97
12.4.4 Die Hilfsprozedur FEJ 98
12.4.5 Die Hilfsprozedur GLIEDER 99
12.4.6 Die Hilfsprozedur NEVE 100
12.4.7 Die Hilfsprozedur OBJEKT 101
12.4.8 Die Hilfsprozedur SERVIN 103
12.4.9 Die Hilfsprozedur SUSA 104
12.4.10 Die Hilfsprozedur TAGOK 105
12.4.11 Die Hilfsprozedur TUKOR 105

13.0 Literatur: 107

14.0 Abbildungen. 109

Anhang A. Die Hilfs-Datei "MEMORY" 123

Anhang B. Aufbau der Datei INR105.SERVUS.LOAD 125

Abbildungsverzeichnis

Abbildung 1. Funktionen eintragen. 110
Abbildung 2. Lesen-Schreiben. 111
Abbildung 3. PLOTEASY-Datei. 112
Abbildung 4. Katalog-Auszug. 113
Abbildung 5. Ausdruck-Spiegel-I. 114
Abbildung 6. Ausdruck-Spiegel-II. 115
Abbildung 7. Ausdruck-Spiegel-III. 116
Abbildung 8. Funktionen als Kurvenschar. 117
Abbildung 9. Aufbau des Bildrahmens. 118
Abbildung 10. Entwicklung der Gasgeschwindigkeiten im Kühlkanal 119
Abbildung 11. Entwicklung der Gasgeschwindigkeiten im Kühlkanal 120
Abbildung 12. Funktionen als Oberfläche I. 121
Abbildung 13. Funktionen als Oberfläche II. 122

Das Handbuch weist folgende wichtige Änderungen gegenüber der ursprünglichen Version (KfK 3911 , April 1985) auf:

Kap. 2.

- Die Einschränkungen bei der 3D-Darstellungen sind gelockert: es sind jetzt in einem Bild maximal 128 Funktionen mit je 256 Punkten darstellbar.

Kap. 5. und 6.

- Man kann jetzt auch Funktionen aus einer PLOTEASY-Datei in eine SERVUS-Datei umkopieren.
- Es gibt ein neues Verfahren zur Änderung der Punktzahl der Funktionen beim kopieren.
- Eine neue Prozedur, FUNKIN erlaubt die Herstellung oder Korrektur von Funktionen (einer SERVUS-Datei) im Vordergrund.

Kap. 7.

Es gibt zusätzliche Möglichkeiten der Datei-Bearbeitung : Funktionen einer Datei kann man jetzt

- renormieren,
- umordnen,
- zusammenfügen,
- invertieren und
- transponieren.
  Man kann sie in ihre
- dekadischen Logarithmen umwandeln und
- auf monotones Wachstum prüfen.
  Mann kan jetzt auch verschiedenen Funktions-Kenngrößen
- "KLASSE" , "NUMMER" , "NAMX" , "MASX" , ... ändern.

Kap. 9. und 10.

- Es besteht die Freiheit, beim Zeichnen verschiedene
  - Linienstärken und
  - Farben

  zu nutzen.
- Die Größe "INST" bei den Stapelprozeduren ist jetzt eine CHARACTER*8 Variable !.
- Die Dialogprozeduren sind verbessert und ausführlich beschrieben.
- Beim Zeichnen im Dialog kann man jetzt auch das Herstellungsdatum mit ins Bild einschreiben.
- Beim Zeichnen im Dialog kann man jetzt sowohl im DIN-Querformat als auch in Hochformat arbeiten.
- Die gespeicherte Bilder kann man jetzt auch in GDF-Format umwandeln und wie in /1/ beschrieben, mit dem IBM-3820-Blattdrucker zeichnen lassen.

Kap. 10.

- Bei der Flächendarstellung kann man jetzt die Funktionsebenen auch in unregelmäßigen Abständen hintereinander verlegen.

Kap. 11. Dieses Kapitel ist neu. Es beschreibt eine neue Prozedur, TEXIMA.

- TEXIMA verarbeitet Text-Dateien zur Bildern in Quer- oder Hochformat.

Kap. 12.

- Sämtliche Kommandoprozeduren sind überarbeitet und zuverlässiger geworden.
- Zusätzliche Prozeduren erleichtern das Ausdrucken , Kopieren und Löschen von Funktionen einer Datei.
- Vier neue Prozeduren, IMMANI , PICOR , PODEL und PONUM helfen jetzt beim Verwalten und Bearbeiten der gespeicherten Bilder.
- Die Kommandoprozedur BRAUS kann jetzt bis zu 24 Bilder eines "Buches" in verschiedenen DIN-Formaten zu einem der Zeichengeräte
  - XYNETICS
  - VERSATEC
  - Farb-VERSATEC

  schicken.
- Die Kommandoprozedur TEXT gibt es nicht mehr, das SERVUS-Handbuch kann man jetzt mit der Prozedur GML (s. /1/) ausdrucken lassen. Die Quelldatei heißt 'INR105.SERVUS.TEXT(SERVUS)'. 'INR105.SERVUS.TEXT(INFO)' erklärt die Handhabung der Datei.

1.0 Das Funktions-Handhabungssystem SERVUS.

1.1 Einführung.

Das Aufkommen elektronischer Rechner hat dazu geführt, daß man heute auch kompliziert zusammenhängende physikalische Vorgänge nachrechnen und verstehen kann. Die Rechengeschwindigkeit und Speicherkapazität heutiger Rechner erlauben es, diese Vorgänge in den Griff zu bekommen, indem man einfach alle zuständige Bewegungsgleichungen für ein großes Feld von Objektpunkten in einem bestimmten Zeitpunkt löst und dieses Verfahren mit einer kleinen Zeitverschiebung solange fortsetzt, bis der gewünschte Endzeitpunkt erreicht ist.

Leider ist der Ingenieur/Physiker - beim heutigen Stand der FORTRAN -Technik - noch gezwungen, bei der Vorbereitung dieser Rechenverfahren unverhältnismäßig viel Arbeit mit der Lösung von Verwaltungsaufgaben zu verbringen :

- der Speicherplatzbedarf für die beim Rechnen auftretenden Variablen muß im voraus festgelegt werden,
- die Variablen müssen vor der Rechnung ihre Anfangswerte erhalten,
- die Variablen, deren Entwicklung genauer verfolgt werden soll und die Art, wie ihre Zwischenwerte ausgedruckt werden sollen, müssen im voraus festgelegt werden, usw.

Insbesonders der letzte Punkt - der Ausdruck der Zwischenergebnisse - kostet viel Zeit und ist sehr lästig. Jedesmal, wenn eine neue Variable auftaucht, die man - um die Vorgänge zu verstehen - auch verfolgen möchte, beginnt aufs Neue die Grübelei, auf welche Seite, zusammen mit welchen anderen Größen man sie ausdrucken soll und die Herumplackerei mit den FORMAT-Anweisungen.

Eine weitere nebenwissenschaftliche Arbeit ergibt sich daraus, daß die vom Rechner gelieferten Ergebnisse für den Anwender noch nicht im richtigen, verwertbaren "Zustand" sind. Der Rechner liefert eine große, mehr oder weniger geordnete Zahlenmenge (mitsamt einigen hundert Gramm schwerhandhabbaren Endlospapier) mit der der Anwender in der Regel nicht viel anfangen kann. Um ein Bild von den sich abspielenden Vorgängen zu gewinnen, muß man diese Zahlengruppen zu Kurven, Kurvenscharen oder zu Funktionsoberflächen umwandeln, ein Vorgang, der sehr viel Arbeit erfordert.

Das Funktions-Handhabungsystem SERVUS dient dazu, wenigstens einen Teil dieser Verwaltung- und Nachbehandlungs-Arbeit dem Anwender abzunehmen.

Das Grundelement des Systems ist die SERVUS-Datei (s. "Über SERVUS-Dateien." auf Seite 5). Alle Variablen einer Rechnung, die man u. U. zum Verständnis braucht, kann man - anstatt sie im Rechenstrom direkt auszudrucken - als Funktionen in SERVUS-Dateien speichern. Mit Hilfe der SERVUS-Prozeduren kann man dann anschließend die gespeicherte Information präsentieren oder aber auch weiterbehandeln:

Einzelne Funktionen aus einer SERVUS Datei kann man

- löschen,
- renormieren,
- umordnen,
- zu eine neue Funktion zusammenfügen,
- invertieren,
- in eine andere SERVUS-Datei kopieren,
- einer laufenden Rechnung zur Verfügung stellen,
- gruppenweise in Standardformat ausdrucken,

- gruppenweise als Kurvenscharen oder als Oberflächen darstellen.

SERVUS-Dateien sind besonders handlich bei den Fortsetzungs-Rechnungen: am Ende einer Fortsetzung werden alle Variablen, die für die Festlegung des System-Zustandes notwendig sind, in eine SERVUS-Datei gespeichert. Die nächste Fortsetzung holt dann diese Funktionen aus der Datei und setzt sie als Anfangswerte der entsprechenden Variablen ein, usw.

SERVUS hilft dem Anwender auch, wenn er seine Druckausgaben in eine Veröffentlichung einbringen will. Falls er die Ergebnisse mit der TSO-Kommando OUT oder mit der EDIT-Kommando in der OUTLIST-Sitzung in eine, oder mehrere Dateien überträgt, hilft ihm SERVUS, anschließend ein DCF-fähiges Bild daraus zu machen.

1.2 Benötigte Hilfsmittel.

Die zu speichernden Funktionen werden mittels FORTRAN-Subroutinen (s. "Ein- und Ausgabe der Funktionen bei SERVUS-Dateien." auf Seite 13) in die SERVUS-Dateien eingetragen. Die Handhabung der Dateien erfolgt mit TSO-Kommandoprozeduren und/oder FORTRAN77-Programmen. Der Benutzer braucht somit keine neue Sprachen zu lernen.
Wichtig ! Achten Sie bitte beim Einschalten darauf, daß als LOGON Prozedur **F** oder **FV** gesetzt ist !

```
------------------------------ TSO/E  LOGON ------------------------------------
PF1/PF13 ==> Help    PF3/PF15 ==> Logoff    PA1 ==> Attention    PA2 ==> RESHOW
You may request specific HELP information by entering a '?' in any entry field.
   ENTER LOGON PARAMETERS BELOW:                    RACF LOGON PARAMETERS:

   USERID     ===> itb00x                           NEW PASSWORD ===>

   PASSWORD   ===>                                  GROUP IDENT  ===>

   PROCEDURE ===> FV

   ACCT NMBR ===> 0abc-xyz-p9x9y

   SIZE       ===> 4000

   PERFORM    ===>

   COMMAND    ===>


   ENTER AN 'S' BEFORE EACH OPTION DESIRED BELOW:

          -NOMAIL          -NONOTICE        -RECONNECT          -OIDCARD
```

Anderseits muß der Benutzer dem SERVUS-System einige Hilfs-Dateien bereitstellen. Neben der SERVUS-Dateien (wie man diese anlegt, s. "Eine SERVUS-Datei wird angelegt." auf Seite 9) braucht man folgende Datenträger:

- Eine gegliederte Datei (partitioned Dataset, s. /2/) tso000.MEMORY zur Aufnahme wichtiger Prozedur-Daten und -Parameter,
- Eine oder mehrere gegliederte Dateien tso000.buch1, tso000.buch2, ... zur Speicherung der gefertigten Bilder.

tso000.MEMORY ist - s. auch "Anhang A. Die Hilfs-Datei "MEMORY"" auf Seite 123 - eine standard TSO-Datei (FB / 3120 / 80). Falls sie fehlt, wird sie automatisch angelegt. Sie muß mindestens folgende Glieder (Members) enthalten :

LAFIGU , LADDFI , LATEXI	enthalten Bildrahmendaten zu den Prozeduren FIGUR, DDFIG und TEXIMA
ABB1 , ... , ABB4	füren bei den Prozeduren TEXIMA, FIGUR, FIGURB, DDFIG oder DDFIGB den Bildern vorgefertigte Texte zu.

Es empfiehlt sich deshalb diese Datei Glied für Glied aus der Datei TSO105.MEMORY zu übernehmen.

Die Buch-Dateien sind DCB.U-Dateien (Beispiel : 'INR105.GSBOOK' in "Funktionen aus einer SERVUS-Datei werden als eine Kurvenschar gezeichnet." auf Seite 41). Jedes gespeichertes Bild wird ein eigenständiges Glied dieser Datei. Für Näheres über die Buch-Dateien s. das GS-Handbuch /3/ . Schließlich braucht man noch die TSO-Kommandoprozeduren und die Stapel-(CNTL-)-Prozeduren zur Handhabung der Dateien und der Funktionen.

Die SERVUS-Kommandoprozeduren

IDA	(Kap. 3) ,
ENDE	(Kap. 4) ,
FUNKIN	(Kap. 5) ,
DATAS	(Kap. 6) ,
KOPIER	(Kap. 6) ,
LOESCH	(Kap. 7) ,
DRUDA	(Kap. 8) ,
FIGUR	(Kap. 9) ,
DDFIG	(Kap. 10) und
TEXIMA	(Kap. 11)

sowie die GS-Kommandoprozeduren,

BRAUS	- Ausgabe der Bilder -
IMMANI	- allgemeine Verwaltung eines Buches -
PICOR	- Ausgabe der Bildstruktur -
PODEL	- Löschen von Bildelementen -
PONUM	- Identifikation von Bildelementen -
PIXI	- allgemeine Arbeit mit GS -

die man zum Weiterbehandeln der Bilder braucht, kann man aus der Datei TSO105.SERVUS.CLIST übernehmen, indem man die entsprechende Glieder kopiert (s. "Die SERVUS-Kommandoprozeduren." auf Seite 63). Gleichzeitig sollte man die Hilfsprozeduren **ADA , ALOGS , BEKI , FEJ , GLIEDER , NEVE , OBJEKT , SERVIN , TAGOK , TUKOR** , die von die Kommandoprozeduren benötigt werden mitübernehmen.

Die Stapel-Prozeduren

KOPIERT	(Kap. 6) ,
LOESCHT	(Kap. 7) ,
FUEGT	(Kap. 7) ,
MISCHT	(Kap. 7) ,
INVERT	(Kap. 7) ,

TRANSP	(Kap. 7) ,
NEWNUM	(Kap. 7) ,
LOGAR	(Kap. 7) ,
DRUCKT	(Kap. 8) ,
FIGURB	(Kap. 9) und
DDFIGB	(Kap. 10)

sind in der Datei TSO105.SERVUS.CNTL. Die Jobkarten in den kopierten Prozeduren - sowie die entsprechenden Angaben in der Prozedurkopf der Hilfsprozedur **FEJ** - muß der Benutzer mit den eigenen Daten auffüllen.

Bemerkung zur Schreibweise der Prozedur-Parameter : in diesem Bericht werden Namen - die der Benutzer in einer Prozedur selber einsetzen/abändern muß - kleingeschrieben, die festgelegten Namen erscheinen dagegen in Blockschrift. Zum Beispiel wird, bei tso000.MEMORY vom Benutzer erwartet, daß er hier seine eigene Benutzernummer, z.B. TSO999 einsetzt :

```
tso000.MEMORY     ===>     TSO999.MEMORY
```

2.0 Über SERVUS-Dateien.

2.1 *Aufbau.*

Eine SERVUS-Datei dient dazu, Funktionen F(X) - die bei einer laufenden Rechnung anfallen - für eine spätere Verwendung zu registrieren. Sie ist vom Typ "DATA" und hat den konventionellen Namen proj.nameda.DATA (z.B. INR105.TEST.DATA).

Eine SERVUS-Datei besteht

- aus einer Kopfzeile
- aus 0 bis mehreren Funktions-Aufzeichnungen
- und aus einer Endzeile.

Die Kopfzeile enthält 4 CHARACTER*8 Wörter, die für die Kennzeichnung und automatische Bearbeitung der Datei gebraucht werden.

Kopfzeile : { NAMEDA NORM NATAL ACTUAL } .

NAMEDA ist der Name der Datei (" Library " im SPF).
NORM bezeichnet die Schreib-Norm der Datei. NORM legt fest, in welcher Weise die Funktions-Daten in der Datei aufgeschrieben sind. Diese Normen werden im Teil 2.3 beschrieben.
NATAL gibt das Anlege-Datum der Datei an und
ACTUAL enthält den Tag, an dem die Datei zum letzten Mal vollständig neu beschrieben wurde.

Eine Funktions-Aufzeichnung besteht aus einer Namenszeile und aus einer oder zwei Datenzeilen. Die Namenszeile enthält Kennungs-Daten, die Anzahl der Elemente und Zeichen-Hilfen für die Funktion, die Datenzeile die Ordinaten der Funktion. Falls es zwei Datenzeilen gibt, sind die Ordinaten in der zweiten Zeile, in der ersten sind dann die Abszissen der Funktion.

Die Funktions-Aufzeichnungen werden bei der Verarbeitung durchnumeriert. Ein späteres Wiederauffinden einer Funktion erfolgt ausschließlich anhand ihrer Ordnungszahl.

Die Endzeile sieht formal wie die Namenszeile einer Funktions-Aufzeichnung aus, enthält aber an bestimmten Stellen das CHARACTER*8 Wort '**ENDE**'

2.2 *Einschränkungen.*

Die Verarbeitungs-Prozeduren erfordern gewisse Einschränkungen von den gespeicherten Funktionen:

- keine darf mehr als 1000 Elemente enthalten.
- Funktionen, die mit der DDFIG- oder DDFIGB-Routine als räumliche Oberfläche gezeichnet werden, dürfen höchstens **256** Elemente enthalten,
- jede solche Oberfläche muß mit höchstens **128** Funktionen auskommen.
- Die Anzahl der Markierungen auf einer Koordinaten-Achse (s. Abbildung 11 auf Seite 120) ist bei der Oberflächen-Erzeugung auf **21** begrenzt.
- Mit der FIGUR- bzw. FIGURB-Prozedur lassen sich höchstens **20** Funktionen in einem Bild zusammenlegen.

Eine weitere Einschränkung betrifft die Anordnung der Funktionselementen in der Dateien : um bei den Abbildung keine Fehler zu erzeugen, müssen alle Funktionen mit monoton-wachsenden Abszissen (x(i) < x(i+1) ...) gespeichert sein.

2.3 *Die Schreib-Normen der SERVUS-Dateien.*

Im SERVUS-System werden zur Zeit vier Schreib-Normen benutzt :

GRA4, GRA8, FOL8 und PLOT.

```
++++++++++++++++++++
+ GRA4 = 'GRAPHIC4' +
++++++++++++++++++++
```

Bei der Schreib-Norm GRA4 besteht jede Funktions-Aufzeichnung aus je drei Datengruppen:

1. { KLASSE , NUMMER , NP ,
   XMI , XMA , XED , FMI , FMA , FED,
   NAMX , MASX , NAMF , MASF } ,
2. { X(j) , j=1,NP } ,
3. { F(j) , j=1,NP } .

Die erste Datengruppe dient zur Kennung der Funktion. Sie enthält auch Hilfsgrößen für eine graphische Darstellung.

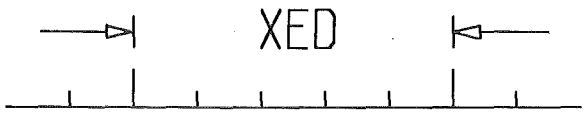
KLASSE,NUMMER sind zwei INTEGER Zahlen zur Identifikation der Funktion (z.B. Fall-Nr. und Nr. des Zeitschrittes).

NP (INTEGER) registriert die Anzahl der Funktions-Elemente.

Die folgenden 6 Größen sind alle REAL*4 Zahlen.

XMI,XMA sind untere bzw. obere Grenzen des Abszissen-Wertebereiches,

XED ist die Länge einer "Skala" auf der X-Achse (s. Skizze).



FMI,FMA, FED sind die entsprechenden Größen bei der F-Achse.

Die restlichen 4 Größen in dieser Gruppe sind alle CHARACTER*8 Konstanten. Sie werden eigentlich nur für die Achsenbeschriftung bei einer eventuellen graphischen Darstellung der Funktion benötigt, man kann sie aber auch als zusätzliche Kennungs-Hilfen benutzen.

NAMX ist die Name der X-Achse (z.B 'Temperat'),

MASX ist die Maß-Einheit (z.B. ' K ') der Achse.

NAMF, MASF sind die entsprechenden Namen für die F-Achse.

X(j) die zweite Datengruppe enthält die Abszissen und

F(j) die dritte Datengruppe enthält die Ordinaten der Funktion.

Bei der Schreibnorm 'GRAPHIC4' sind sowohl die Abszissen als auch die Ordinaten allesamt REAL*4 Zahlen. Die Endzeile bei dieser Norm ist die Datengruppe

{ KLASSE,NUMMER,NP,XMI,XMA,XED,FMI,FMA,FED,ENDE,ENDE,ENDE,ENDE }

wobei ENDE = '**ENDE**' ist.

Die Schreib-Norm GRA4 = 'GRAPHIC4' ist unbedingt erforderlich, falls man die Funktionen graphisch verarbeiten will ; die benötigten Routinen können keine REAL*8 Zahlen verarbeiten.

```
++++++++++++++++++++
+ GRA8 = 'GRAPHIC8' +
++++++++++++++++++++
```

Bei dieser Schreib-Norm sind ebenfalls die gleichen drei Datengruppen je Funktions-Aufzeichnung vorhanden :

1. { $KLASS , $NUMMR , $NP ,
   YMI , YMA , YED ; GMI , GMA , GED,
   NAMX , MASX , NAMF , MASF } ,
2. { Y(j) , j=1,NP } ,
3. { G(j) , j=1,NP } .

Die Daten [$KLASS , ... , G(NP)] haben dieselbe Bedeutungen, wie die Daten [KLASSE , ... , F(NP)] bei der Norm GRA4. Abweichend von der GRA4-Norm sind hier aber sowohl die Zahlen Y,G,YMI,...,GED, als auch die Nummern $KLASS, $NUMMR, $NP allesamt REAL*8 Zahlen. Die Endzeile bei der GRA8-Norm ist die Datengruppe

{ $KLASS,$NUMMR,$NP,YMI,YMA,YED,GMI,GMA,GED,ENDE,ENDE,ENDE,ENDE }

mit ENDE = '**ENDE**'.

Der Vorzug dieser Normierung ist, daß man die anfallenden Funktionen ohne Verlust an Genauigkeit speichern kann. Außerdem ist diese Speicherung mit der SPEAKEASY-System verträglich, so daß möglich ist, SPEAKEASY-Rechenergebnisse in SERVUS-GRA8-Dateien zu speichern und/oder Funktionen dieser Dateien in einer SPEAKEASY-Sitzung anzuschauen bzw. weiterzubehandeln. GRA8 eignet sich aber nicht zur Bild-Darstellung. Umkopieren ist notwendig!

```
++++++++++++++++++++
+ FOL8 = '*FOLIA*8' +
++++++++++++++++++++
```

Bei der Schreib-Norm FOL8 werden die zu notierenden Funktionen gruppenweise, als "Blätter" aufgezeichnet.

Jedes "Blatt" enthält mehrere Funktions-Aufzeichnungen, wobei aber immer nur die Ordinaten registriert werden. Die Abszissen werden nur einmal - am Anfang jedes "Blattes" - aufgeschrieben und gelten als gemeinsame X-Koordinaten für alle Funktionen auf diesem "Blatt".

Die Aufzeichnungen auf einem - auf dem $BL-ten - "Blatt" sehen wie folgt aus:

Als erstes erscheinen die Daten zum Abszissen-Bereich :

1,a) {$KLASS,$NUMMR, $NP, $BL,
YMI,YMA,YED,NAMX,MASX } ,
1,b) { Y(j) , j=1,NP } .

Dann folgen die Ordinaten-Daten von 1 - $NF Funktionen :

```
2,a)      {$KLASS,$NUMMR,-$BL, 1.D+0,
             GMI,GMA,GED,NAMF,MASF } ,
2,b)      { G(j) , j=1,NP } .
          .....
          .....
          .....
$NF+1,a)  {$KLASS,$NUMMR,-$BL, $NF,
             GMI,GMA,GED,NAMF,MASF } ,
$NF+1,b)  { G(j) , j=1,NP } .
```

Die Bezeichnungsweise der Größen entspricht im Wesentlichen der Bezeichnungsweise der GRA8-Norm. $NP ist die Zahl der Elemente der **ersten** Funktion. Die Zahl $BL ist die Ordnungszahl des Blattes. In jeder Funktions-Eintragung erscheint auch die Ordnungszahl der Funktion auf diesem Blatt (1.D+0 ,... , $NF).

Die Endzeile bei der FOL8-Norm lautet:

```
{ $KLASS,$NUMMR,$NP,$NF,GMI,GMA,GED,ENDE,ENDE } .
```

Diese Schreibnorm ist von Vorteil, wenn man beim Aufzeichnen Platz sparen will, setzt aber voraus, daß alle Funktionen eines "Blattes" auf demselben Abszissen-Bereich definiert sind. FOL8 ist auch nicht geeignet für die graphische Darstellung !

```
++++++++++++++++++++
+ PLOT = 'PLOTEASY' +
++++++++++++++++++++
```

Dies ist die bekannte PLOTEASY-Aufzeichnungsnorm /4/. Die Datengruppen sind hier:

1. { NP , NAMF , KLASSE , NUMMER } ,
2. { X(j) , j=1,NP } ,
3. { F(j) , j=1,NP } .

Die Abszissen und die Ordinaten sind hier REAL*4 Zahlen, NAMF ist eine CHARACTER*8 Konstante und die restlichen Größen sind INTEGERs. Die Bedeutung der Größen ist dieselbe wie bei der Norm GRA4. Die Endzeile bei der PLOT-Norm ist { NP,ENDE,KLASSE,NUMMER } . PLOT-Dateien eignen sich auch nicht zur direkten Bild-Herstellung !

3.0 Eine SERVUS-Datei wird angelegt.

Man kann - mit Hilfe der Kommandoprozedur IDA - eine SERVUS-Datei an legen, oder sie umtaufen/umnormieren. Falls die Datei schon besteht, wird nur die Kopfzeile der Datei neu geschrieben. Falls es sich um eine neue Datei handelt, wird sie erst allokiert und anschließend die Kopfzeile geschrieben.

Bei einer Umtaufe/Umnormierung geht der bisherige Inhalt der Datei verloren !

```
Aufruf :  EX  'tso000.SERVUS.CLIST(IDA)' .
```

Die Prozedur IDA fragt vom Benutzer die erforderlichen Angaben zu der Datei (Project, Name, Platte, Norm) ab und erledigt die Arbeit im Vordergrund. Zum Abschluss wird - als Bestätigung - die neue Kopfzeile der Datei am Bildschirm ausgegeben.

Die Liste der Prozedur IDA ist im Kap. 12 abgedruckt. Dem Benutzer wird empfohlen, diese Datei zu kopieren, wobei die kleingeschriebenen Namen in der Datei mit den eigenen Ausdrücken zu ersetzten sind. In diesem Falle vereinfacht sich der Aufruf zu :

```
EX SERVUS(IDA) .
```

Bemerkung: Beim Neuanlegen einer Datei geht IDA mit dem zu allokierenden Speicherraum recht großzügig um ; jede Datei erhält 10 Zylinder Primärraum. Falls man nicht so viel (oder noch mehr) Platz anlegen will, empfiehlt es sich, die Datei in Handarbeit zu allokieren (SPF-Menü 3.2) und IDA nur die Normierung überlassen.

4.0 Eine SERVUS-Datei wird mit einer Endzeile versehen.

Jede SERVUS-Datei erhält beim Anlegen eine Endzeile. Es kann aber vorkommen, daß diese zerstört wird, oder verloren geht, z.B. wenn ein Versuch gemacht wird, in eine - schon vollgeschriebene Datei - weitere Funktionen einzutragen. Eine Datei ohne Endzeile können die Lese- und Schreib-Routinen aber nicht mehr bearbeiten, somit würde der Verlust der Endzeile zum Verlust der ganzen Datei führen.

Die Kommandoprozedur ENDE ermöglicht es, in eine SERVUS-Datei hinter einer beliebigen Funktion eine Endzeile zu setzen. Man kann daher mit dieser Prozedur nicht nur beschädigte Dateien retten, sondern eine lange Liste unerwünschter Funktionen am Ende einer Datei schnell beseitigen.

```
Aufruf : EX  'tso000.SERVUS.CLIST(ENDE)' .
```

Die Prozedur arbeitet im Vordergrund. Erst wird vom Benutzer im Kommando-Dialog der Name der betreffenden Datei erfragt, dann setzt ein FORTRAN-Dialog ein zur Ermittlung der zukünftigen Stelle der Endzeile in der Datei. Der Benutzer wird dabei um die Ordnungszahl der letzten, zu erhaltenden Funktion gebeten. Bei der Eingabe dieser Zahl bestehen folgende Alternativen :

- **-** Falls man eine negative Zahl eingibt, dann erfolgt die Eintragung der Endzeile an Stelle der zuletzt gelesenen Funktion.
- **0** Falls man eine Null eingibt, endet die Prozedur.
- **+** Bei der Eingabe einer positiven Zahl wird - normalerweise - die Katalog-Eintragung (Namenszeile der Funktion) zu dieser Ordnungszahl vorgespielt. Falls es aber in der Datei keine Funktion mehr zu dieser Zahl gibt und auch keine Endzeile, dann endet die Prozedur fehlerhaft.

Die Liste der Prozedur ENDE ist ebenfalls in Kap. 12 zu finden.

5.0 Ein- und Ausgabe der Funktionen bei SERVUS-Dateien.

5.1 Öffnung der Dateien.

Bevor man Funktionen aus einer Datei lesen und/oder welche in eine Datei eintragen kann, muß man diese Dateien in Bereitschaft versetzen: sie müssen symbolische Nummern erhalten, bei einer Schreib-Datei muß der Schreibmodus festgelegt werden, usw. Zu diesem Zweck ist es unerläßlich, daß man vor dem ersten Schreib- oder Lesebefehl die Subroutine SERDIO aufruft.

Mit einem SERDIO-Aufruf kann man gleichzeitig je eine Schreib- und Lesedatei anregen.

```
Aufruf :          CALL SERDIO ( IRE,NORMRE,IWR,NORMWR,MODUS,KPRI ) .
```

Folgende Größen sind beim Aufruf anzugeben:

IRE **(INTEGER*4)** Symbolische Nummer der Ein/Ausgabe-Einheit zum lesen einer Datei. Für positive IRE erwartet SERDIO eine SERVUS-Lesedatei. Bei negativen IRE wird angenommen, daß die zu lesende Datei eine PLOTEASY-Datei ist - d.h. eine Datei, die in PLOT-Norm beschrieben ist und keine Kopfzeile (s. "Aufbau.") hat. Falls man keine Lesedatei benötigt, so setze man IRE = 0.

IWR **(INTEGER*4)** Symbolische Nummer zum Schreiben in die Datei. Falls man keine Schreibdatei braucht, kann man IWR = 0 setzen.

MODUS (CHARACTER*8) Schreib-Modus. Er kann folgende Werte haben:

MODUS **='**OLD***'** Das Schreiben in die Datei beginnt hinter dem letzten Eintrag, d.h. die alten Daten bleiben erhalten !

MODUS **='*NEWPAG*'** Das Schreiben in die Datei beginnt hinter dem letzten Eintrag.

Bei einer FOL8-Datei wird außerdem - im Gegensatz zum OLD-Modus - ein neues Blatt aufgeschlagen.

MODUS **='**NEW***'** Das Schreiben in die Datei beginnt am Anfang, hinter der Kopfzeile, d.h. die alten Daten werden zerstört!

KPRI (INTEGER*4) ist ein Maß für die Intensität der Ausgabe: für

KPRI = 0 wird nichts ausgedruckt, für

KPRI = 1 wird beim Schreiben die Namenszeile der notierten Funktion ausgedruckt, für

KPRI = 2 werden auch die Namenszeilen der gelesenen Funktion ausgedruckt.

KPRI > 2 wird nicht empfohlen. Es werden die Namenszeilen aller Funktionen ausgegeben, die bei der Lese- oder Schreib-Prozedur berührt werden.

Die restlichen Größen werden vom Programm ausgefüllt :

NORMRE (CHARACTER*8) Schreib-Norm der Lese-Datei.
NORMWR (CHARACTER*8) Schreib-Norm der Schreib-Datei.

Anwendungs-Beispiele für diese Routine kann man im Absatz 6 finden.

5.2 *Schreiben in eine Datei.*

Im folgenden werden die zum Schreiben einer Funktion benötigten Routinen zusammengestellt. Die Wahl der Schreib-Routine hängt von der Schreib-Norm der Datei ab.

Falls die Datei nach der **GRA4**-Norm beschrieben werden soll, wird die Subroutine SDING4 benutzt. Aufruf :

```
Aufruf :          CALL SDING4 ( KLASSE,NUMMER,NP
                                ,X,XMI,XMA,XED,NAMX,MASX
                                ,F,FMI,FMA,FED,NAMF,MASF
                                                   ,MODUS )  .
```

Für die folgenden Größen muß man beim Aufruf sinnvolle Werte angeben:

KLASSE (INTEGER)	2 Kenngrößen zur Identifikation
NUMMER (INTEGER)	der Funktion.
NP (INTEGER)	Anzahl der Funktions-Elemente, NP muß ≤ 1000 sein !
X (REAL*4 Feld)	Die Abszissen bzw.
F (REAL*4 Feld)	die Ordinaten der Funktion.
MODUS (CHARACTER*8)	Schreib-Modus, s. Routine SERDIO.

Die folgenden Größen brauchen beim Aufruf nur formal vorhanden sein:

XMI,XMA (REAL*4)	Bereichsgrenzen der Abszissen
FMI,FMA (REAL*4)	bzw. der Ordinaten.
XED (REAL*4)	Länge einer "Skala" (s. S. 6)
FED (REAL*4)	auf der X- bzw. F-Achse.
NAMX (CHARACTER*8)	Name für die X- bzw. für
NAMF (CHARACTER*8)	die F-Achse (z.B. ' ZEIT ').
MASX (CHARACTER*8)	Einheit für die X- bzw. für
MASF (CHARACTER*8)	die F-Achse (z.B. ' SEC ').

Bei einer Abbildung der Funktion werden die Koordinatenachsen mit NAMX MASX bzw. NAMF MASF beschriftet.

Falls die Datei nach der **GRA8**-Norm beschrieben werden soll, wird die Subroutine SDING8 verwendet.

```
Aufruf :          CALL SDING8 ( KLASSE,NUMMER,NP
                                ,Y,YMI,YMA,YED,NAMX,MASX
                                ,G,GMI,GMA,GED,NAMF,MASF
                                                   ,MODUS )  .
```

Größen, die man beim Aufruf sinnvoll setzen muß :

KLASSE (INTEGER)	2 Kenngrößen zur Identifikation
NUMMER (INTEGER)	der Funktion.
NP (INTEGER)	Anzahl der Funktions-Elemente. NP muß ≤ 1000 sein !
Y (REAL*8 Feld)	Die Abszissen bzw.
G (REAL*8 Feld)	die Ordinaten der Funktion.
MODUS (CHARACTER*8)	Schreib-Modus, s. Routine SERDIO.

Größen, die nur formal vorhanden sein müssen:

YMI,YMA (REAL*8)	Bereichsgrenzen der Abszissen
GMI,GMA (REAL*8)	bzw. der Ordinaten.
YED (REAL*8)	Länge einer "Skala"
GED (REAL*8)	auf der X- bzw. F-Achse.
NAMX (CHARACTER*8)	Name für die X- bzw. für
NAMF (CHARACTER*8)	die F-Achse (z.B. ' ZEIT ').
MASX (CHARACTER*8)	Einheit für die X- bzw. für
MASF (CHARACTER*8)	die F-Achse (z.B. ' SEC ').

Falls die Datei nach der **FOL8**-Norm beschrieben werden soll, ruft man die Subroutine SDINF8 auf.

```
Aufruf :        CALL SDINF8 ( KLASSE,NUMMER,NP
                            ,Y,YMI,YMA,YED,NAMX,MASX
                            ,G,GMI,GMA,GED,NAMF,MASF
                                                ,MODUS,NEWP )  .
```

Größen, die beim Aufruf sinnvolle Werte verlangen:

KLASSE (INTEGER)	2 Kenngrößen zur Identifikation
NUMMER (INTEGER)	der Funktion.
NP (INTEGER)	Anzahl der Funktions-Elemente. NP muß $\leq$ 1000 sein !
NEWP (integer)	für NEWP > 0 wird die Funktion auf ein neues "Blatt" geschrieben, sonst kommt sie auf das angefangene "Blatt".
Y (REAL*8 Feld)	Die Abszissen bzw.
G (REAL*8 Feld)	die Ordinaten der Funktion.
MODUS (CHARACTER*8)	Schreib-Modus, s. Routine SERDIO.

Größen, die nur formal vorhanden sein müssen:

YMI,YMA (REAL*8)	Bereichsgrenzen der Abszissen
GMI,GMA (REAL*8)	bzw. der Ordinaten.
YED (REAL*8)	Länge einer "Skala"
GED (REAL*8)	auf der X- bzw. F-Achse.
NAMX (CHARACTER*8)	Name für die X- bzw. für
NAMF (CHARACTER*8)	die F-Achse (z.B. ' ZEIT ').
MASX (CHARACTER*8)	Einheit für die X- bzw. für
MASF (CHARACTER*8)	die F-Achse (z.B. ' SEC ').

Bemerkung zur NP : die Zahl der Elemente aller Funktionen eines Blattes, NP wird durch die erste eingetragene Funktion entschieden. Nachfolgend geschriebene Funktionen werden - bei einer abweichenden NPi - entweder verkürzt, falls NPi > NP ist, oder - im Falle NPi < NP - die fehlende Werte mit dem letzten Funktionswert F(NPi) überschrieben.

Bei einer Datei mit der **PLOT**-Norm wird die Subroutine SDINPY benutzt.

```
Aufruf :        CALL SDINPY ( KLASSE,NUMMER,NP,X,F,NAMF,MODUS )  .
```

Hier werden folgenden Größen beim Aufruf benötigt:

KLASSE (INTEGER)	2 Kenngrößen zur Identifikation
NUMMER (INTEGER)	der Funktion.

NP	**(INTEGER)**	Anzahl der Funktions-Elemente, NP muß ≤ 1000 sein !
X	**(REAL*4 Feld)**	Die Abszissen bzw.
F	**(REAL*4 Feld)**	die Ordinaten der Funktion.
NAMF	**(CHARACTER*8)**	Name für die F-Achse (z.B. ' DRUCK ').
MODUS	**(CHARACTER*8)**	Schreib-Modus, s. Routine SERDIO.

5.3 Schreiben im Dialogverfahren.

Es besteht auch die Möglichkeit, Funktionen in einem Dialog herstellen, korrigieren, und in eine SERVUS-Datei schreiben zu lassen. Dazu wird die Kommandoprozedur FUNKIN (s. "Die SERVUS-Kommandoprozeduren.") benötigt. FUNKIN erfragt in einem FORTRAN-Dialog vom Benutzer die Funktionswerte sowie die Kenngrößen und trägt diese handgemachte Funktion in die vorliegende Schreibdatei ein. FUNKIN erlaubt es, neben der Schreib-Datei auch eine Lese-Datei (als Funktions-Vorlage) zu benutzen.

Die Erfragung des Benutzers geschiet mit Hilfe von Entscheidubgs-Listen (Menüs). Die wichtigste Entscheidungen sind im Hauptmenü zusammengestellt :

```
PROGRAMMSTATUS / 0 / =:

  "< 0 "   STOP :
  "  1 "   FUNKTION AUSDRUCKEN :
  "  2 "   EINGABE DER ROHFUNKTION :
  "  3 "   KORREKTUR DER FUNKTIONSWERTEN :
  "  4 "   KORREKTUR DER REIHENFOLGE :
  "  5 "   PUNKTE EINFUEGEN :
  "  6 "   PUNKTE LOESCHEN :
  "  7 "   KORREKTUR DER KENNGROESSEN :
  "  8 "   FUNKTION UMFORMEN :
  "  9 "   FUNKTION REGISTRIEREN :
```

Je nach dem, welche Zahl jetzt der Benutzer eintippt, erfolgt eine bestimmte Aktion. Z.B. das Eintippen von 9 führt dazu, daß die grade aktive Funktion in die vorliegende Schreibdatei eingetragen wird, bei einer 1 werden die Funktionswerte der Reihe nach am Bildschirm gezeigt, usw. Eine negative Zahl im Hauptmenü beendet die ganze Prozedur.

Einige der Wahlmöglichkeiten des Hauptmenüs sind selber (Unter-) Menüs , so

- die Korrektur der Kenngröße :

```
KORREKTUR DER KENNGROESSEN / 0 / =:

  "  0 "   ALLES IN ORDNUNG :
  "  1 "   KENNGROESSEN AUSDRUCKEN :
  "  2 "   KLASSE , NUMMER :
  "  3 "   NAMX , MASX  :
  "  4 "   XMI , XMA , XED :
  "  5 "   NAMF , MASF  :
  "  6 "   FMI , FMA , FED :
```

- und die Umformung der Rohfunktion :

```
WEITERE BEHANDLUNGEN  =:

  "  1 "   F(X)   ===>  MONOTON( F(X) )  :
  "  2 "     X  <===>  F                  :
  "  3 "   F      ===>  F + KONSTANTE     :
  "  4 "   F      ===>  F * KONSTANTE     :
  "  5 "    X  =  PERMUTATION ( F )  ?   :
```

Aus jedem Untermenü kehrt man mit einer 0 in das Hauptmenü zurück. Aus dem Hauptmenü kann man auch direkt in eine Entscheidung einer Untermenü gelangen durch das eintippten einer zweistelligen Zahl. Die Zahl 7.2 im Hauptmenü fürt zu der Entscheidung " KLASSE , NUMMER " im siebten Untermenü. Aus einer Untermenü gelangt man widerum mit einer negativen Zahl in eine andere Untermenü : -2. aus dem Untermenü 3 fürt in das 2. Untermenü.

Die Prozedur ruft man mit

EX 'tso000.SERVUS.CLIST(FUNKIN)'

auf.

5.4 *Lesen aus einer Datei.*

Auch beim Lesen einer Datei muß die Leseroutine dem jeweiligen Datei-Norm entsprechen.

Bei einer **GRA4**-Datei liest man mit der Subroutine SDEXG4.

```
Aufruf :          CALL SDEXG4 ( KLASSE,NUMMER,NP
                                ,X,XMI,XMA,XED,NAMX,MASX
                                ,F,FMI,FMA,FED,NAMF,MASF
                                                    ,LOS )  .
```

Bei diesem Aufruf muß man allein LOS angeben :

LOS **(INTEGER)** Ordnungszahl der zu lesenden Funktion in der Datei.

Alle anderen Größen : KLASSE,NUMMER,NP ,X,XMI,XMA,XED,NAMX,MASX, F,FMI,FMA,FED,NAMF und MASF werden von der Leseroutine gefüllt. Ihre Beschreibung findet man in 5.2, bei der Schreibroutine SDING4. Falls es in der Datei keine LOS-te Funktion gibt, dann erfolgt eine Fehlernachricht (KPRI im SERDIO-Aufruf muß > 0 sein !) und die Character-Größe NAMF wird zu '**ENDE**' gesetzt.

Bei einer **GRA8**-Datei liest man mit der Subroutine SDEXG8.

```
Aufruf :          CALL SDEXG8 ( KLASSE,NUMMER,NP
                                ,Y,YMI,YMA,YED,NAMX,MASX
                                ,G,GMI,GMA,GED,NAMF,MASF
                                                    ,LOS )  .
```

Auch bei diesem Aufruf muß man nur LOS angeben :

LOS **(INTEGER)** Ordnungszahl der zu lesenden Funktion in der Datei.

Alle anderen Größen : KLASSE,NUMMER,NP ,Y,YMI,YMA,YED,NAMX,MASX, G,GMI,GMA,GED,NAMF und MASF werden von der Leseroutine gefüllt. Für ihre Beschreibung s. die Schreibroutine SDING8 in 5.2. Falls die LOS-te Funktion nicht zu finden ist, erfolgt eine Fehler nachricht und NAMF wird zu '**ENDE**' gesetzt.

Bei einer **FOL8**-Datei liest man mit der Subroutine SDEXF8.

```
Aufruf :        CALL SDEXF8 ( KLASSE,NUMMER,NP
                            ,Y,YMI,YMA,YED,NAMX,MASX
                            ,G,GMI,GMA,GED,NAMF,MASF
                                                ,LOS,NEWP ) .
```

Beim Aufruf muß man nur LOS angeben :

LOS **(INTEGER)** Ordnungszahl der zu lesenden Funktion in der Datei.

Alle anderen Größen : KLASSE,NUMMER,NP ,Y,YMI,YMA,YED,NAMX,MASX, G,GMI,GMA,GED,NAMF,MASF und NEWP werden von der Leseroutine gefüllt. Die Beschreibung dieser Größen steht in 5.2 bei der Routine SDINF8.

NEWP enthält immer die Ordnungszahl des jeweiligen Blattes IBL. Bei einer Funktion, die am Anfang des Blattes steht, erscheint diese Zahl, bei anderen Funktionen wird -IBL zurückgegeben.

Bei einer erfolglosen Suche wird hier auch - nebst einer Fehlernachricht - NAMF zu '**ENDE**' gesetzt.

Bei einer **PLOT**-Datei liest man mit der Subroutine SDEXPY.

```
Aufruf :        CALL SDEXPY ( KLASSE,NUMMER,NP,X,F,NAMF,LOS ) .
```

Anzugeben braucht man beim Aufruf nur LOS :

LOS **(INTEGER)** Ordnungszahl der zu lesenden Funktion in der Datei.

Die Größen KLASSE,NUMMER,NP,X,F und NAMF (Beschreibung bei der Routine SDINPY) werden von SDEXPY geliefert. Bei nichtvorhandenen Funktionen erscheint nach dem Aufruf NAMF als '**ENDE**'.

Bei einer "Kopflosen"-PLOT-Datei (s. "Öffnung der Dateien.") wird mit der Routine PDEXPY gelesen. Der Aufruf lautet :

```
                CALL PDEXPY ( KLASSE,NUMMER,NP,X,F,NAMF,LOS ) .
```

5.5 *Benötigte Sonder-Steuerkarten.*

Damit der Benutzer bei einer Stapel-Job Funktionen in eine SERVUS-Datei schreiben und/oder Funktionen aus einer Datei lesen kann, muß er beim Hintergrund-Job zusätzliche JCL-Karten ausfüllen. Um die Dateien zu allokieren, braucht er folgende DD-Karten:

```
//G.FTxxF001 DD DSN=inr000.ergebnis.DATA,DISP=(OLD,SHR)
//G.FTyyF001 DD DSN=inr000.inform.DATA,DISP=(OLD,SHR)
```

Hier ist z.B. xx = IWR die symbolische Nummer der Schreib-Datei und yy=IRE die der Lese-Datei. Diese beiden Dateien müssen immer mit verschiedenen symbolischen Nummern allokiert sein.

Außerdem muß er den Modul INR105.SERVUS.LOAD(SERDIO) dem Job zu fügen. Dies kann man z.B. mit den folgenden L-Karten erreichen :

```
//L. ...
//L.bibl DD DISP=SHR,DSN=INR105.SERVUS.LOAD
 INCLUDE bibl(SERDIO)
 ENTRY MAIN
//*
//G. ...
```

5.6 *Beispiele.*

Das erste Beispiel zeigt einen Stapel-Job, der eine Anzahl von Funktionen errechnet und sie anschließend in die Datei reform.DATA einträgt. Eine Lesedatei wird nicht benötigt.

```
//inr000B1 JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(0,0),
// MSGCLASS=H
//SPEICHRN EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C
      REAL*8 Y(1000),G(1000),YMI,YMA,YED,GMI,GMA,GED
      CHARACTER*8 NAMX,MASX,NAMF,MASF,NORMR,NORMW,MODUS
C
C     ************** E I N G A B E ****************
      DATA IRE/0/,IWR/16/,MODUS/'**NEW***'/,KPRI/2/
     =,KLASSE/1/,NP/50/
     =,NAMX/'   X    '/,MASX/' - 1 -  '/
     =,NAMF/'SIN(I*X)'/,MASF/' - 1 -  '/
C     ***********************************************
C
      CALL SERDIO(IRE,NORMR,IWR,NORMW,MODUS,KPRI)
      IF(NORMW .NE. 'GRAPHIC8')                    GOTO 100
C
      Y(1)=0.D-0
      YMI = Y(1)
      DO 11 J=2,NP
   11 Y(J)=Y(J-1)+0.5D-1
      YMA=Y(NP)+0.5D-1
      GMA=1.D-0
      GMI=-GMA
      YED=0.2*(YMA-YMI)
      GED=0.2*(GMA-GMI)
      DO 51 NUMMER=1,5
      DO 21 J=2,NP
   21 G(J)=SIN(NUMMER*Y(J))
      IF(NUMMER .EQ. 1)      NAMF(5:5)='1'
      IF(NUMMER .EQ. 2)      NAMF(5:5)='2'
      IF(NUMMER .EQ. 3)      NAMF(5:5)='3'
      IF(NUMMER .EQ. 4)      NAMF(5:5)='4'
      IF(NUMMER .EQ. 5)      NAMF(5:5)='5'
      CALL SDING8(KLASSE,NUMMER,NP
     =,Y,YMI,YMA,YED,NAMX,MASX,G,GMI,GMA,GED,NAMF,MASF,MODUS)
   51 CONTINUE
C
  100 STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO)
 ENTRY MAIN
//*
//G.FT16F001 DD DSN=inr000.szuszog.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
//
```

Abbildung 1 auf Seite 110 zeigt die Nachrichten, die dieser Job erzeugt hat.

Das zweite Beispiel zeigt einen Stapel-Job, der eine Anzahl von Funktionen von einer Datei liest, diese mit sich selber multipliziert und die so gewonnenen, neuen Funktionen am Ende derselben Datei einspeichert.

```
//inr000B2 JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(0,0),
// MSGCLASS=H
//EINAUS EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C
      REAL*8 Y(500),G(500),F(500),YMI,YMA,YED,GMI,GMA,GED
      CHARACTER*8 NAMX,MASX,NAMF,MASF,NORMR,NORMW,MODUS,NAMN
C
C     ********** E I N G A B E **********
      DATA IRE/15/,IWR/16/,MODUS/'**OLD***'/,KPRI/2/
     =,KLASSN/2/,NAMN/'SI2(I*X)'/,LOS/1/
C     ***********************************************
C
      CALL SERDIO(IRE,NORMR,IWR,NORMW,MODUS,KPRI)
      IF(NORMW .NE. 'GRAPHIC8')                    GOTO 100
C
      DO 51 NUMMNE=1,5
      CALL SDEXG8(KLASSE,NUMMER,NP
     =,Y,YMI,YMA,YED,NAMX,MASX,G,GMI,GMA,GED,NAMF,MASF,LOS)
      IF(NAMF .EQ. '**ENDE**')                     GOTO 100
      DO 21 J=2,NP
   21 F(J)=G(J)*G(J)
      IF(NUMMNE .EQ. 1)      NAMN(5:5)='1'
      IF(NUMMNE .EQ. 2)      NAMN(5:5)='2'
      IF(NUMMNE .EQ. 3)      NAMN(5:5)='3'
      IF(NUMMNE .EQ. 4)      NAMN(5:5)='4'
      IF(NUMMNE .EQ. 5)      NAMN(5:5)='5'
      CALL SDING8(KLASSN,NUMMNE,NP
     =,Y,YMI,YMA,YED,NAMX,MASX,F,GMI,GMA,GED,NAMN,MASF,MODUS)
   51 LOS=LOS+1
C
  100 STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.szuszog.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD DSN=inr000.szuszog.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
//
```

Die Nachrichten dieses Jobs sind in Abbildung 2 auf Seite 111 zu sehen.

Das dritte Beispiel zeigt einen Stapel-Job, der sämtliche Funktionen einer PLOTEASY-Datei in eine GRA8-SERVUS-Datei umkopiert.

```
//inr000B3 JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(0,0),
// MSGCLASS=H
// EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C
      REAL*4 X(500),F(500)
      REAL*8 Y(500),G(500),YMI,YMA,YED,GMI,GMA,GED
      CHARACTER*8 NAMX,MASX,NAMF,MASF,NORMR,NORMW,MODUS
C
C     ********** E I N G A B E **********
      DATA IRE/-15/,IWR/16/,MODUS/'**OLD***'/,KPRI/2/
     =,LOS/1/,NAMX,MASX,MASF/3*'-FEHLT-'/
C     ***********************************************
C
      CALL SERDIO(IRE,NORMR,IWR,NORMW,MODUS,KPRI)
      IF(NORMW .NE. 'GRAPHIC8')                    GOTO 100
C
      DO 51 NUMMNE=1,99
      CALL PDEXPY(KLASSE,NUMMER,NP,X,F,NAMF,LOS)
      IF(NAMF .EQ. '**ENDE**')                     GOTO 100
      DO 21 J=1,NP
      Y(J)=X(J)
   21 G(J)=F(J)
      CALL SDING8(KLASSE,NUMMER,NP
     =,Y,YMI,YMA,YED,NAMX,MASX,G,GMI,GMA,GED,NAMF,MASF,MODUS)
   51 LOS=LOS+1
C
  100 STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.plotes.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD DSN=inr000.szuszog.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
//
```

Für die Jobmeldungen s. Abbildung 3 auf Seite 112.

6.0 Eine SERVUS-Datei wird gesichtet und kopiert.

6.1 Das Verfahren.

Man kann von den Funktionen, die in einer SERVUS-Datei auf gezeichnet sind , einen Katalog bekommen. Dieser Katalog listet die Namenszeilen der in der Datei registrierten Funktionen der Reihe nach auf. Die Daten der Namenszeile erscheinen in folgender Ordnung:

1. Druckzeile : Ordnungzahl der Funktion auf der Datei, KLASSE , NUMMER , NAMX , MASX , NAMF, MASF ,
2. Druckzeile : X(1) , NP , X(NP) , XMI , XMA , XED ,
3. Druckzeile : F(1) , NP , F(NP) , FMI , FMA , FED
(s. z.B. Abbildung 4 auf Seite 113) .

Man kann auch die SERVUS-Dateien - im Gegensatz zu normalen "DATA" -Dateien - kopieren, allerdings nur in andere SERVUS-Dateien. Beim Kopieren besteht auch die Möglichkeit, die Daten im gewissen Umfang zu verändern. Diese Änderungen erfolgen dann aber für alle Funktionen, die kopiert wurden. Man kann

- die Funktionen in ein vorgegebenes Datenfenster einfügen,
- die Zahlenwerte der Funktionen einschränken,
- die Funktionen mittels einer "Präge"-Funktion an einem vorgegebenen Abszissen-Gitter abzubilden.

Beim Zurecht-Schneiden der Funktionen durch ein vorgegebenes Daten fenster werden Elemente, deren Abszissen in das "Abszissen-Fenster"

$$[\ XMIST \leq X \leq XMAST\ ]$$

nicht hineinpassen, beim Kopieren weggelassen. Außerdem werden Ordinaten-Werte, die aus einem "Ordinaten-Fenster"

$$[\ FMIST \leq F \leq FMAST\ ]$$

hinausragen, durch die Werte FMIST oder FMAST ersetzt. Die Eingangswerte der Größen XMI, XMA, FMI, FMA ersetzt man dabei mit XMIST, XMAST, FMIST, FMAST. Falls kein sinnvolles Fenster vorgelegt wurde, ermittelt das Programm - anstatt zurechtzuschneiden - ein minimales Datenfenster.

Die Einschränkung der Zahlenwerte erfolgt, indem man die Abszissen und/oder Ordinaten so lange mit dem Faktor 0.001 oder 1000. multipliziert, bis die Beträge aller diesen Zahlen im Bereich

$$(\ 0.001\ ,\ 1000.\ )$$

liegen. Die Faktoren bleiben dabei im Bereich

$$[\ 1.E\text{-}75\ ,\ 1.E+75\ ].$$

Diese Umnormierung wird dann auch in dem Namen MASX und/oder MASF vermerkt : das Eingangs-Wort der jeweiligen Größe wird durch das Wort ' *1.E+yz' (= 1/Umnormierungsfaktor) ersetzt.

Bei der Abbildung der Funktionen an einem festgelegten Abszissen-Gitter werden die neue Ordinaten in der Regel mit einem SPLINE-5-Verfahren aus den alten Werten errechnet. Man kann aber auch lineare Interpolation benutzen. Die angepaßte Funktionen übernehmen auch die Achsen-Characteristika der Präge-Funktion:

XMI, XMA, XED, NAMX, MASX bzw. FMI, FMA, FED, NAMF, MASF .

6.2 *Sichten bzw. Kopieren im Stapel-Betrieb.*

Zum Sichten/Kopieren im Stapel-Betrieb kann man die Hintergrund-Prozedur KOPIERT benutzen. Sie sieht wie folgt aus:

Name : tso000.SERVUS.CNTL(KOPIERT)

```
//inr000k  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
// EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C               PROGRAMM KOPIERT
      CALL COPYDA
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT14F001 DD DSN=inr000.bilder.DATA,DISP=(OLD,KEEP)
//G.FT12F001 DD DSN=inr000.vielbild.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LAKOPI IRE=14,IWR=12,MODUS='**OLD***',INLIST=0,LAUF=1,31,
 LAX=0,LAF=0,KPRI=1,MUSTR=10,IQ=1, &END
 &LAKOPI IRE=12,IWR=0,INLIST=0,LAUF=1,99,KPRI=2, &END
 &LAKOPI IRE=0, &END
//
```

Eingabebeschreibung : Die Daten werden im NAMELIST-Format eingegeben. Die Liste der Eingabedaten lautet :

```
NAMELIST /LAKOPI/ IRE,IWR,MODUS,INLIST,LAUF,KPRI,LAX,LAF,
        XMIST,XMAST,XEDST,FMIST,FMAST,FEDST,MUSTR,IQ
```

Von den Listen-Größen sind XMIST, XMAST, XEDST, FMIST, FMAST, FEDST REAL*4 Zahlen, MODUS ist ein CHARACTER*8 Name, alle anderen Zahlen sind INTEGERs. LAUF ist ein INTEGER-Feld mit 50 Elementen.

Die Bedeutungen der einzelnen Eingabe-Größen:

IRE , IWR sind die symbolischen Nummern der Lese- bzw Schreib-Datei. Falls IWR = 0 oder IWR = IRE ist, wird nicht geschrieben, es wird nur die Datei IRE gelesen. Bei negativer IRE wird eine PLOTEASY-Datei (s.a. "Öffnung der Dateien.") erwartet. Eine Eingabe-Liste mit IRE = 0 stoppt die Prozedur.

INLIST , LAUF bestimmen zusammen eine Menge von Ordnungszahlen. Gelesen werden dann die Funktionen, die in der Lese-Datei zu diesen Ordnungszahlen gehören.

Falls

INLIST > 0 ist, besteht die Zahlenmenge aus den Ordnungszahlen

{LAUF(1),LAUF(2),...,LAUF(INLIST)} ,

falls

INLIST = 0 ist, so sind es die Ordnungszahlen

{LAUF(1) ≤ i ≤ LAUF(2)} bzw. {LAUF(1) ≥ i ≥ LAUF(2)}

und falls

INLIST < 0 ist, hat die Zahlenmenge folgende -INLIST Elemente

{LAUF(1), [LAUF(1)+LAUF(2)], [LAUF(1)+2*LAUF(2)], ...}.

KPRI ist ein Maß für die Intensität der Ausgabe : für

KPRI = 0 wird nichts ausgedruckt, für

KPRI = 1 werden die Namenszeilen der notierten Funktionen ausgedruckt, für

KPRI = 2 werden auch die Namenszeilen der gelesenen Funktion ausgedruckt.

KPRI > 2 wird nicht empfohlen. Es werden zu viele Daten ausgedruckt und es kann zur Zeilenüberschreitung kommen.

Die folgenden Eigabedaten werden nur dann gebraucht, falls in eine Datei geschrieben werden soll (IWR > 0).

MODUS ist der Schreib-Modus. Er kann folgende Werte haben:

MODUS='OLD***'** das Schreiben in die Datei beginnt hinter dem letzten Eintrag, d.h. die alten Aufzeichnungen bleiben erhalten !

MODUS='NEW***'** das Schreiben in die Datei beginnt am Anfang, hinter der Kopfzeile, d.h. die alten Aufzeichnungen werden zerstört!

MODUS='*NEWPAG*' das Schreiben in die Datei beginnt hinter dem letzten Eintrag. Bei einer FOL8-Datei wird außerdem - im Gegensatz zum OLD-Modus - ein neues Blatt aufgeschlagen.

LAX steuert die Korrektur der Abszissen beim Kopieren. Bei

LAX = 0 werden die Abszissen unverändert kopiert, falls

LAX = 1 | -1 ist, dann werden die Funktion durch das vorgegebene Abszissen-Fenster eingeschränkt und falls

LAX > 0 ist, dann werden auch die Zahlenwerte der Abszissen in den (0.001,1000.)-Bereich überführt.

LAF steuert die Korrektur der Ordinaten beim Kopieren. Bei

LAF = 0 werden die Ordinaten unverändert kopiert, falls

LAF = 1 | -1 ist, dann werden die Ordinaten mit dem vorgegebenen Ordinaten-Fenster zurechtgeschnitten und falls

LAF > 0 ist, dann werden auch die Zahlenwerte der Ordinaten auf den (0.001,1000.)-Bereich eingeschränkt.

XMIST , XMAST Abszissen-Fenster. Falls man XMIST ≥ XMAST eingibt, bestimmt die Prozedur das optimale Fenster anhand der Abszissenmenge selbständig.

FMIST , FMAST Ordinaten-Fenster. FMIST ≥ FMAST erzeugt ein optimales Ordinaten-Fenster.

XEDST , FEDST vorgegebenen Skalen-Längen auf der X- bzw. F- Achse (s. S. 6).

MUSTER und IQ erlauben es, die Funktionen beim Kopieren einheitlich in einem vorgewählten Abszissen-Gitter abzubilden.

MUSTR ist die Ordnungszahl der "Präge"-Fuktion in der Schreib-Datei. Bei

MUSTR = 0 wird keine "Präge"-Fuktion beim Schreiben benutzt.

IQ steuert die Berechnungsart der neuen Ordinaten : bei

IQ < 0 werden diese linear interpoliert und bei

IQ ≥ 0 mit dem SPLINE-5-Verfahren berechnet.

Eingabe-Beispiele :

```
&LAKOPI IRE=12,IWR=0,INLIST=0,LAUF=1,99,KPRI=2, &END
```

Teilkatalog der SERVUS-Datei "12", von der 1. bis zum 99. Funktion. Als Ausgabe-Beispiel s. Abbildung 4 .

```
&LAKOPI IRE=13,IWR=16,MODUS='**NEW***',KPRI=0,LAUF=13,20,
INLIST=-90,LAX=1,LAF=1,XMIST=1.0,XMAST=50.,
FMIST=-2.E+2,FMAST=0.0E+2, &END
```

90 Funktionen der Datei "13" werden in die Datei "16" kopiert und zwar die 13., die 33., usw. Die Datei "16" wird neu beschrieben ! Es erfolgt keine Druckausgabe. Sowohl die Abszissen, als auch die Ordinaten werden eingepasst und eingeschränkt.

```
&LAKOPI IRE=14,IWR=12,MODUS='**OLD***',INLIST=0,LAUF=1,31,KPRI=0,
LAX=0,LAF=0,MUSTR=10,IQ=1, &END
```

Die ersten 31 Funktionen der Datei "14" werden in die Datei "12" kopiert. Alle kopierten Funktionen haben dieselben Abszissen , als die 10. Funktion der Datei "12" und die neue Ordinaten werden durch ein SPLINE-5-Verfahren berechnet.

```
&LAKOPI IRE=0, &END
```

Immer erforderliche End-Eingabe. Sie ist stets die letzte NAMELIST.

Die Zusammenstellung der Steuerkarten eines Kopier-Prozedur kann mann vereinfachen, indem man die Kommandoprozedur "KOPIER" (s. "Die SERVUS-Kommandoprozeduren.") benutzt. KOPIER erfragt vom Benutzer die notwendigen Eigaben und baut daraus eine CNTL-Datei zusammen; diese kann man dann entweder sofort ausfüren lassen, oder noch weiter bearbeiten. Die Prozedur ruft man mit

```
EX 'tso000.SERVUS.CLIST(KOPIER)'
```

auf.

6.3 *Sichten/Kopieren im Dialog.*

Es gibt auch ein Dialog-Verfahren zur Durchsicht und zum eventuellen Kopieren einer Datei am Bildschirm : DATAS . DATAS stellt bis zu drei Dateien zum Lesen bereit und eine vierte zum Schreiben. Die Angaben über diese Dateien werden in einer Kommando-Dialog erfragt. Anschließend setzt ein FORTRAN-Gespräch ein, in dem die Eingabedaten zum Kopieren bzw. zum Listen erfragt werden. Diese Eingabedaten entsprechen den Eingabedaten der Stapelprozedur.

Der Aufruf lautet :

```
EX 'tso000.SERVUS.CLIST(DATAS)' .
```

Die Prozedurliste ist im Kap. 12 abgedruckt.

7.0 Änderungen der Funktionen einer SERVUS-Datei.

7.1 Einige Funktionen der Datei werden renormiert oder gestrichen.

Man kann Funktionen, die in einer SERVUS-Datei nicht mehr erwünscht sind, mit Hilfe der Hintergrund-Prozedur LOESCHT eliminieren. Die Prozedur kopiert die betroffene Datei in eine Hilfsdatei, wobei die unerwünschten Funktionen unberücksichtigt bleiben. Die Hilfsdatei wird dann wieder in die ursprüngliche Datei zurückkopiert. Von der bereinigten Datei wird anschließend ein neuer Katalog erstellt. Anstatt die fragliche Funktionen zu löschen, kann man sie im Rahmen dieser Prozedur auch renormieren, d.h. die Ordinaten aller betroffenen Funktionen mit einem Faktor multiplizieren.

Die Hintergrund-Prozedur (mit anschließendem Katalogschritt) sieht aus wie folgt:

Name : tso000.SERVUS.CNTL(LOESCHT)

```
//inr000e  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//ERASE  EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C               PROGRAMM LOESCHT
      CALL ERASE
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.vielbild.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
//G.SYSIN DD *
 &LALOES IRE=15,IWR=16,NUMMAX=300,NIX=219,220,KPRI=0, &END
 &LALOES IRE=0, &END
//*
```

```
//KATALOG EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C                    KATALOG
      CALL COPYDA
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.szuszog.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LAKOPI IRE=15,IWR=0,MODUS='**OLD***',INLIST=0,LAUF=1,1000,
 KPRI=2, &END
 &LAKOPI IRE=0, &END
//
```

Eingabebeschreibung:
Die Daten werden im NAMELIST-Format eingegeben. Die Liste ist die folgende:

```
NAMELIST /LALOES/ NUMMAX,NIX,IRE,IWR,KPRI,FAKTOR
```

Alle Listen-Größen außer FAKTOR sind INTEGERs, NIX ist ein Feld von 50 Elementen, FAKTOR ist eine REAL*4-Zahl.

Die Bedeutungen der einzelnen Eingabe-Größen:

NUMMAX	ist die Ordnungszahl der letzten Funktion, die bei der Prozedur noch erhalten bleiben soll.
NIX(1),NIX(2), ...,NIX(k),...	sind die Indizes der Funktionen, die gelöscht oder werden sollen.
IRE	ist die symbolische Nummer der zu korrigierenden Datei. Eine Liste mit IRE = 0 beendet die Prozedur.
IWR	ist die symbolische Nummer der Hilfsdatei.
KPRI	Druckintensität, es empfiehlt sich KPRI = 0.

FAKTOR ist ein Renormierungsfaktor. Falls

FAKTOR = 0.0	ist, dann werden die angegebene Funktionen gelöscht,
FAKTOR ≠ 0.0	werden die Ordinaten der angegebenen Funktionen mit FAKTOR multipliziert.

Auch bei diesem Stapel-Prozedur kann man die Steuerkarten halb-mechanisch, mit Hilfe einer Kommando-Prozedur erstellen lassen. Der Aufruf lautet

```
EX 'tso000.SERVUS.CLIST(LOESCH)'
```

Für die Beschreibung der Kommandoprozedur "LOESCH" s. "Die SERVUS-Kommandoprozeduren.".

7.2 *Die Reihenfolge der Funktionen in der Datei wird abgeändert.*

Man kann auch die Reihenfolge der Funktionen in einer Datei manipulieren: die Hintergrund-Prozedur MISCHT erlaubt es, die Ordnung von maximal 1000 Funktionen auf ein-

mal zu ändern. Hierbei wird - wie auch bei der Prozedur LOESCHT - von einer kurzlebigen Hilfsdatei gebrauch gemacht. Es folgt ein Muster dieser Prozedur :

Name : tso000.SERVUS.CNTL(MISCHT)

```
//inr000m  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//MIX    EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C               PROGRAMM MISCHT
      CALL MIX
      STOP
      END
//*
//L.SYSPRINT DD DUMMY
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.test.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
//G.SYSIN DD *
 &LAMISH IRE=15,IWR=16,NUMMAX=999,KPRI=1,MUSTR=120, &END
 &LAMISH IRE=15,IWR=16,NUMMAX=999,KPRI=1,MUSTR=0,
 NEU=3,2,1,6,5,4,9,8,7,12,11,10, &END
 &LAMISH IRE=0, &END
//
```

Die Eingabe-Namenliste ist hier die folgende:

```
NAMELIST /LAMISH/ IRE,IWR,MUSTR,NEU,KPRI
```

Alle Listen-Größen sind INTEGERs, NEU ist ein Feld von 1000 Elementen.

Die Bedeutungen der einzelnen Eingabe-Größen:

IRE ist die symbolische Nummer der zu korrigierenden Datei. Eine Liste mit IRE = 0 beendet die Prozedur.

IWR ist die symbolische Nummer der Hilfsdatei.

Die neue Reihenfolge der Funktionen wird entweder durch die Größe MUSTR - falls diese positiv ist - bestimmt , oder - für MUSTR $\leq$ 0 - durch die Nummern NEU(1),NEU(2), ...

MUSTR > 0 ist die Laufnummer einer "Service-Funktion" in der Datei IRE , die die neue Reihenfolge der Funktionen als Abszissen (X(i)) enthält. Die Ordinaten dieser Funktion beschreiben die alte Reihenfolge in der Datei , d.h. F(i) = i .

NEU(1),NEU(2), ...,NEU(letzt) ist die neue Reihenfolge der Funktionen in der Datei , falls MUSTR = 0 ist.

KPRI Druckintensität der Ausgabe.

Hinweis : Die Herstellung einer Service-Funktion zur Beschreibung der geäderten Reihenfolge kann man im Handumdrehen mit Hilfe der Kommandoprozedur FUNKIN (s. "Schreiben im Dialogverfahren.") erledigen .

Bemerkung : Um Fehler zu vermeiden, muß bei MUSTR $\leq$ 0 die neue Reihenfolge

```
NEU(1),NEU(2),NEU(3),  ...  ,NEU(letzt)
 eine Permutation der Ordnungszahlen
erste,erste+1,erste+2,...,erste+letzt-1
```

sein bzw. im Falle MUSTR > 0 müssen die X(i)-s der Service-Funktion eine Permutation von der F(i)-s sein.

7.3 Mehrere Funktionen der Datei werden zu einer neuen Funktion zusammengefügt.

Die Prozedur FUEGT baut aus mehreren alten Funktionen eine neue Funktion und speichert sie anschließend in einer SERVUS-Datei. Die erste dieser Funktionen wird als Präge-Funktion benutzt - d.h. die zu bildende neue Funktion übernimmt die Namen, Kenngrößen, usw. von dieser Funktion. Die Abszissen und Ordinaten der nachfolgenden Funktionen werden dann in die entsprechenden Wertebereiche der ersten Funktion eingefügt. Die Punkte der so entstandenen neuen Funktion werden dann so lange umgeordnet, bis eine streng-monoton wachsende Abszissenmenge entsteht.

Bei Punkten mit gemeinsamen Abszissen wird die streng-monotone Wachstum wie folgt erzwungen :

entweder wird nur der erste Auftritt dieses Punktes registriert,
oder wird die einzige, gemeinsame Ordinate zu dieser Abszisse als arithmetisches Mittel der ursprünglichen Ordinaten dieser Punkte gebildet,
oder aber werden die gemeinsame Abszissen durch Mikro-Veränderungen auseinandergeschoben. Die erste Funktion hat dabei die kleinste Abszisse, usw.

Ein Auftrags-Beispiel zu dieser Prozedur :

Name : tso000.SERVUS.CNTL(FUEGT)

```
//inr000m  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//UNION  EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C               PROGRAMM FUEGT
      CALL UNION
      STOP
      END
//*
//L.SYSPRINT DD DUMMY
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=inr000.test.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD DSN=inr000.test.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LAUNIO IRE=15,IWR=16,LAUF=29,-30,31,-33,23,-31,31,-51,IQ=1,
 MODUS='*NEWPAG*',KPRI=2, &END
 &LAUNIO IRE=0, &END
//
```

Die Eingabe-Namenliste seht hier wie folgt aus :

```
NAMELIST /LAUNIO/ IRE,IWR,MODUS,LAUF,KPRI,IQ
```

Alle Listen-Größen sind INTEGERs, LAUF ist ein Feld von 50 Elementen.

Die Bedeutungen der einzelnen Eingabe-Größen:

LAUF ist die Liste der Funktionen, die zusammengefügt werden sollen. Es gilt hier : mit

LAUF(k) > 0 sind die Präge-Funktionen bezeichnet, die den Charakter der neuen Funktionen bestimmen. Mit

LAUF(k) < 0 gibt man die Funktionen an, die in die vorangehende Präge-Funktion eingefügt werden sollten. Mit ein

LAUF(k) = 0 kann man die Funktionsliste abschließen.

IQ regelt die Behandlung von gemeinsamen Punkten. Bei

IQ = 0 wird die Ordinate der ersten Funktion als gemeinsame Ordinate benutzt, bei

IQ = 1 wird der Mittelwert der Ordinaten eingesetzt und bei

IQ > 1 werden die gemensamen Abszissen auseinandergeschoben.

IRE ist die symbolische Nummer der Ausgangdatei. Eine Liste mit IRE = 0 beendet die Prozedur.

IWR ist die symbolische Nummer der Schreibdatei die die neue Funktionen aufnehmen soll.

MODUS ist der Schreib-Modus. Er kann folgende Werte haben:

MODUS='OLD***'** das Schreiben in die Datei beginnt hinter dem letzten Eintrag, d.h. die alten Aufzeichnungen bleiben erhalten !

MODUS='NEW***'** das Schreiben in die Datei beginnt am Anfang, hinter der Kopfzeile, d.h. die alten Aufzeichnungen werden zerstört!

MODUS='*NEWPAG*' das Schreiben in die Datei beginnt hinter dem letzten Eintrag. Bei einer FOL8-Datei wird außerdem - im Gegensatz zum OLD-Modus - ein neues Blatt aufgeschlagen.

KPRI Ausgabe-Intensität.

Bemerkung : Die Lese- und Schreib-Dateien können hier identisch, oder auch verschieden sein, in der Schreibnorm müssen sie in jedem Fall übereinstimmen !

7.4 Einige Funktionen der Datei werden invertiert.

Die Prozedur INVERT vertauscht Abszissen mit Ordinaten bei einigen, ausgewählten Funktionen der Datei. Dabei werden auch die Achsennamen und die Datenfenster (s. "Eine SERVUS-Datei wird gesichtet und kopiert.") vertauscht. Die invertierten Funktionen erhalten die Kennnummern der Ausgangsfunktionen und werden am Ende der Datei eingetragen. Zum umkopieren wird wieder eine Wegwerfdatei (FT15F001 in dem folgendem Beispiel) benutzt.

Die Prozedur sieht aus wie folgt:

Name : tso000.SERVUS.CNTL(INVERT)

```
//inr000m  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//INVERT EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C               PROGRAMM INVERT
      CALL INVERT
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
 ENTRY MAIN
//*
//G.FT14F001 DD DSN=INR000.reform.DATA,DISP=(OLD,KEEP)
//G.FT15F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
//G.SYSPRINT DD SYSOUT=*
//G.SYSIN DD *
 &LAINVR IRE=14,IWR=15,INLIST=0,LAUF=78,87,0,
 KPRI=2, &END
 &LAINVR IRE=0, &END
//
```

Die Eingabe-Namenliste ist hier :

```
NAMELIST /LAINVR/ IRE,IWR,INLIST,LAUF,KPRI
```

Alle Listen-Größen sind INTEGERs, LAUF ist ein Feld mit 50 Elementen.

Die Bedeutungen der einzelnen Eingabe-Größen:

IRE ist die symbolische Nummer der zu SERVUS-Datei Eine Liste mit IRE = 0 beendet die Prozedur.

IWR ist die symbolische Nummer der Hilfsdatei.

INLIST , LAUF bestimmen zusammen eine Menge von Ordnungszahlen. Invertiert werden dann die Funktionen, die in der SERVUS-Datei zu diesen Ordnungszahlen gehören.

Falls

INLIST > 0 ist, besteht die Zahlenmenge aus den Ordnungszahlen

{LAUF(1),LAUF(2),...,LAUF(INLIST)},

falls

INLIST = 0 ist, sind es die Ordnungszahlen

{LAUF(1) ≤ i ≤ LAUF(2)} bzw. {LAUF(1) ≥ i ≥ LAUF(2)}

und falls

INLIST < 0 ist, hat die Zahlenmenge folgende -INLIST Elemente

{LAUF(1), [LAUF(1)+LAUF(2)], [LAUF(1)+2*LAUF(2)], ...}.

KPRI Ausgabe-Intensität.

7.5 *Eine Funktionenschar der Datei wird transponiert.*

Die in folgendem beschriebene Prozedur dient dazu, um bei einer Funktions-Oberfläche, die durch eine Funktionenschar

F(s,X) , X1 ≤ X ≤ XN , s=n1,n1+1,n1+2, ...

beschrieben wird die gemeinsamen Abszissen X mit den Scharen-Parameter s zu vertauschen. Die so erzeugte neue Funktionenschar

F(x,S) , S1 ≤ S ≤ SN , x=m1,m1+1,m1+2, ...

wird in dieselbe SERVUS-Datei eigetragen, in der die ursprüngliche Funktionenschar lag.

Die Zuordnung der neuen Abszissen S zu dem alten Scharenparameter s erfolgt mittels einer Servicefunktion **F*(X*)** die ebenfalls in der besagten SERVUS-Datei liegen muß. Die Ordinaten dieser Funktion enthalten die laufenden Nummern der Funktionen, die transponiert werden sollen, **F*** = **"s"**. Die Abszissen sind die, zu diesen Funktionsnummern gehörenden neuen Abszissen, **X*** = **"S"**. Ebenfalls von dieser Servicefunktion **F*** übernehmen die neue Funktionen die Größen (s. S. 6)

KLASSE , XMI , XMA , XED , NAMX und MASX.

Von der ersten der alten Funktionen werden die folgende Kenngrößen übernommen :

FMI , FMA , FED , NAMF , MASF und NP.

Falls die nachfolgenden alten Funktionen mehr oder weniger Punkte haben, als die erste , so werden sie abgeschnitten , oder mit Nullen aufgefüllt.

Die Numerierung der neuen Funktionen erfolgt anhand der NUMMER-Kenngröße der Funktion **F*** : die neue NUMMER-kenngrößen sind

NUMMER+1,NUMMER+2, ...

Ein Beispiel für diese Prozedur :

Name : tso000.SERVUS.CNTL(TRANSP)

```
//inr000t  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//TRANSPO EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C               PROGRAMM TRANSP
      CALL TRANSP
      STOP
      END
//*
//L.SYSPRINT DD DUMMY
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=INR000.cvvap.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
//G.SYSIN DD *
 &LATRAN IRE=15,IWR=16,KPRI=2,MUSTR=320,LAG=-75, &END
 &END
//
```

Die Eingabe-Namenliste der Prozedur ist

```
NAMELIST /LATRAN/ IRE,IWR,MUSTR,LAG,KPRI
```

Alle Listen-Größen sind INTEGERs.

Die Bedeutungen der einzelnen Eingabe-Größen:

IRE	ist die symbolische Nummer der zu SERVUS-Datei
IWR	ist die symbolische Nummer der temporären Hilfsdatei.
MUSTR	enthält die Laufnummer der Servicefunktion F*(X*) in der Datei.
LAG	verschiebt die in F* enthaltene Kurvenordnungszahlen um LAG, d.h. die laufende Nummern der zu transponierenden Funktionen sind F* + LAG.
KPRI	Ausgabe-Intensität.

Hinweis : Es empfiehlt sich die Service-Funktion **F*** mit der Kommandoprozedur FUNKIN (s. "Schreiben im Dialogverfahren.") zu erzeugen.

7.6 *Funktions-Kenngrößen werden abgeändert.*

Bei der 3D-Darstellung von Funktionen (s. "Funktionen aus einer SERVUS-Datei werden als eine Funktions-Oberfläche gezeichnet.") ist es manchmal besser, den gegenseitigen Abstand der Funktions-Ebenen mit der Kenngröße "NUMMER" (s. S. 6) zu steuern. Dazu ist es dann notwendig , diese Kenngröße der betroffenen Funktionen nachträglich entsprechend umnumerieren. Es kann aber auch notwendig sein , andere Größen der Namenszeile (s. "Eine SERVUS-Datei wird gesichtet und kopiert.") wie "KLASSE" , "NAMX" , "MASX" usw. abzuändern. Die Prozedur NEWNUM erledigt diese Aufgaben.

Der Umtausch der alten zu den neuen Größen erfolgt hier auch mittels einer Service-Funktion **F#(X#)** der betroffenen SERVUS-Datei. Die Ordinaten dieser Funktion **F#** enthalten die laufenden Nummern der Funktionen, die umnummeriert werden sollen, in **X#** sind die neuen **NUMMER** bzw. **KLASSE**-Kenngrößen gespeichert. Falls eine der CHARACTER-Größen der Funktionen : **NAMX**, **MASX**, **NAMF** oder **MASF** geändert werden soll, dann hat **X#** eine andere Bedeutung. In diesem Falle werden die neuen Namen mit einem Feld von CHARACTER*8-Größen - **NAMMAS** genannt - der Prozedur zugefürt. **X#** enthält jetzt die jeweiligen Ordnungszahl des neuen Namens in Feld NAMMAS der bei der Funktion **F#** benutzt werden soll.

Die neunumerierten Funktionen ersetzen die alte Funktionen, d.h. die Funktionen mit der alten Numerierung gehen dabei verloren !

Ein Beispiel für die Umnumerierung einer Funktionenschar :

Name : tso000.SERVUS.CNTL(NEWNUM)

```
//inr000t  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//NEWNUM EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C                PROGRAMM NEWNUM
      CALL NEWNUM
      STOP
      END
//*
//L.SYSPRINT DD DUMMY
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=INR000.cvvapm.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
//G.SYSIN DD *
 &LANUMM IRE=15,IWR=16,KPRI=2,MUSTR=321,LAG=0,IQ=2, &END
 &END
//
```

Im nächsten Beispiel werden die CHARACTER-Größen "NAMF" umgeschrieben:

```
//inr000t  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//NEWNUM EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C                PROGRAMM NEWNUM
      CALL NEWNUM
      STOP
      END
//*
//L.SYSPRINT DD DUMMY
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=INR000.inform.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
//G.SYSIN DD *
 &LANUMM IRE=15,IWR=16,KPRI=2,MUSTR=111,LAG=0,IQ=5,
 NAMMAS='ZC,VM...','DT/DRH..','PT#,V...','PT#1,V..',
 'PT#2,V..','PT#C,VM.','Z,V.....','ZC,VM0..','ZC1,VM0.','ZC2,VM0.',
 'ZC,VM...','ZC,V....','PT#C,V..',
 &END
 &LANUMM IRE=0,&END
//
```

Die Eingabe-Namenliste der Prozedur ist

```
NAMELIST /LANUMM/ IRE,IWR,MUSTR,LAG,KPRI,IQ,NAMMAS
```

NAMMAS ist hier ein Feld von 100 CHARACTER*8 Namen, alle anderen Listen-Größen sind INTEGERs.

Die Bedeutungen der einzelnen Eingabe-Größen:

IRE ist die symbolische Nummer der SERVUS-Datei. Eine Liste mit IRE = 0 beendet die Prozedur.

IWR ist die symbolische Nummer der temporären Hilfsdatei.

MUSTR enthält die Laufnummer der Service-Funktion F#(X#) in der Datei.

LAG verschiebt die in F# enthaltene Kurvenordnungszahlen um LAG, d.h. die laufende Nummern der zu transponierenden Funktionen sind F# + LAG.

IQ bestimmt die Kenngröße : für

IQ = 1 wird "KLASSE" abgeändert, für

IQ = 2 "NUMMER" .

Für 2 < IQ < 7 werden die Achsen-Beschriftungen geändert : für

IQ = 3 "NAMX" , für

IQ = 4 "MASX" , für

IQ = 5 "NAMF" und für

IQ = 6 "MASF" .

KPRI Ausgabe-Intensität.

NAMMAS Characterkonstanten zum Umschreiben der Größen

NAMX | MASX | NAMF | MASF

Hinweis : Die Service-Funktion **F#** schreibt man am besten mit der Kommandoprozedur FUNKIN (s. "Schreiben im Dialogverfahren.") in die Datei.

7.7 Umwandlung der Funktionen in ihren Logarithmen , Prüfung des monotonen Wachstums.

Die Stapel-Prozedur LOGAR errechnet für eine Funktionenschar einer SERVUS-Datei die logarithmische Darstellungen - und speichert die neue Funktionenschar in eine (andere) Datei. Benutzt werden dabei die dekadischse Logarithmen. Umgewandelt werden jeweils nur die Abszissen , jeweils nur die Ordinaten oder auch alle beide Funktionsteile.

Dieselbe Prozedur erlaubt es, bei Funktionen einer Funktionenschar die streng monotonen Wachstum zu erzwingen.

Prozedurbeispiel :

Name : tso000.SERVUS.CNTL(LOGAR)

```
//inr000t  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
//LOG10    EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSPRINT DD DUMMY
//C.SYSIN DD *
C                PROGRAMM LOGFUN
C      X ===> LOG10(X) ,  F ===> LOG10(F)
       CALL LOGFUN
       STOP
       END
//*
//L.SYSPRINT DD DUMMY
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,FENST4)
 ENTRY MAIN
//*
//G.FT15F001 DD DSN=INR105.simmeq.DATA,DISP=(OLD,KEEP)
//G.FT16F001 DD DSN=INR105.test.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LALOGA IRE=15,IWR=16,KPRI=2,INLIST=0,LAUF=13,48,
 IQ=2,MODUS='*NEWPAG*', &END
 &LALOGA IRE=0, &END
//
```

Die Eingabe-Namenliste der Prozedur ist

```
NAMELIST /LALOGA/ IRE,IWR,MODUS,INLIST,LAUF,KPRI,LQ,IQ
```

Alle Listen-Größen sind INTEGERs.

Die Bedeutungen der einzelnen Eingabe-Größen:

IRE ist die symbolische Nummer der Lesedatei. Eine Liste mit IRE = 0 beendet die Prozedur.

IWR ist die symbolische Nummer der Schreibdatei.

INLIST , LAUF bestimmen zusammen eine Menge von Ordnungszahlen. Umgewandelt werden dann die Funktionen, die in der SERVUS-Datei zu diesen Ordnungszahlen gehören.

	Falls
INLIST > 0	ist, besteht die Zahlenmenge aus den Ordnungszahlen
	{LAUF(1),LAUF(2),...,LAUF(INLIST)},
	falls
INLIST = 0	ist, sind es die Ordnungszahlen
	{LAUF(1) ≤ i ≤ LAUF(2)} bzw. {LAUF(1) ≥ i ≥ LAUF(2)}
	und falls
INLIST < 0	ist, hat die Zahlenmenge folgende -INLIST Elemente
	{LAUF(1), [LAUF(1)+LAUF(2)], [LAUF(1)+2*LAUF(2)], ...}.

LQ bestimmt die Art der Umwandlung : für

LQ = 1	erfolgt X = = = > LOG10(X) , für
LQ = 2	erfolgt F = = = > LOG10(F) und für
LQ = 3	werden sowohl die Abszissen als auch die Ordinaten umgewandelt.

IQ : für **0 ≤ IQ ≤ 2** werden die ausgewählten Funktionen streng monoton wachsend gemacht. IQ regelt auch was bei identischen Abszissen mit den Ordinaten geschehen soll (s. auch "Mehrere Funktionen der Datei werden zu einer neuen Funktion zusammengefügt.") : für

IQ = 0	wird die Ordinate der ersten Funktion als gemeinsame Ordinate benutzt, bei
IQ = 1	wird der Mittelwert der Ordinaten eingesetzt und bei
IQ > 1	werden die gemensamen Abszissen auseinandergeschoben.
KPRI	Ausgabe-Intensität.

8.0 Funktionen aus einer SERVUS-Datei werden ausgedruckt.

Funktionen, die in einer SERVUS-Datei aufgezeichnet sind, kann man mit dem Schnelldrucker - im Rahmen eines Hintergrund-Jobs - ausdrucken lassen.

Die Funktionen werden als Zahlen-Kolonnen ausgedruckt : die Ordinaten einer Funktion erscheinen untereinander. Die Kolonne kann entweder mit dem ersten, oder mit dem letzten Funktions-Element beginnen. Auf jede Druckseite, über der Kolonne werden - als Beschriftung - beide Namen NAMF und MASF der betroffenen Funktion sowie die Kenngrößen KLASSE, NUMMER ausgedruckt (s. Abbildung 6 auf Seite 115).

Man kann sich darauf beschränken, nur die Ordinaten einer Funktion auszudrucken. In diesem Fall können bis zu 8 Funktionen neben einander auf einer Druckseite Platz finden. Man kann aber auch eine Abszissen-Kolonne am linken Rand des Druckseite ausgeben, oder für jede Funktion Abszissen und Ordinaten ausdrucken. In diesem letzten Falle passen dann nur noch 4 Funktionen auf eine Seite.

Die Genauigkeit der Ausgabe richtet sich nach der auszudruckenden Datei ; GRA8 und FOL8 Dateien werden mit 8-Stellen ausgedruckt, die anderen mit 6.

Die Hintergrund-Prozedur DRUCKT sieht aus wie folgt:

Name : tso000.SERVUS.CNTL(DRUCKT)

```
//inr000p  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1)
// EXEC F7CLG,USER='INR105.SERVUS.LOAD'
//C.SYSIN DD *
C              PROGRAMM DRUCKT
      CALL PAGEFF
      STOP
      END
//*
//L.SYSIN DD *
 INCLUDE SYSLIB(SERDIO,COPYDA,NEXT)
 ENTRY MAIN
//*
//G.FT16F001 DD DSN=inr000.scdar3.DATA,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LADRUC IRE=16,INLIST=0,LAUF=220,225,MODPRI='UP.EINX1',KPRI=0,
 &END
 &LADRUC IRE=0, &END
//
```

Eingabebeschreibung: Die Daten werden im NAMELIST-Format eingegeben. Die Liste lautet :

```
NAMELIST /LADRUC/ IRE,INLIST,LAUF,MODPRI,JOAN,JOEN,JO,KPRI
```

Bei den Listen-Größen ist nur MODPRI ein CHARACTER*8 Name, alle anderen Daten sind INTEGERs. LAUF ist ein INTEGER-Feld mit 50 Elementen.

Die Bedeutungen der einzelnen Eingabe-Größen:

IRE ist die symbolische Nummer der auszudruckenden Datei. Eine Eingabe-Liste mit IRE = 0 stoppt die Prozedur.

INLIST , LAUF bestimmen zusammen eine Menge von Ordnungszahlen. Ausgedruckt werden dann die Funktionen, die in der Datei zu diesen Ordnungszahlen gehören.

Falls
INLIST > 0 ist, besteht die Zahlenmenge aus den Zahlen

{ LAUF(1),LAUF(2),...,LAUF(INLIST) }.

Bei dieser Eingabe kann man die Besetzung eines Blattes steuern : ein positives LAUF(j) bedeutet, daß mit der LAUF(j)-ten Funktion der Datei ein neues Blatt beginnt, ein negatives LAUF(j+1) bewirkt, daß die [-LAUF(j+1)]-te Funktion auf dem aufgeschlagenen Blatt ausgegeben wird, usw. LAUF(k)=0 schließt die Druckseite ab.

INLIST = 0 entspricht der folgende Zahlenmenge :

{ LAUF(1) ≤ i ≤ LAUF(2) } bzw. { LAUF(1) ≥ i ≥ LAUF(2) }.

Falls
INLIST < 0 ist, hat man -INLIST Ordnungszahlen:

{ LAUF(1), [LAUF(1)+LAUF(2)], [LAUF(1)+2*LAUF(2)], ... }.

Bei den beiden letzten Eingaben-Typen, INLIST=0 und INLIST<0 werden die Blätter "dicht gepackt", d.h. es werden soviel Kolonnen wie möglich auf einen Blatt untergebracht.

MODPRI steuert die Besonderheiten des Ausdruckes. für
MODPRI='DN.....' wird mit dem ersten Element begonnen, für
MODPRI='UP.....' mit dem letzten Element der Funktion.
MODPRI='.....0' bedeutet, daß nur die Ordinaten gedruckt werden
MODPRI='.....1' daß die Abszissen der - auf dieser Seite - ersten Funktion auch erscheinen und bei
MODPRI='...sonst' werden zu jeder Funktion Abszissen und Ordinaten ausgedruckt.
JOAN,JOEN,JO "Rhythmus" des Ausdruckes. Der Druck beginnt mit dem JOAN-ten Element und hört mit dem JOEN-ten auf. In diesem Bereich wird jede JO-te Element ausgedruckt.
KPRI ist wieder ein Maß für die Intensität der Ausgabe. Die Funktions-Elemente werden hier aber auch bei KPRI=0 ausgegeben.

Der Ausdruck, der in den Abbildung 5 - Abbildung 7 dargestellt ist, entspricht der Eingabe :

```
&LADRUC IRE=16,MODPRI='DN.EINX1',INLIST=7,JOAN=1,JO=1,
LAUF=4,-5,-3,-6,-13,-14,-1, &END
```

Man kann auch bei der Druck-Prozedur die Zusammenstellung der Steuerkarten einer Kommandoprozedur - "DRUDA" (s. "Die SERVUS-Kommandoprozeduren.") - überlassen :

```
EX 'tso000.SERVUS.CLIST(DRUDA)'.
```

9.0 Funktionen aus einer SERVUS-Datei werden als eine Kurvenschar gezeichnet.

9.1 Über das Abbildungsverfahren.

Die im folgenden (Kap. 9 und 10) beschriebene bildhafte Ausgabe der gespeicherten Funktionen beruht auf dem Graphischen System (GS) der HDI /3/. Dabei werden zum Zeichnen, Schreiben und bei der Herstellung der Bild-Rahmen die im TRACEGS-System /5/ zusammengefaßten Routinen verwendet.

GS erlaubt es, die gezeichneten Bilder zu speichern. Diese Bildspeicherung macht eine vielfältige nachträgliche Handhabung der einmal gezeichneten Bilder möglich : der Benutzer kann

- von dem Bild beliebig viele Abzüge in verschiedenen Größen und an verschiedenen Zeichengeräten zu fertigen,
- das Bild nachträglich - mit Hilfe von GS-Befehlen, oder mit der im Kap. 12.3 beschriebenen GS-Prozeduren - ändern (z.B. den Namen oder den Titel den des Bildes),
- ein GDF-Abzug von dem Bild machen (Kommandoprozedur IMMANI). Diese Abzüge bringt man dann mit Hilfe der GDDM-Menü in eine sog. Segment-Bibliothek. Das Textverarbeitungssystem DCF holt dann diese Segmente und fügt sie als Bilder in den vorgesehenen Stellen in den Text ein (ausführlicher in /1/ , s. 20).

Für die Bildspeicherung braucht der Benutzer ein "Buch" - eine gegliederte Datei mit der Namens-Form 'inr000.gsbook' - in die nachher die Bilder als Glieder (Members) eingfügt werden (s. /3/ oder /5/) . Diese Datei kann man mit der folgenden TSO-Kommandofolge anlegen:

```
ATTRIB buch BLKSIZE(19069) RECFM(U)
ALLOC DA('inr000.gsbook') NEW UNIT(DISK) VOLUME(bat00c)  +
      SPACE(10,10) TRA DIR(5) USING(buch)
FREE ATTRLIST(buch)
```

Die kleingedruckten Namen, sowie den Raumbedarf (10,10) und (5) muß der Benutzer ausfüllen.

Die abzubildende Datei muß in GRA4 normiert sein !

9.2 Stapel-Verfahren für die Kurven-Darstellung.

Die Prozedur FIGURB erzeugt Kurvendarstellungen der gespeicherten Funktionen im Stapelbetrieb. In einem Auftrag kann mann bis zu 100 Bilder - Kurven-Darstellungen - erzeugen, die dann zu einem Gesamtbild zusammengelegt und durch eine der nachgeschalteten Zeichengeräte (XYNETICS, VERSATEC oder Farb-VERSATEC) ausgegeben werden. Man kann auch dieses Gesamtbild unter den Namen T001 in einem "Buch" speichern.

Der Benutzer kann das Aussehen der Bilder weitgehend nach eigenem Vorstellungen gestalten - oder aber auch Standard-Darstellungen verwenden, die sehr wenig Aufwand bei der Eingabe erfordern.

In der Regel - bei Mindesteingabe - werden zu jeder gezeichneten Funktion die Funktions-Namen NAMF(i) sowie die Kenngrößen KLASSE(i) und NUMMER(i) in das Bild miteingezeichnet. NAMF (und MASF) der ersten Funktion des Bildes werden - im Regelfall - neben die Ordinaten-Achse gedruckt. KLASSE(1) und NUMMER(1) erscheinen an einer - von dem Benutzer bestimmten - Stelle des Bildes. Die Kenn-Nummern und Namen der folgenden Funktionen (wenn es welche gibt) werden übereinander, in einer "wachsender" Kolonne ausgegeben und zwar über die Kenn-Nummer der ersten Funktion beginnend. Über die Abszissen-Achse werden NAMX und MASX der ersten Funktion geschrieben (s. Abbildung 8). In der Regel werden die einzelne Teilbilder im T001 mit 1 beginnend durchnumeriert und sie erhalten keine Bildtitel außer der Bildnummer. Die "Abb. 1" in Abbildung 8 zeigt ein Bild, das mit Mindesteingabe erzeugt wurde (s. erste Eingabekarte bei der nachfolgend abgedruckten Prozedur).

Es besteht aber auch die Möglichkeit, Bildnummer und Bildtitel vorzugeben. Man kann auch für die einzelnen Kurven anstatt der Kenn-Nummern und Namen Beschriftungen vorgeben. Auch die Achsen-Beschriftungen kann man nach eigenen Vorstellungen gestalten. Außerdem kann man - mit Hilfe der Dateien tso000.MEMORY(ABB1) ... ABB4) - Bildtitel mit Groß- und Kleinschrift und mit Sonderzeichen zeichnen zu lassen (Teilbild rechts oben in Abbildung 8 auf Seite 117). Die Regeln zum Reinschreiben der Vorlagen in die obigen Dateien findet man im TRACEGS-Handbuch /5/ unter "TEXT" (S. 35). Schließlich kann man - zusätzlich zu den Funktionen - eine Null-Linie in das Bild eintragen lassen.

Die Liste der Hintergrund-Prozedur FIGURB :

Name : tso000.SERVUS.CNTL(FIGURB)

```
//inr000f  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1),
//    REGION=4096K
//OLDVERS EXEC F7CLG,PARM.C='LANGLVL(77)',
// PARM.L='MAP,LIST',PLOT=GS7,PLOU=V,USER='SYS2.TRACEGS7'
//C.SYSIN DD *
C      F I G U R B    ( STAPELVERSION VON FIGUR )
       CHARACTER*8 INST
       DATA IRE/15/,IDIN/5/,NETZ/2/,MEM/30/,KPRI/2/,INST/'INR '/
       CALL FIGURB(IRE,IDIN,NETZ,MEM,KPRI,INST)
       STOP
       END
//*
//L.BIBS DD DSN=INR105.SERVUS.LOAD,DISP=SHR
//L.SYSIN DD *
 INCLUDE BIBS(SERDIO,FIGURB,NEXT,PAPIER,FENST4,NARS)
 ENTRY MAIN
//*
```

```
//**VRFOUT  DD DUMMY
//G.PLOTPARM DD *
 &PLOT LUNIT=26, &END
//G.FT26F001 DD SYSOUT=*
//G.COMM     DD SYSOUT=*
//G.FT15F001 DD DSN=INR105.VIELBILD.DATA,DISP=(OLD,KEEP)
//G.FT31F001 DD DSN=TSO105.MEMORY(ABB1),DISP=(OLD,KEEP)
//G.FT32F001 DD DSN=TSO105.MEMORY(ABB2),DISP=(OLD,KEEP)
//G.FT33F001 DD DSN=TSO105.MEMORY(ABB3),DISP=(OLD,KEEP)
//G.FT34F001 DD DSN=TSO105.MEMORY(ABB4),DISP=(OLD,KEEP)
//G.TRACEGS7 DD DSN=INR105.GSBOOK,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LAFIGU INLIST=1,LAUF=1, &END
 &LAFIGU INLIST=2,LAUF=7,-8,48*0,ABTEX='  ZWEITENS',NUL=1,NABB=5,
 LSR=1,LS=2,1,0,0,LST=3,
 KFR=1,KF=0,1,2,3,4,5,6,0,1,2,3,4,5,6,0,1,2,3,4,5,KFT=5,
 FMIST=-5.,FMAST=5.,FEDST=1.,XZUS$=5.,FZUS$=5., &END
 &LAFIGU INLIST=2,LAUF=15,-16,48*0,ABTEX='  DRITTENS ',IQS=-8,
 FMIST=-300.,FMAST=200.,FEDST=100.,XZUS$=10.5,FZUS$=1., &END
 &LAFIGU INLIST=3,LAUF=17,-12,-2,47*0,ABTEX='  VIERTENS ',
 TEXTX='       HOEHE ( CM )',TEXTF=' GESCHWINDIGKEIT ( CM/SEC )'/,
 TEXTZ='ERSTE  KURVE','ZWEITE KURVE','DRITTE KURVE',IQ=0,0,0,0,
 FMIST=-5.,FMAST=3.,FEDST=1.,ITX=31, &END
 &LAFIGU INLIST=-5,LAUF=18,18,48*0,ABTEX='  FUENFTENS',NUL=1,ITX=0,
 IQS=0,XZUS$=13.,FZUS$=1.0,FMIST=-150.,FMAST=150.,FEDST=50., &END
 &LAFIGU INLIST=3,LAUF=17,-12,-2,47*0,ABTEX='  SECHSTENS',IQS=2,
 FMIST=-5.,FMAST=3.,FEDST=1., &END
 &LAFIGU LAUF(1)=0, &END
//
```

Die abgedruckte Stapel-Prozedur holt die zu zeichnende Funktionen aus der Datei inr000.vielbild.DATA und speichert das fertige Bild (Abbildung 8) in das "Buch" inr000.gsbook. Die Bilder werden mit Hilfe des alten Versatec-Gerätes gezeichnet. Die Datei tso000.MEMORY(ABB1) wird bei der 4. Eingabe (ITX=31) gebraucht um den groß/klein-geschriebenen Bildtitel zu übertagen.

Falls man die Bilder am Farb-Versatec-Gerät ausgeben will, dann sieht die Prozedurkopf wie folgt aus :

```
//inr000f  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1),
//   REGION=4096K
//COLOR EXEC F7CLG,PARM.C='LANGLVL(77)',
// PARM.L='MAP,LIST',PLOT=GS7,PLOU=X,USER='SYS2.TRACEGS7'
//C.SYSIN DD *
C     F I G U R B   ( STAPELVERSION VON FIGUR )
      CHARACTER*8 INST
      DATA IRE/15/,IDIN/5/,NETZ/2/,MEM/40/,KPRI/2/,INST/'INR '/
      CALL FIGURB(IRE,IDIN,NETZ,MEM,KPRI,INST)
      STOP
      END
```

Eingabebeschreibung:
Die Eingabe besteht aus allgemeinen Angaben zum Zeichnungs-Verfahren und aus speziellen Anweisungen für jedes einzelne Bild. Die allgemeinen Verfahrens-Hinweise

sind im Aufruf-Kopf der Prozedur enthalten und man kann sie in der DATA-Anweisung ändern. Diese Steuergrößen sind:

```
( IRE,IDIN,NETZ,MEM,KPRI,INST ).
```

Von diesen Größen ist nur INST ein CHARACTER*8 Wort, alle anderen sind INTEGERs. Sie haben die folgenden Bedeutungen:

IRE ist die symbolische Nummer der SERVUS-Datei,
IDIN ist die DIN-Größe des Bildes.
IDIN > 0 zeichnet das Bild in DIN-A-"IDIN"-Querformat und
IDIN < 0 in DIN-A-"-IDIN"-Hochformat.
NETZ > 0 erzeugt ein Trennungsnetz zwischen den Bildern. Falls
NETZ 2 | -2 ist und ABS(IDIN) < 5, dann erhält jedes Teilbild das KFK-Emblem (s. z.B. Abbildung 11).
MEM wählt den Zeichengerät: für
MEM= 1 | 10 wird die Schwarzweiss-Version des Farb-VERSATECs benutzt, für
MEM= 2 | 20 der XYNETICS-Plotter für
MEM= 3 | 10 der alte VERSATEC-Plotter und für
MEM= 4 | 30 wird die Farb-Version des Farb-VERSATECs benutzt. Falls
MEM= 10 | 20 | 30 | 40 ist, dann wird das Gesamtbild T001 auch gespeichert.
KPRI ist die Ausdruck-Intensität des Begleit-Protokolls.
INST 8 Zeichen, die auf dem KfK-Emblem mit erscheinen (Institutskürzel, +INR + auf der Abbildung 11 auf Seite 120).

Die speziellen Bild-Informationen werden im NAMELIST-Format eingegeben, für jedes Teil-Bild eine Liste "LAFIGU" :

```
NAMELIST /LAFIGU/ INLIST,LAUF,XMIST,XMAST,XEDST,FMIST,FMAST,FEDST,
       SCR$,XABB$,FABB$,XAUF$,FAUF$,XRAR$,FRAR$,XZUS$,FZUS$,
       TAX$,TAF$,TAZ$,NABB,NUL,ITX,IQ,IQS,ABTEX,TEXTX,TEXTF,TEXTZ
       LSR,LST,LS,KFR,KFT,KF
```

Von den Listen-Größen sind ABTEX, TEXTX und TEXTF CHARACTER*70 Namen, TEXTZ ist ein Feld von 20 CHARACTER*70 Elementen, NABB, INLIST, ITX, LSR, LST, KFR, KFT IQS und NUL sind INTEGERs, LS, KF, IQ und LAUF sind INTEGER-Felder von je 20, 20, 20 bzw. 50 Elementen. Alle anderen Daten sind REAL*4 Zahlen.

Die Bedeutungen der einzelnen Listen-Größen:

INLIST , LAUF dürfen auch bei Mindesteingabe nicht fehlen. Sie bestimmen zusammen die Zahlenmenge der zu zeichnenden Funktionen.

Falls
INLIST > 0 ist, besteht sie aus den Zahlen

$$\{LAUF(1), LAUF(2), \ldots, LAUF(INLIST)\}.$$

Bei dieser Eingabe kommt nach der ersten Funktion eine weitere nur dann in das angefangene Bild, falls das zugehörige LAUF(j) negativ ist. Ein Index LAUF(k) ≥ 0 beendet das Bild. Falls
INLIST = 0 ist, ist die Zahlenmenge die folgende:

$$\{LAUF(1) \leq i \leq LAUF(2)\} \quad \text{bzw.} \quad \{LAUF(1) \geq i \geq LAUF(2)\}$$

und falls

INLIST < 0 ist, hat man die -INLIST Ordnungszahlen:

{LAUF(1), [LAUF(1)+LAUF(2)], [LAUF(1)+2*LAUF(2)], ...}.

Bei den Eingabe-Typen INLIST = 0 und INLIST < 0 werden - im Gegensatz zu dem "INLIST > 0"-Fall - alle Funktionen, die durch die Zahlenmenge bestimmt sind, in dasselbe Bild eingezeichnet.

LAUF(1) = 0 eine Eingabe-Liste mit LAUF(1)=0 stoppt die Prozedur FIGURB.

Die folgenden Daten umreissen das - für das Bild vorgesehene - Datenfenster (s. auch "Über SERVUS-Dateien."und "Eine SERVUS-Datei wird gesichtet und kopiert."):

XMIST,XMAST bestimmen den erlaubten Werte-Bereich auf der X-Achse,
FMIST,FMAST den entsprechenden Bereich auf der F-Achse.
XEDST,FEDST sind die vorgegebenen Skalenlängen (s. S. 6).

Falls bei einer Eingabe XMIST ≥ XMAST ist, oder XEDST = 0, dann benutzt die Prozedur anstelle dieser Größen XMI, XMA und XED von der ersten Funktion des Bildes. Falls auch diese Größen ungeeignet sind, dann werden diese Daten anhand der Abszissenmenge von der Prozedur ermittelt. Entsprechendes gilt bei der Größen FMIST, FMAST und FEDST.

Die folgenden Daten bestimmen das Aussehen des Bildes:

SCR$ ist die Höhe der verwendeten Schriftzeichen.
XAUF$,FAUF$ sind die Koordinaten der linken, unteren Ecke des inneren Rahmens,
XRAR$,FRAR$ die Abstände zum Bildrand am rechten, bzw. am oberen Ende des inneren Rahmens.
XZUS$,FZUS$ bestimmen den Anfang der Kolonne, in die die Kennun gen der einzelnen Funktionen kommen, d.h. die Stelle, wo KLASSE(1), NUMMER(1) hingeschrieben wird.
XABB$,FABB$ sind die Anfangs-Koordinaten des Bildtitels.

Alle hier beschriebenen Größen SCR$,XAUF$ - ... - FABB$ sind in den relativen Einheiten UNITX bzw. UNITF anzugeben : UNITX ist 1/32. der X-Ausdehnung des Bildes, UNITF ist 1/32. der F-Ausdehnung.

**TAX$** ist der Abstand der X-Achsen-Beschriftung über der X-Achse in Einheiten der Schrifthöhe. Bei TAX$ < 0 kommt die Schrift unter die X-Achse.
**TAF$** ist der Abstand der F-Achsen-Beschriftung rechts von dieser Achse (in Schrifthöhe). Bei TAF$ < 0 liegt die Schrift links von der F-Achse.
TAZ$ in Schrifthöhe-Maß legt die Größe der Kurven-Markierungen fest.

Die Abbildung 9 zeigt die Bedeutung der einzelnen Größen XAUF$ ...

Die restlichen Daten beziehen sich auf die Beschriftungen und auf die Kurvendarstellung.

NABB ist die vorgesehene Bild-Nummer. Falls
NABB = 0 ist, werden die Bilder durchnumeriert.
NUL > 0 erzeugt eine Null-Linie zusätzlich.

ITX entscheidet über die Beschriftung :

ITX = 0 erzeugt ein Bild, bei der nur ABTEX und die Funktionsgrößen KLASSE(i), NUMMER(i), NAMF(i), MASF(1) und MASX(1) zum Beschriften benutzt werden. Bei
ITX > 0 werden die Achsen mit TEXTX bzw TEXTF beschriftet und die Kurven mit den entsprechenden TEXTZ(i). Falls hierbei
ITX = 30+j ist, mit 0 < j < 5, dann kommt der Bildtitel aus der der Datei MEMORY(ABBj), ansonsten wird wieder ABTEX benutzt.
ABTEX enthält den Bildtitel (≤ 70 Zeichen),
TEXTX,TEXTF enthalten die Beschriftungen für die X- bzw. für die F-Achse (je ≤ 70 Zeichen).
TEXTZ(1), ...,TEXTZ(20) ermöglichen es, die einzelnen Kurven zu beschriften (je ≤ 70 Zeichen).

IQ und IQS wählen die Kurvendarstellung. Bei ITX > 0 wird IQ benutzt, sonst IQS.

IQ(i) > 0 zeichnet die i-te Kurve als eine Linie, mit jeden IQ -tem Punkt markiert,
IQ(i) = 0 zieht eine unmarkierte Linie und
IQ(i) < 0 zeigt nur jeden IQ-ten Punkt der Funktion. Falls
IQS > 0 ist, wird auf der ersten Kurve jede IQS-te Punkt auf den zweiten jede IQS-1-te usw. markiert.
IQS = 0 erzeugt unmarkierte Kurven und bei
IQS < 0 werden alle Punkte ohne Verbindungen gezeigt.

LSR, LST, LS steuern die Linienstärke beim zeichnen. Der Wertbereich dieser Integers ist [0,20], größere Zahlen bedeuten einen stärkeren Strich.

LSR ist für die Rahmen-Zeichnung zuständig,
LST steuert den Bildtitel und das KfK-Emblem,
..,LS(i),.. bestimmt die Dicke der i-ten Kurve.

KFR, KFT, KF bestimmen die Farbe des Gezeichnetem. Die Zahl-Farbe-Zuordnung stammt aus der GS /3/ :

0	rot
1	hellblau
2	grün
3	weiß | schwarz (kontrast)
4	magenta
5	dunkelblau
6	gelb
7	farblos
8	weiß

usw.

KFR bestimmt die Farbe des Rahmens,
KFT die Farbe des Bildtitels,
..,KF(i),.. bestimmt die Farbe der i-ten Kurve.

Bemerkung zu Strichstärke : Es empfiehlt sich nicht, mit Zahlen > 3 zu arbeiten : die Striche werden zu dick und die Bilder verbrauchen zu viel Speicherplatz.

Bemerkung zur Text-Dateien 31-34 (MEMORY(ABB1)-MEMORY(ABB4)) : FIGURB erwartet, das die Texte in diesen Dateien für eine Papier Größe DIN A 4-Querformat zugeschnitten sind. Falls IDIN eine andere Größe vorschreibt, formt FIGURB die Texte entsprechend um.

Bemerkung zur Bildspeicherung : die durch die NAMELIST-Reihe hintereinander abgerufene Einzelbilder erscheinen am nachgeschalteten Zeichengerät als ein einziges Bild. Falls es zur Bildspeicherung kömmt, wird dieses Gesamtbild gespeichert. Dieses gespeicherte Bild (Bildname = T001) ist wie folgt in Teilbilder aufgegliedert: das zuerst erzeugte Bild wird als

```
T001.RAHMEN01   ,   T001.INHALT01   ,   T001.TEXT01
```

aufbewahrt , das zweite als

```
T001.RAHMEN02   ,   T001.INHALT02   ,   T001.TEXT02
```

usw. Die Bildteile INHALT0j erfahren eine weitere Unterteilung :

```
INHALT0j  =  { XTEXT , FTEXT , KURVEN , KURVTX } .
```

In den Teil RAHMEN0j kommt nur der äußere Rahmen des Teilbildes. Die Funktionen, der innere Rahmen, die Achsen- und die Kurven werden im Teil INHALT0j gespeichert.

Die X- bzw. F -Achsen (mitsamt Beschriftung) kommen dabei in XTEXT, bzw. FTEXT, die Funktionszüge in KURVEN und die Kurvennamen in KURVTX. Der Teil TEXT0j nimmt den Bildtitel auf.

Diese Gliederung erleichtert es, die gespeicherte Bilder mit Hilfe der GS-Kommandos /3/ nachträglich zu Korrigieren.

9.3 *Kurven-Darstellung im Dialog.*

Funktionen einer SERVUS-Datei kann man auch im Dialogverfahren zu Bildern zusammenstellen und an einem Graphischen Arbeitsplatz (für die Liste der geeigneten Geräte s. /3/) anschauen. Der Vorteil dieses Verfahrens gegenüber dem Stapelbetrieb ist, daß man die Darstellungen, Beschriftungen, usw. sofort korrigieren kann, der Hauptnachteil ist, daß man in diesem Modus keinen direkten Zugang zu den Zeichengeräten hat.

Man kann dieses Hindernis umgehen, indem man die fertigen Bilder im Buch 'inr000.gsbook' speichert. Die gespeicherten Bilder kann man dann anschließend - zum Beispiel mit Hilfe der Prozedur B8 /3/ , BIPLOT /6/ oder BRAUS ("Die SERVUS-Kommandoprozeduren.") - an einem der Geräte XYNETICS ,VERSATEC oder Farb-VERSATEC zeichnen lassen. Abbildung 10 auf Seite 119 zeigt ein Bild, das auf diesem Wege - FIGUR - GSBOOK - BRAUS - entstand.

Die Prozedur FIGUR entspricht im wesentlichen der Stapel-Prozedur FIGURB. Sie benötigt aber eine zusätzliche Datei, 'tso000.MEMORY(LAFIGU)', um die - erfahrungsgemäß mühsam erarbeiteten - Daten des Bilder-Rahmens und die Rahmen-Beschriftungen speichern zu können. Diese Datei enthält in der Regel die letzte Version der schon besprochenen Namenliste LAFIGU (s. S. 44). Anders als FIGURB zeichnet FIGUR immer nur in DINA-4-Größe. Es ist auch bei FIGUR möglich, mit Hilfe der Dateien MEMORY(ABB1) - ABB4) vorgefertigte Bild-Texte in Bereitschaft zu halten.

```
Prozedur-Aufruf : EX 'tso000.SERVUS.CLIST(FIGUR)' .
```

In der ersten Prozedur-Stufe läuft FIGUR als Kommandoprozedur ab (s. auch die Bemerkungen in "Die SERVUS-Kommandoprozeduren."). Diese Kommandoprozedur erfragt von dem Benutzer

1. seinen graphischen Arbeitsplatz,
2. den Namen der SERVUS-Datei , die die Kurven bereit hält,
3. den Namen des "Buches" zum Bildspeichern,
4. das Format des Bildes (Hoch oder Querformat),
5. die Art der Quelle, die einzelne Angaben über Rahmen, Schriftgröße, Beschriftungen usw. des Bildes enthält (externe Datei oder die FORTRAN-Prozedur),
6. den vorgesehenen Bildnamen,
7. ob es eine KfK-Emblem ins Bild kommen soll und
8. ob die Herstellungsdatum ins Bild eingetragen werden soll.

   Zusätzlich erfragt FIGUR vom Benutzer
9. den vollen Namen der Datei, der die Rahmen-Angaben enthält (in der Regel "tso000.memory(lafigu)") und
10. die Glied-Namen von der Hilfs-Dateien, die den Bildern Texte zuführen sollen ("tso000.MEMORY(abb1)", ... , (abb4)").

Dann werden die benötigte Dateien zusammengefügt und die Führung mitsamt der Eingabedaten an den FORTRAN-Prozedur FIGUR übergeben.

Bemerkung zu der Frage 5 : beim ersten Benutzen der Prozedur ist die Datei LAFIGU in der Regel noch nicht gefüllt, es empfehlt sich deshalb in diesem Falle als Quelle die

Standard-Daten der FORTRAN-Prozedurzu nehmen.: In der zweiten Prozedur-Stufe beginnt ein FORTRAN77-Dialog. Hier wird der Benutzer mit Hilfe von Entscheidungs-Tafeln ("Menüs") nach den gewünschten Funktionen abgefragt, sowie nach den Daten, die zur Rahmen-Herstellung und für die Beschriftungen notwendig sind. Das Hauptmenü sieht wie folgt aus :

```
PROGRAMMSTATUS / 0 / =:

  "< 0"   STOP :
  "  1"   NEUE KURVEN ZUSAMMENSUCHEN :
  "  2"   NEUE RAHMEN-DATEN :
  "  3"   TEXTDATEI - TEXTKORREKTUR :
  "  4"   KURVENGESTALTUNG :
  "  5"   BILDAUSGABE :
```

Das Eintippen einer negativen Zahl in dieser Menü beendet das Programm, eine 5 zeigt das Bild, 0 und die nicht aufgeführten Zahlen bringen keine Entscheidung.

Das Untermenü 1 (eine 1 wurde als Antwort eingetippt) hilft dem Benutzer die für das Bild benötigten Funktionen zusammenzusuchen. Er wird hier zunächst nach der Ordnungszahl dieser Funktionen in der Datei (" LOS ") abgefragt.

Mit den Funktions-Nummern wird, wie in FIGURB verfahren: eine nicht negative Zahl, LOS $\geq$ 0 beendet die Kurven-Zusammenstellung, eine negative Zahl bewirkt dagegen, daß die (-LOS)-te Funktion in das angefangene Bild mit hineinkommt.

Der Benutzer kann die Kurven eines Bildes einzeln aus der Datei zu sammensuchen. In diesem Falle muß er bei der entsprechenden Frage JEDE = 0 setzen. Er kann aber - sofern die Funktionen in der Datei den gleichen "Abstand" zueinander haben - ein halbautomatisches Suchverfahren be nutzen : wenn er z.B. JEDE = 6 setzt, dann werden die Funktionen LOS, LOS + 6, LOS + 12, ... aus der Datei ausgesucht bis zum erreichen der Ordnungszahl LETZTE. Danach kann die Suche manuell weitergehen. Falls man JEDE negativ setzt, dann geht das Suchen von den hohen zu den niedrigen Ordnungszahlen : LOS, LOS-6, LOS-12, usw.

Jedesmal, wenn eine Funktion bereit steht, wird ihre Namenszeile ausgedruckt (wie auf der Abbildung 4 auf Seite 113) und der Benutzer um Zustimmung gebeten.

```
IST DIE KURVE IN ORDNUNG ? / 0 / =:
( "0":  JA,  "1":  "FAKTOR" NACHPRUEFEN,  "9":  NEIN )
```

Hier besteht die Möglichkeit, falls die Funktion - verglichen mit den anderen Funktionen - zu klein oder zu groß ist - mit "1" zu antworten. Bei dieser Antwort wird vom Benutzer ein Faktor erfragt, mit dem FIGUR dann die Ordinaten multipliziert.

Das Untermenü 2 dient zur Herstellung des Bildrahmens. Es hat folgende Optionen :

```
RAHMENGESTALTUNG  / 0 / =:

    "  0"   ALLES IN ORDNUNG ,
    "  1"   NEUER INNERER RAHMEN :
    "  2"   SCHRIFTGROESSEN :
    "  3"   BILDTITEL VERSCHIEBEN :
    "  4"   ACHSENTEXTE VERSCHIEBEN :
    "  5"   KURVENNAMEN VERSCHIEBEN :
    "  6"   EXTRA NULLLINIE :
    "  7"   STANDARD-MARKIERUNGEN :
    "  8"   RAHMENDATEN AUSDRUCKEN :
```

Die Daten, die in dieser Untermenü erfragt werden sind in der Abbildung 9 auf Seite 118 zusammengestellt. Die Antwort "7" dient zur Ändernung der Größe IQS (s. LAFIGU , S. 44).

Das Untermenü 3 ist folgendes Befragungsmuster :

```
TEXTKORREKTUR / 0 / =:

    "  0"   ALLES IN ORDNUNG ,
    "  1"   TEXTQUELLEN :
    "  2"   TEXT BEI DER X-ACHSE :
    "  3"   TEXT BEI DER F-ACHSE :
    "  4"   BILDTITEL :
    "  5"   FARBEN UND LINIENSTAERKEN DER BESCHRIFTUNGEN :
```

In diesem Menü kann man die Achsen-, die Funktions-Beschriftungen sowie den Bildtitel - anhand der Funktions-Kenngrößen - selber schreiben und diese Texte dann für mehrmalige Anwendung speichern. Die Antwort "1"

```
TEXTQUELLEN / "ITX" / =:

"   0":  ES WERDEN NUR DIE KURVEN-KENNGROESSEN
         UND ABTEX VERWENDET,
"> 0":  GESPEICHERTE TEXTE,
"> 30":  GESPEICHERTE TEXTE, BILDTITEL AUS DER DATEI "ITX"
```

entscheidet über die Art der Beschriftung des Bildes. Falls man die Funktionen nur mit den Texten beschriften will, die sie in der SERVUS-Datei führen, dann muß man hier mit "0" antworten, ansonsten werden die in "LAFIGU" gespeicherte Texte verwendet.

Das Untermenü 4 dient schließlich dazu,

- Beschriftungen,
- Zeichnungsart & Markierungen sowie
- Farben & Lienienstärken

der einzelnen Funktionen zu ändern.

Mit einer 0 als Eingabe kehren die Untermenü s "2", "3" und "4" in das "STATUS"-Hauptmenü zurück. Der Benutzer kann hier auch gezielt in eine der Optionen des Hauptmenüs gelangen, indem er die Nummer dieser Option mit -1 multipliziert eintippt. Beispiel : das eintippen von -5 in der TEXT-Untermenü bewirkt, daß das fertige Bild gezeigt wird, -1 führt dazu, das neue Funktionen gesucht werden.

Das Untermenü 5 zeigt das fertige Bild und speichert es, falls erwünscht. Vom Benutzer wird dabei Abbildungsnummer und Bildname erfragt. Falls im "Buch" bereits ein Bild mit diesem Namen gibt, erfolgt eine Warnung. Der Benutzer kann in dieser Untermenü ein Teil des Bildes auch vergrößert anschauen.

Die FIGUR - Bilder sind wie folgt gegliedert:

```
            FIGUR0ij.RAHMEN
          ( FIGUR0ij.KFK              )
            FIGUR0ij.INHALT.XACHS
            FIGUR0ij.INHALT.XTEXT
            FIGUR0ij.INHALT.FACHS
            FIGUR0ij.INHALT.FTEXT
            FIGUR0ij.INHALT.KURVEN
            FIGUR0ij.INHALT.KURVTX
          ( FIGUR0ij.DATE             )
            FIGUR0ij.TEXT
```

FIGUR0ij ist der, vom Programm vorgeschlagener Bildname bei einer Bildnummer ij. Anstatt FIGUR0ij kann der Benutzer einen anderen - GS-zulässigen - Namen einsetzen. Die Verteilung der Bildelemente in die obigen Teilbilder entspricht in etwa der Verteilung beim Stapel-Verfahren (s. S. S. 46). Die Achsen und ihre Beschriftungen sind beim FIGUR getrennt untergebracht. Das Teilbild DATE enthält das Herstellungsdatum des Bildes, KFK das KfK-Emblem.

10.0 Funktionen aus einer SERVUS-Datei werden als eine Funktions-Oberfläche gezeichnet.

10.1 Das Verfahren.

Das Prozedurpaar FIGUR/FIGURB kann mehrere Elemente einer Funktionenschar gleichzeitig in einer Bild-Ebene zeigen. Die Prozeduren DDFIG/DDFIGB dienen dazu, "verwandte" Funktionen als eine dreidimensionale Oberfläche abzubilden. Sie stellen zu diesem Zweck die Funktionen in parallelen Ebenen nebeneinander und zeichnen von der so entstandenen Funktions-Oberfläche - mit Hilfe der DDPLO1 Routine /7/ - ein perspektivisches Bild. Die Ebenen der einzelnen Funktionen können entweder die gleiche Abstand zueinander haben , oder der Benutzer kann die Größe dieser Abstände mit Hilfe der Kurven-Kenngröße "NUMMER" (s. S. 6) steuern. Die fertigen Bilder können genau so, wie dies bei den FIGUR-Prozeduren der Fall war, gespeichert werden (s. "Funktionen aus einer SERVUS-Datei werden als eine Kurvenschar gezeichnet.").

Als Vergleichs-Beispiel zeigen die Abbildung 10 und Abbildung 11 dieselben Funktionen, beim ersten Male mit FIGUR und beim zweiten Male mit DDFIG dargestellt.

Der Aufbau eines Oberflächen-Bildes unterscheidet sich von der Kurvenschar-Darstellung. Es gibt z.B. zwar auch hier einen inneren Rahmen, er ist aber unsichtbar und ist nur da, um das Oberflächen-Bild in die Mitte des Blattes einzuordnen. Auch die Markierung und die Beschriftung der einzelnen Funktionen entfällt bei diesen 3D-Prozeduren.

Die Oberflächen-Bilder sind dafür vielschichtiger: die Fläche selbst besteht aus der X = konst.- und aus der Z = konst.-Linien (im Folgenden ist Z immer der Scharen-Parameter der gezeigten Kurven, s. auch Abbildung 12). Dazu können noch folgende Seh-Stützen kommen (s. Abbildung 11 auf Seite 120) :

- die seitlichen Begrenzungsflächen,
- ein Richtungspfeil mit Beschriftung (17 Zeichen) je Achse,
- ein - die Oberfläche umschließender - "Koordinatenwürfel", bestehend aus ausgezogenen oder aus gestrichelten Linien,
- eine Bemaßung der Achsen dieses Würfels durch Markierungen.

10.2 Stapel-Verfahren für die Oberflächen-Darstellung.

Die Prozedur DDFIGB erzeugt Oberflächen-Bilder von der gespeicherten Funktionen im Stapelbetrieb. In einem Auftrag kann man bis zu 100 Oberflächen abbilden, die dann zu einem Gesamtbild zusammengelegt und dieses dann durch das Xynetics- oder VERSATEC-Zeichengerät ausgegeben wird. Das Gesamtbild wird auch hier als "T001" gespeichert.

Die einzelnen Teilbilder von T001 haben folgende Aufbau:

```
T001.RAHMEN0j
T001.INHALT0j.KONSX0j
T001.INHALT0j.KONSZ0j    ,    0j = 01 , ... , 99 , 00
T001.INHALT0j.LINKS
T001.INHALT0j.RECHTS
T001.INHALT0j.HOCH
T001.INHALT0j.KOORD0j
T001.TEXT0j      .
```

Das Teilbild RAHMEN0j enthält nur den äußeren Rahmen (Bild -Trennnetz), in den Teil TEXT0j kommt der Bildtitel, alles andere wird im Teil INHALT0j. gespeichert. Die X=konst. bzw. Z=konst. Linien kommen in die Teilbilder KONSX0j bzw. KONSZ0j, die Pfeile, nebst Achsen-Beschriftungen kommen in die drei Teile LINKS, RECHTS bzw. HOCH und der Koordinaten-Würfel in den Teil KOORD0j .

Der Benutzer kann auch hier das Bild nach eigenem Geschmack aufbauen - was u.U. umfangreiche Eingabe erfordert, er kann aber auch die wenig Arbeit verlangenden Standard-Darstellungen verwenden. In der Regel (Mindesteingabe) wird nur die "nackte" Oberfläche gezeigt und zwar in Parallelprojektion, mit Kipp- und Drehwinkel von 75 bzw. 300 Grad. Das Bild ist unverzerrt, d.h. alle Raumrichtungen werden gleich behandelt. Im Regelfall werden die einzelnen Teilbilder im T001 mit 1 beginnend durchnumeriert. Sie erhalten keinen Bildtitel außer der Bildnummer. Die "Abb. 1" im Bild Abbildung 12 zeigt eine Fläche mit Mindesteingabe.

Es besteht aber - wie bei FIGURB - auch die Möglichkeit, Bildnummer, Achsen-Beschriftungen und Bildtitel vorzugeben. Der Benutzer kann den für die Abbildung günstigsten Sichtpunkt aussuchen. Er kann auch die gespiegelte oder die "komplementäre" Version seiner Fläche zeichnen lassen (vgl. Abbildung 13 auf Seite 122), usw.

Die Liste der Hintergrund-Prozedur DDFIGB (Zeichengerät=VERSA-COLOR)

Name : tso000.SERVUS.CNTL(DDFIGB)

```
//inr000d  JOB (0abc,xyz,p9x9y),Benutzer,MSGLEVEL=(1,1),
//    REGION=2500K
//COLOR    EXEC F7CLG,PARM.C='LANGLVL(77)',
// PARM.L='MAP,LIST',PLOT=GS7,PLOU=X,USER='SYS2.TRACEGS7'
//C.SYSIN DD *
C       D D F I G B  ( STAPELVERSION VON DDFIG )
        CHARACTER*4 INST
        DATA IRE/15/,IDIN/5/,NETZ/2/,MEM/40/,KPRI/2/,INST/'INR '/
        CALL DDFIGB(IRE,IDIN,NETZ,MEM,KPRI,INST)
        STOP
        END
//*
//L.BIBS DD DSN=INR105.SERVUS.LOAD,DISP=SHR
//L.SYSIN DD *
 INCLUDE BIBS(SERDIO,DDFIGB,NEXT,PAPIER,WUERFL,FENST4,NARS)
 ENTRY MAIN
//*
```

```
//G.PLOTPARM DD *
 &PLOT LUNIT=26, &END
//G.FT26F001 DD SYSOUT=*
//G.COMM     DD SYSOUT=*
//G.FT15F001 DD DSN=INR105.VIELBILD.DATA,DISP=(OLD,KEEP)
//G.FT31F001 DD DSN=TSO105.MEMORY(ABB1),DISP=(OLD,KEEP)
//G.FT32F001 DD DSN=TSO105.MEMORY(ABB2),DISP=(OLD,KEEP)
//G.FT33F001 DD DSN=TSO105.MEMORY(ABB3),DISP=(OLD,KEEP)
//G.FT34F001 DD DSN=TSO105.MEMORY(ABB4),DISP=(OLD,KEEP)
//G.TRACEGS7 DD DSN=INR105.GSBOOK,DISP=(OLD,KEEP)
//G.SYSIN DD *
 &LADDFI INLIST=-6,LAUF=18,18, &END
 &LADDFI INLIST=-6,NABB=12,ABTEX(1:8)='ZWEITENS',ISCH=1,KZ=2,
 BADH=0.6,IMO=5,FMIST=-150.,FMAST=150.,FEDST=50., &END
 &LADDFI INLIST=-6,LAUF=18,18,LSX=0,LSZ=2,LST=1,LSR=1,
 ABTEX(1:8)='DRITTENS',ISCH=2, &END
 &LADDFI ABTEX(1:9)='FUENFTENS',ABSTND=0.,ITX=1,BADH=0.6,PHI=280.,
 TEXTX='KANALHOEHE ( CM )',TEXTF='GESCHW. (50 CM/S)',INLIST=-11,
 TEXTZ='ZEIT  (0.002 SEC)',THETA=70.,FRAR$=6.,
 KFX=0,KFZ=1,KFR=4,KFT=5, &END
 &LADDFI ABTEX(1:9)='SECHSTENS',ITX=33,ISCH=80,FRAR$=6., &END
 &LADDFI LAUF(1)=0, &END
//
```

Die gezeigte Eingabe entspricht dem Bild Abbildung 12.

Eingabebeschreibung:
Die Eingabe besteht aus allgemeinen Angaben zum Zeichnungs-Verfahren und aus speziellen Anweisungen für jedes einzelne Bild. Die allgemeinen Verfahrens-Hinweise sind im Aufruf-Kopf der Prozedur enthalten und man kann sie in der DATA-Anweisung ändern. Diese Steuergrößen sind:

(IRE,IDIN,NETZ,MEM,KPRI,INST).

Von diesen Größen ist nur INST ein CHARACTER*8 Wort, alle anderen sind INTEGERs. Sie haben die folgenden Bedeutungen:

IRE ist die symbolische Nummer der SERVUS-Datei,

IDIN ist die DIN-Größe des Bildes,

IDIN > 0 zeichnet das Bild in DIN-A-"IDIN"-Querformat und

IDIN < 0 in DIN-A-"-IDIN"-Hochformat.

NETZ > 0 erzeugt ein Trennungsnetz zwischen den Bildern. Falls

NETZ = 2 | -2 ist und ABS(IDIN) < 5, dann erhält jedes Teilbild das KFK-Emblem (s. z.B. Abbildung 11).

MEM wählt den Zeichengerät: für

MEM= 1 | 10 wird die Schwarzweiss-Version des Farb-VERSATECs benutzt, für

MEM= 2 | 20 der XYNETICS-Plotter für

MEM= 3 | 10 der alte VERSATEC-Plotter und für

MEM= 4 | 30 wird die Farb-Version des Farb-VERSATECs benutzt. Falls

MEM= 10 | 20 | 30 | 40 ist, dann wird das Gesamtbild T001 auch gespeichert.

KPRI ist die Ausdruck-Intensität der Begleit-Druckausgabe.

INST 8 Zeichen, die auf dem KfK-Emblem mit erscheinen (Institutskürzel, +INR + auf der Abbildung 11).

Die speziellen Bild-Informationen werden im NAMELIST-Format eingegeben, für jedes Teil-Bild eine Liste "LADDFI" :

```
NAMELIST /LADDFI/ INLIST,LAUF,XMIST,XMAST,XEDST,FMIST,FMAST,FEDST,
      XABB$,FABB$,XAUF$,FAUF$,XRAR$,FRAR$,SCR$,TAP$,TAT$,TAZ$,
      BADH,ABSTND,THETA,PHI,KZ,ISCH,IRIS,DWT,IMO,MRE,
      NABB,ITX,ABTEX,TEXTX,TEXTF,TEXTZ,LSR,LST,LSX,LSZ,
      KFR,KFT,KFX,KFZ
```

Von den Listen-Größen ist ABTEX ein CHARACTER*70 Name, TEXTX, TEXTZ und TEXTF sind CHARACTER*17 Namen, KZ,ISCH,IRIS,IMO,MRE,NABB, INLIST, LSR,LST,LSX,LSZ, KFR,KFT,KFX,KFZ und ITX sind INTEGERs und LAUF ist ein 50-er Feld von INTEGERs. Alle anderen Daten sind REAL*4 Zahlen.

Die Bedeutungen der einzelnen Listen-Größen:

INLIST , LAUF dürfen auch bei Mindesteingabe nicht fehlen. Sie bestimmen zusammen die Zahlenmenge der zu zeichnenden Funktionen.

INLIST > 0 Falls ist, besteht sie aus den Zahlen

$$\{LAUF(1), LAUF(2), \ldots, LAUF(INLIST)\}.$$

Bei dieser Eingabe kommt nach der ersten Funktion eine weitere nur dann in das angefangene Bild, falls das zugehörige LAUF(j) negativ ist. Ein Index LAUF(k) ≥ 0 beendet das Bild.

INLIST = 0 Falls ist, ist die Zahlenmenge die folgende:

$$\{LAUF(1) \leq i \leq LAUF(2)\} \text{ bzw. } \{LAUF(1) \geq i \geq LAUF(2)\}$$

INLIST < 0 und falls ist, hat man die -INLIST Ordnungszahlen:

$$\{LAUF(1), [LAUF(1)+LAUF(2)], [LAUF(1)+2*LAUF(2)], \ldots\}.$$

Bei den Eingabe-Typen INLIST = 0 und INLIST < 0 werden - im Gegensatz zu dem "INLIST > 0"-Fall - alle Funktionen, die durch die Zahlenmenge bestimmt sind, in dasselbe Bild eingezeichnet.

LAUF(1) = 0 Eine Eingabe-Liste mit LAUF(1) = 0 stoppt die Prozedur DDFIGB.

Die folgenden Daten umreißen das - für das Bild vorgesehene - Datenfenster (s. auch "Über SERVUS-Dateien." und "Eine SERVUS-Datei wird gesichtet und kopiert.") :

XMIST,XMAST bestimmen den erlaubten Werte-Bereich auf der X-Achse,
FMIST,FMAST den entsprechenden Bereich auf der F-Achse.
XEDST,FEDST sind die vorgegebenen Skalenlängen (s. S. 6).

Falls bei einer Eingabe XMIST $\geq$ XMAST ist, oder XEDST = 0, dann benutzt die Prozedur anstelle dieser Größen XMI, XMA und XED von der ersten Funktion des Bildes. Falls auch diese Größen ungeeignet sind, dann werden diese Daten anhand der Abszissenmenge von der Prozedur ermittelt. Entsprechendes gilt bei der Größen FMIST, FMAST und FEDST.

Die folgenden Daten bestimmen den inneren Rahmen und die Schrift :

SCR$ ist die Höhe der verwendeten Schriftzeichen,
XAUF$,FAUF$ sind die Koordinaten der linken, unteren Ecke des unsichtbaren, inneren Rahmens und
XRAR$,FRAR$ sind die Abstände zum Bildrand am rechten, bzw. am oberen Ende des inneren Rahmens.
XABB$,FABB$ sind die Anfangs-Koordinaten des Bildtitels.

Die Bedeutungen der obigen Größen kann man auch aus der Abbildung 9 auf Seite 118 entnehmen.

Alle hier beschriebenen Größen SCR$,XAUF$ - ... - FABB$ sind in den relativen Einheiten UNITX bzw. UNITF anzugeben : UNITX ist 1/32. der X-Ausdehnung des Bildes, UNITF ist 1/32. der F-Ausdehnung.

Eingabegrößen für die Richtungspfeile (in Schrifthöhe-Einheiten):

**TAP$** ist der Abstand des Richtungpfeiles zur jeweiligen Achse. für TAP$=0 werden die Pfeile unterdrückt.

TAT$ ist der entsprechende Abstand der Achsen-Beschriftung.

TAZ$ bestimmt die Größe der Markierungen auf dem Koordinaten-Würfel.

Die folgenden Daten beziehen sich auf die Abbildung und auf die Art der Flächen-Darstellung:

ABSTND ist der Abstand des Betrachters von der Fläche in cm,

ABSTND $\leq$ 0 bedeutet Parallelprojektion.

THETA , PHI in Grad bestimmen der Standpunkt des Betrachters:

THETA ist der Kipp-(Höhen-)-Winkel des Sichtpunktes und

PHI ist der Dreh-(Azimutal-)-Winkel.

BADH > 0 ist die Höhe/Breite-Verhältnis im Bild. Beim

BADH = 1. entsteht ein unverzerrtes Bild.

KZ = +1,-1 zeichnet nur die nackte Oberfläche, bei

KZ = +2,-2 sind auch die seitlichen Begrennzungsflächen sichtbar.

KZ > 0 zeigt nur die sichtbaren Linien der Fläche, bei

KZ < 0 werden auch die verdeckten Linien gezeigt.

IRIS steuert die Spiegelungen der Funktionsfläche :

IRIS = 0 liefert die ursprüngliche,

IRIS = 1 ergibt die komplementäre und

IRIS = 2 die gespiegelte Fläche (s. Abb 13).

ISCH steuert die zusätzliche Sichtunterstützungen: bei

ISCH = 0 wird nur die Fläche gezeichnet, bei

ISCH = 1 werden zusätzlich die X- , Z- und F-Achsen beschriftet,

ISCH = 2 werden auch die Markierungen auf die Achsen gezeichnet.

ISCH $\geq$ 3 zeigt auch den Koordinatenwürfel :

ISCH = 3 zeichnet einen Würfel mit ausgezogenen Kanten

ISCH > 3 ergibt einen gestrichelten Würfel und zwar

ISCH = j zeigt j-2 Striche je Kante.

DWT,IMO,MRE steuern die Zusammenstellung und Markierung der Funktions-Ebenen , d.h. die Gestaltung der Z-Achse.

DWT ist ein Maß für die Abstand dieser Ebenen : bei

DWT $\neq$ 0 sind alle diese Abstände gleich (DWT) und jede **IMO** -te Ebene wird markiert, beginnend mit der **MRE** -ten. Bei

DWT = 0 entsprechen die Abstände der Diiferenz der Kenngröße "NUMMER" bei den benachbarten Funktionen. In diesem Falle werden alle die Ebenen markiert , deren "NUMMER"-n um **IMO** voneinander abweichen , wobei **MRE** ist die erste "NUMMER"-differenz.

Die restlichen Daten beziehen sich auf die Beschriftungen und auf die Kurvendarstellung:

NABB ist die vorgesehene Bild-Nummer. Falls

NABB = 0 ist, werden die Bilder durchnumeriert.

ITX entscheidet über die Beschriftung :

ITX = 0 erzeugt ein Bild, bei der nur ABTEX und die Funktionsgrößen NAMX(1), MASX(1) und NAMF(1), MASF(1) zum Beschriften benutzt werden. Über der Z-Achse steht jetzt "SCHAREN-PARAMETER" (Abbildung 11, "ZWEITENS"). Bei

ITX > 0 werden die Achsen mit den Texteingaben TEXTX, TEXTZ bzw. TEXTF beschriftet. Falls hierbei

ITX = 30+j ist, mit 0 < j < 5, dann kommt der Bildtitel aus der Datei MEMORY(ABBj), ansonsten wird wieder ABTEX benutzt.
ABTEX enthält den Bildtitel (≤ 70 Zeichen),
TEXTX,TEXTZ enthalten die Pfeil-Beschriftungen für
,TEXTF die X-, Z- und F-Achse (≤ 17 Zeichen).

LSR,...,LSZ steuern die Linienstärke beim zeichnen. Der Wertbereich dieser Integers ist [0,20], größere Zahlen bedeuten einen stärkeren Strich. Es sind :

LSR Strichstärke der Rahmen-Zeichnung,
LST Strichstärke des Bildtitels und des KfK-Emblems,
LSX,LSZ Strichstäken der X=konst. resp. Z=konst. Linien.

KFR,...,KFZ bestimmen die Farbe des Gezeichnetem. für die Farbtafel s. /3/ oder S. 46. Hier sind :

KFR Farbe der Rahmen-Zeichnung,
KFT Farbe des Bildtitels und des KfK-Emblems,
KFX,KFZ Farben der X=konst. resp. Z=konst. Linien.

Zu der Strichstärke und Text-Dateien s. die Bemerkungen am Ende des Abschnittes 9.2.

10.3 Oberflächen-Darstellung im Dialog.

Die Prozedur DDFIG entspricht im wesentlichen der Stapel-Prozedur DDFIGB. Hier wird auch eine zusätzliche Datei, 'tso000.MEMORY(LADDFI)' benötigt, um die Daten des Rahmens, des Sicht-Punktes und der Rahmen-Beschriftungen speichern zu können. Diese Daten werden in MEMORY(LADDFI) in der Form einer Namelist gespeichert, die mit der ab S. 54 besprochenen Liste - LADDFI identisch ist. Auch DDFIG zeichnet immer auf DIN A4 Blätter und man kann auch hier die Dateien MEMORY(ABB1) - ABB4) benutzen, um im Laufe der Prozedur nach vorgefertigten Bildtitel zu greifen.

```
Prozedur-Aufruf : EX 'tso000.SERVUS.CLIST(DDFIG)' .
```

Die Prozedur DDFIG läuft in etwa so ab wie FIGUR. Die Kommandoprozedur DDFIG erfragt von dem Benutzer

1. seinen graphischen Arbeitsplatz,
2. den Namen der SERVUS-Datei , die die Kurven bereit hält,
3. den Namen des "Buches" zum Bildspeichern,
4. das Format des Bildes (Hoch oder Querformat),
5. die Art der Quelle (externe Datei | FORTRAN-Programm) in der die Daten des Bilder-Rahmens, des Sicht-Punktes und der Rahmen-Beschriftungen gespeichert sind,
6. den vorgesehenen Bildnamen,
7. ob es eine KfK-Emblem ins Bild kommen soll und
8. ob die Herstellungsdatum ins Bild eingetragen werden soll.

   Außerdem wird nachgefragt
9. wie die Datei - der die Bild-Angaben enthält - mit vollem Namen heißt, (in der Regel "tso000.memory(laddfi)") und
10. welche Glied-Namen die Hilfs-Dateien haben, die den Bildern Texte übergeben sollen ("tso000.MEMORY(abb1)",...,(abb4)").

Dann werden die benötigte Dateien zusammengefügt und die Führung an den FORTRAN-Prozedur DDFIG übergeben (s. auch die Bemerkung in "Kurven-Darstellung im Dialog.").

Das Hauptmenü in der FORTRAN-DDFIG sieht wie folgt aus :

```
PROGRAMMSTATUS / 0 / =:

  "< 0"   STOP :
  "  1"   NEUE FUNKTIONS-FLAECHE :
  "  2"   NEUER BILDRAHMEN :
  "  3"   TEXTDATEI & KORREKTUR :
  "  4"   BILDGESTALTUNG :
  "  5"   BILDAUSGABE :
```

Eine negativen Zahl als Antwort beendet das Programm, eine 5 zeigt das Bild.

Das Untermenü 1 (eine 1 wurde als Antwort eingetippt) hilft dem Benutzer die für das Bild benötigten Funktionen zusammenzusuchen. Die Suchprozedur läuft genau wie bei der Prozedur FIGUR ab (s. "Kurven-Darstellung im Dialog.").

Das Untermenü 2 dient auch hier zur Änderung des Bildrahmens :

```
RAHMENGESTALTUNG  / 0 / =:

  "  0"   ALLES IN ORDNUNG ,
  "  1"   NEUER INNERER RAHMEN :
  "  2"   SCHRIFTGROESSEN :
  "  3"   BILDTITEL VERSCHIEBEN :
  "  4"   RAHMENDATEN AUSDRUCKEN :
```

Das Untermenü 3 erledigt eventuelle Beschriftungs-Änderungen :

```
TEXTKORREKTUR / 0 / =:

  "  0"   ALLES IN ORDNUNG ,
  "  1"   TEXTQUELLEN :
  "  2"   TEXT BEI DER "X"-ACHSE :
  "  3"   TEXT BEI DER "Z"-ACHSE :
  "  4"   TEXT BEI DER "F"-ACHSE :
  "  5"   BILDTITEL :
  "  6"   FARBEN UND LINIENSTAERKEN DER BESCHRIFTUNGEN :
```

Das Untermenü 4 erlaubt es, das Bild der Funktionsfläche nach eige nen Vorstellungen zu gestalten :

```
BILDDARSTELLUNG / 0 / =:

  "  0"   ALLES IN ORDNUNG ,
  "  1"   KOMPLEMENTAERE/GESPIEGELTE FLAEHE :
  "  2"   SEITENFLAECHEN & SICHTBARKEIT :
  "  3"   LINIENSTAERKEN UND FARBEN DER X- UND Z-LINIEN :
  "  4"   ACHSENPFEILE :
  "  5"   SICHTHILFEN :
  "  6"   PROJEKTIONSABSTAND UND -WINKEL:
```

Speichern kann man das fertige Bild auch hier im Untermenü 5. Bei drohender Bild-Überschreibung erfolgt eine Warnung.

Wie bei der Routine FIGUR kehrt man mit einer 0 als Eingabe aus den Untermenüs "2", "3" und "4" in das "STATUS"-Hauptmenü zurück. Es besteht auch hier die Möglichkeit, mit einem negativen Index direkt in eine der Optionen des Hauptmenüs zu gelangen : z.B. das eintippen von -5 in der "BILDDARSTELLUNG"-Untermenü bewirkt, daß das fertige Bild gezeigt wird, -2 führt dazu, daß der Bildrahmen geändert wird.

Die DDFIG - Bilder sind wie die Einzelbilder der DDFIGB-Routine aufgebaut :

```
          FIGUR00j.RAHMEN
        ( FIGUR00j.KFK              )
          FIGUR00j.INHALT.KONSX
          FIGUR00j.INHALT.KONSZ
          FIGUR00j.INHALT.LINKS
          FIGUR00j.INHALT.RECHTS
          FIGUR00j.INHALT.HOCH
          FIGUR00j.INHALT.KOORD
        ( FIGUR00j.DATE             )
          FIGUR00j.TEXT                              .
```

j steht hier für die laufende Nummer des Bildes. Anstatt FIGUR00j kann der Benutzer einen anderen - GS-zulässigen - Namen einsetzen.

Wichtig: die Prozedur DDFIG braucht sehr viel Speicherraum. Schon eine Fläche von 50•50 Punkten kann 200 K Platz beanspruchen. Falls man das Bild auch noch speichert, braucht man den doppelten Raum. Beim Einstieg in die TSO-Sitzung ist also ein Space-Parameter S(2000) sehr zu empfehlen. Andernfalls bricht DDFIG beim ersten auf wendigen Bild unvermutet die Arbeit mit einer Fehlermeldung ab.

11.0 Eine Text-Datei wird zum GS-Bild umgewandelt.

11.1 Das Verfahren.

Es kommt oft vor, daß man Rechenergebnisse, die im 132-Spaltigen (Schnelldrucker-) Format vorliegen in eine Veröffentlichung einbringen will. Um die Handhabung solcher Texten zu ermöglichen dient die Prozedur TEXIMA.

TEXIMA wandelt die ihr zugeführten Texte unverzerrt in GS-Bilder um. Diese Bilder kann man anschliesend

- verkleinern,
- drehen,
- umfärben,
- zeilenwise hervorheben,

usw. Das nachbehandelte Bild des Textes wird anschließend

1. in eine GDF-Datei gebracht (mit der Prozedur IMMANI),
2. in einen Bild-Segment umgewandelt (mit der Prozedur GDDM-TEST , Option 7),
3. das Bild-Segment mit der DCF-Anweisung

```
:FIG ID=wauwau FRAME=NONE PLACE=INLINE.
.SI bild INLINE WIDTH xxMM DEPTH yyMM
:EFIG.
```

   an der gewünschten Stelle des Schriftes gelegt und
4. im Rahmen der DCF-Textverarbeitung (/1/) am IBM-3820-Drucker ausgedruckt.

Die Textbeispiele Abbildung 1 , ... , Abbildung 7 sowie die Tafel auf S. 2. sind alle in dieser Weise erzeugt.

11.2 Spielregeln der Prozedur TEXIMA :

Die Texte, die abgebildet werden sollten legt man in die Dateien MEMORY(ABB1) , ... , MEMORY(ABB4) ab (TEXIMA kann bis zu 4 Texte in einen Bild einbringen). Die Prozedur arbeitet mit Hilfe der TRACEGS-Prozedur TEXT (s. /5/). Daraus folgt :

- die Texte müssen eine feste Satzlänge von 80 Zeichen haben,
- die jeweilige erste Zeile enthält 5 Steuergrößen - 3 beliebige REAL*4 Zahlen und zwei beliebige INTEGERs,
- die Zeichen "$" , "¢" und "#" werden als Steuerzeichen benutzt, so daß - falls man eine dieser Zeichen abbilden will - sie doppelt eigeben muß, z.B. "$$" wird als "$" gezeichnet.

Die Dateien werden verzerrungsfrei abgebildet, d.h. mehrere, hintereinander folgende Leerstellen werden nicht zusammengelegt. Abweichend von der Eingabebeschreibung im TRACEGS-Handbuch dürfen Steuergrößen in jeder Text-Datei nur einmal - am Anfang - vorkommen.

TEXIMA arbeitet nur im vordergrund. Aufruf :

```
EX 'tso000.SERVUS.CLIST(TEXIMA)'.
```

Die Kommandoprozedur TEXIMA erfragt von dem Benutzer

1. seinen graphischen Arbeitsplatz,
2. den Namen des "Buches" zum Bildspeichern,
3. das Format des Bildes (Hoch oder Querformat),
4. die Art der Datei, in der einzelne Format-Angaben des Bildes gespeichert sind (externe Datei oder die FORTRAN-Prozedur),
5. den vorgesehenen Bildnamen,
6. ob es eine KfK-Emblem ins Bild kommen soll,
7. ob die Herstellungsdatum ins Bild eingetragen werden soll,
8. den vollen Namen der Datei, der die Bild-Angaben enthält (in der Regel "tso000.memory(latexi)") und
9. die Glied-Namen der Dateien, die die zu verarbeitende Texte enthalten ("tso000.MEMORY(abb1)", ... , (abb4)").

Dann übergibt sie die Führung der FORTRAN77-Prozedur TEXIMA.

Bemerkung zu der Frage 4 : diese Angaben sind in der Regel in der Datei 'tso000.MEMORY(LATEXI)' als eine Namenliste

```
NAMELIST /LATEXI/ XAUF$,FAUF$,XRAR$,FRAR$,SCR$,DSX$,
      DSF$,DXA$,DFA$,NABB,NIT,ITX,LSI,KFI,LST,KFT
```

gespeichert. Mit ein "0" als Antwort holt man diese Daten zur Bildgestaltung. Falls aber diese Datei noch nicht Beschrieben ist, kann man - mit ¬0 als Antwort - anstatt LATEXI die in der FORTRAN TEXIMA-Prozedur vorgesehene Standard-Daten benutzen.

Die Bedeutungen der einzelnen Größen in der obigen Nameliste :

XAUF$,FAUF$ sind die Koordinaten der linken-unteren Ecke des inneren Rahmens,

XRAR$,FRAR$ sind die Abstände zum Bildrand am rechten, bzw. am oberen Ende des inneren Rahmens,

SCR$(i) , i=1,...,4 sind die, im TEXTi verwendeten Schriftzeichen-Höhen,

DSX$(i), DSF$(i) sind die horizontale und vertikale Abstände zwischen den Zeichen - in Einheiten der Schrifthöhe in TEXTi .

DXA$(1),DFA$(1) sind die Anfangs-Koordinaten (linke-obere Ecke) des Textbildes,

DXA$(i),DFA$(i) , i=2,3,4 sind die Verschiebungen der Teil-Texten TEXTi zum TEXT1.

Alle hier beschriebenen Größen mit Ausnahme von DSX$, DSF$ sind in den relativen Einheiten UNITX bzw. UNITF anzugeben : UNITX ist 1/32. der X-Ausdehnung des Bildes, UNITF ist 1/32. der F-Ausdehnung.

Weitere Größen auf der Liste :

NABB ist die vorgesehene Bild-Nummer.

NIT gibt an, wieviel Teil-Texte in das Bild eigetragen werden sollten,

ITX = 30+i sind die Quellennummer der vier Textdateien MEMORY(ABB1), ... , MEMORY(ABB4).

LSI , LST steuern die Linienstärke beim Zeichnen :

LSI ist für die Zeichnung des inneren Rahmens zuständig,

LST(i) steuert die Strichstärke im i-ten Textteil.

KFI, KFT bestimmen die Farbe des Gezeichnetem :

KFI bestimmt die Farbe des Rahmens,

KFT(i) die Farbe des i-ten Texteils.

Für die Zahl-Farbe-Zuordnung s. S. 46 .

11.3 Die FORTRAN-TEXIMA-Prozedur

TEXIMA arbeitet - ebenso wie FIGUR und DDFIG - mit Entscheidungstafeln / Menüs. Die Hauptmenü dieser Prozedur sieht folgendermaßen aus :

```
PROGRAMMSTATUS / 0 / =:

  "< 0"   STOP :
  "  1"   BILDGESTALTUNG :
  "  2"   ANGABEN ZU DEM TEXT #1:
  "  3"   ANGABEN ZU DEM TEXT #2:
  "  4"   ANGABEN ZU DEM TEXT #3:
  "  5"   ANGABEN ZU DEM TEXT #4:
  "  6"   BILDAUSGABE :
```

Die Untermenü 1 , "BILDGESTALTUNG" , bestimmt die Form des Bildes:

```
BILDGGESTALTUNG  / 0 / =:

"  0"   ALLES IN ORDNUNG ,
"  1"   NEUER INNERER RAHMEN :
"  2"   FARBE UND LINIENSTAERKE DES INNEREN RAHMENS :
"  3"   ANZAHL DER TEXTDATEIEN :
"  4"   RAHMENDATEN AUSDRUCKEN :
```

Mit der Farbe Nr. 8 , "weiss" kann man den inneren Rahmen unsichtbar machen.

Die Untermenü 2 , ... , 5 erlauben es, mit den einzelnen Texten, die in der Dateien MEMORY(ABB1) , ... , MEMORY(ABB4) liegen herumzuspielen :

```
ANGABEN ZUR TEXTi / 0 / =:

"  0"   ALLES IN ORDNUNG ,
"  1"   TEXTQUELLE :
"  2"   TEXTAUFPUNKT :
"  3"   SCHRIFHOEHE :
"  4"   ABSTAENDE ZWISCHEN DEN ZEICHEN :
"  5"   FARBE UND LINIENSTAERKE DES TEXTES :
```

Mit "TEXTQUELLE" kann man die gewünschte Teil-Text ("TEXTi") identifizieren. Dabei gilt die Zuordnung :

```
tso000.MEMORY(ABB1) <===> 31
tso000.MEMORY(ABB2) <===> 32
tso000.MEMORY(ABB3) <===> 33
tso000.MEMORY(ABB4) <===> 34
```

TEXTAUFPUNKT : Falls mehrere Texte in einem Bild zusammengelegt werden dann bestimmt das erste Text mit "TEXTAUFPUNKT" den Ort (linke obere Ecke) des gesamten Textes. Bei den restlichen Teil-Texten gibt "TEXTAUFPUNKT" die Verschiebung zu den ersten Text an.

In der Untermenü "BILDAUSGABE" wird vom Benutzer Abbildungsnummer und Bildname erfragt (j , FIGUR00) und das fertige Bild gezeigt. Auf Wunsch kann man hier auch eine Teilvergrößerung des Bildes anschauen. Anschließend kann man das Bild FIGUR00j im vorliegendem GS.-Buch (s.a. "Kurven-Darstellung im Dialog.") Speichern.

Die TEXIMA-Bilder haben den folgenden Aufbau :

```
          FIGUR00j.RAHMEN
        (  FIGUR00j.KFK              )
          FIGUR00j.INHALT.INNRHM
          FIGUR00j.INHALT.TEXT1
        (  FIGUR00j.INHALT.TEXT2     )
        (  FIGUR00j.INHALT.TEXT3     )
        (  FIGUR00j.INHALT.TEXT4     )
        (  FIGUR00j.DATE             )
```

12.0 Die SERVUS-Kommandoprozeduren.

12.1 Über Dialoge im Kommando- und in FORTRAN-Mode.

Sowohl der FORTRAN- als auch der Kommando-Dialog erfragen vom Benutzer die für die jeweilige Prozedur benötigten Eingabedaten. Alle im SERVUS-System zu Wort kommenden Dialoge zeigen - bevor sie vom Benutzer neue Angaben anfordern - den zuletzt benutzten Wert der fraglichen Größe am Bildschirm in folgender Form an :

```
EINGABE / letzte Wert / =:
```

Mehrere "=:" -Zeichen bedeuten entsprechend viele Eingabe-Größen.

Leider muß man sich daran gewöhnen, daß sich die Regeln des FORTRAN-Dialogs (s. /8/) von der Regeln des Kommando-Dialogs in einigen Punkten wesentlich und verwirrend unterscheiden :

- Jedes Mal, wenn die FORTRAN-Prozedur vom Benutzer eine Eingabe erwartet, erscheint am Bildschirm ein "?". Der Kommando-Dialog hat keine besondere Zeichen zum Eingabe-Fordern.
- im FORTRAN Dialog werden alle Zeichen der Tastatur akkzeptiert. Im gegensatz dazu sind beim Kommando-Dialog die folgenden Zeichen Tabu:

+ - * / = > < | ¬ &
(, '

  Die erste 10 dieser Zeichen führen zum sofortigen Abbruch der Prozedur, die letzte 3 werden als Leerzeichen mißdeutet. Besonders unangenehm ist diese Geschichte beim Zeichen " / " das man erfahrungsgemäß bei einem FORTRAN-Dialog sehr häufig benutzt, um mehrere Eingabegrößen mit "einem Wort" zu bestätigen (s. u.).
- Beim Kommando-Dialog bedeutet eine leere Eingabe (einfaches Drücken auf die "Enter"-Taste) ein "ja", bzw. daß die fragliche Größe ihren bisherigen Wert behält. Von der FORTRAN-Prozedur wird eine leere Eingabe nicht akzeptiert (sie reagiert mit einen "?"). Falls man den bisherigen Wert der Größe beibehalten möchte, tippt man eine Komma " , " ein. Wenn man mehrere geforderte Größen unverändert stehen lassen will, kann man das mit einem "*/*" erreichen.
- Textvariablen nimmt der Kommando-Dialog ohne Hochkommata (TEXTTEXT), " ' " wird hier als Leerzeichen angesehen. Der FORTRAN -Dialog nimmt sie nur mit Hochkommata ('TE0TTEXT').
- Der FORTRAN-Dialog nimmt übel, wenn er - anstatt einer erwarteten Zahl - ein anderes Zeichen als Antwort bekommt und reagiert mit einer geharnischten Fehlernachricht. Die Kommandoprozedur nimmt sowas gelassen hin.
- Bei einer fehlerhaften Eingabe (Zeichen statt Ziffer z.B.) wiederholt der FORTRAN-Dialog die ganze Frage (wichtig bei Fragen, bei denen die Prozedur eine Antwort aus mehreren Zahlen/Wörtern erwartet). Der Kommando-Dialog fragt nie nach.

Hinweis : in der jetzt folgenden Liste der SERVUS-Kommandoprozeduren wird stets angenommen, daß der Benutzer sämtlichen vorkommenden Hilfsprozeduren in seine Datei 'tso000.SERVUS.CLIST' übernommen habe, d.h. alle diese Prozeduren werden in der Kurzform

```
EX  SERVUS(Hilfsprozedur)
```

angesprochen. Die allgemeinste Form des Aufrufs lautet dagegen :

```
EX  'tso000.SERVUS.CLIST(Hilfsprozedur)'
```

12.2 Haupt-Prozeduren

12.2.1 Die Kommandoprozedur IDA

Prozedur zum anlegen einer SERVUS-Datei.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdateien :

- keine

Die Prozedurliste:

```
PROC 0 USER(&SYSUID.) DD(TEST) VOL(bat00x) NORM(FOL8)
CONTROL MAIN MSG PROMPT
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE
WRITE PROZEDUR      IDA
WRITE EINE SERVUS-DATEI WIRD ANGELEGT | RENORMIERT
WRITE
WRITE ANGABEN ZU DER DATEI :
SET &N1 = &USER
SET &N2 = &DD
SET &N3 = &STR(DATA)
EX SERVUS(NEVE) 'LQ(1) CQ(1) SQ(1)'
IF &N1  =  &STR(#) THEN GOTO FIN
SET &USER = &N1
IF &SYSDSN('&TON') ¬ = OK THEN DO
   WRITENR PLATTE / &VOL / =: ( "ENTER" = JA )
   READ &ANS
   IF &ANS ¬ = &STR() THEN SET &VOL = &ANS
   EX SERVUS(ADA) '&TON LIN(X) WO(&VOL)'
   END
WRITE DIE DATEI IST ANGELEGT . (RE-)NORMIERUNG DER DATEI:
WRITENR NORM / &NORM / =: +
( "ENTER"(=JA) | GRA4 | GRA8 | FOL8 | PLOT )
READ &ANS
IF &ANS ¬ = &STR() THEN SET &NORM = &ANS
SET &SYSDVAL  =  &N2 &NORM &USER &VOL
ALLOC DA('&TON') F(FT15F001) SHR REU
CALL 'INR105.SERVUS.LOAD(IDAJOB)' '&SYSDVAL  '
WRITE ENDE DER PROZEDUR IDA
EXIT
```

12.2.2 Die Kommandoprozedur ENDE

Die Prozedur schreibt eine Endzeile in eine SERVUS-Datei.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdatei :

- tso000.MEMORY(DATAS)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE
WRITE PROZEDUR      ENDE
WRITE EINE SERVUS-DATEI WIRD MIT EINER ENDZEILE VERSEHEN
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(DATAS)'
SET &SYSDVAL=&KAMEL
READDVAL &N1 &N2 &PR2 &D2 &PR3 &D3 &PRR &DR
SET &KAMEL=&STR()
SET &N3 = &STR(DATA)
WRITE DATEI :
EX SERVUS(NEVE) 'LQ(1)'
IF &N1 = &STR(#) THEN GOTO FIN
SET &KAMEL=&KAMEL &PR2 &D2 &PR3 &D3 &PRR &DR
BEG: EX SERVUS(BEKI) 'MEM(DATAS)'
WRITE BEGINN DER PROZEDUR
ALLOC DA('&TON') F(FT15F001) SHR REU
CALL 'INR105.SERVUS.LOAD(ENDE)'
WRITE ENDE DER PROZEDUR "ENDE"
FIN: EXIT
```

12.2.3 Die Kommandoprozedur FUNKIN

FUNKIN hilft - im Rahmen eines Dialogs - dem Benutzer, eine Funktion zusammenzustellen und in eine SERVUS-Datei einzutgragen.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdatei :

- tso000.MEMORY(DATAS)

Die Prozedurliste:

```
PROC 0 IWR(16) IRE(15)
CONTROL MAIN MSG PROMPT END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE
WRITE PROZEDUR      FUNKIN
WRITE EINE FUNKTION WIRD IN EINE SERVUS DATEI GESCHRIEBEN .
WRITE
SET &EA = 0
EX SERVUS(BEKI) 'MEM(DATAS)'
SET &SYSDVAL = &KAMEL
READDVAL &PR &DR &PR2 &D2 &PR3 &D3 &PW &DW
SET &KAMEL = &STR()
SET &N3 = &STR(DATA)
WRITE LESEDATEI :
SET &N1 = &PR
SET &N2 = &DR
EX SERVUS(NEVE) 'LQ(1) CQ(0)'
IF &N1 = &STR(#) | &IRE = 0 THEN DO
   SET &N2 = &STR(*?#!@¬|%)
   SET &IRE = &STR(00)
   QUIT
ELSE DO
   IF &LENGTH(&STR(&IRE)) < 2 THEN SET &IRE = &STR(0&IRE)
   SET &FRE = &STR(FT)&STR(&IRE)&STR(F001)
   ALLOC DA('&TON') F(&FRE) SHR REU
   QUIT
SET &PR = &N1
SET &DR = &N2
SET &KAMEL = &KAMEL &STR(&PR2 &D2 &PR3 &D3)
WRITE SCHREIBDATEI :
IF &LENGTH(&STR(&IWR)) < 2 THEN SET &IWR = &STR(0&IWR)
SET &FWR = &STR(FT)&STR(&IWR)&STR(F001)
SET &N1 = &PW
SET &N2 = &DW
EX SERVUS(NEVE) 'LQ(1) CQ(0)'
IF &N1 = &STR(#) THEN GOTO FIN
SET &PW = &N1
SET &DW = &N2
ALLOC DA('&TON') F(&FWR) SHR REU
BEG: EX SERVUS(BEKI) 'MEM(DATAS)'
WRITE BEGINN DER PROZEDUR
SET &SYSDVAL = &STR(&IRE &IWR &PW &DW &PR &DR)
CALL 'INR105.SERVUS.LOAD(FUNKIN)' '&SYSDVAL '
WRITE ENDE DER PROZEDUR FUNKIN
FIN: EXIT
```

12.2.4 Die Kommandoprozedur KOPIER

Die Prozedur stellt die Steuerkarten eines Kopier-Auftrages - im Dialog mit dem Benutzer - zusammen .

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)

- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(FEJ)
- tso000.SERVUS.CLIST(SUSA)

Benötigte Hilfsdatei :

- tso000.MEMORY(KOPIER)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE
WRITE PROZEDUR      KOPIER
WRITE EINE SERVUS DATEI WIRD GELESEN ( UND KOPIERT ) .
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(KOPIER)'
SET &SYSDVAL=&STR(&KAMEL)
READDVAL &N1 &N2 &N3 &PROW &WRDAT &INL &KE &KL
SET &KAMEL=&STR()
WRITE LESEDATEI :
WRITENR MIT KOPF ( "ENTER"=JA ) =:
READ &ANS
IF &ANS¬ =&STR() THEN SET &URE=&STR(-11)
ELSE                SET &URE=11
EX SERVUS(NEVE)
IF &N1 = &STR(#) THEN GOTO FIN
SET &DARE=&TON
WRITE SCHREIBDATEI :
SET &N1=&PROW
SET &N2=&WRDAT
SET &N3=&STR(DATA)
EX SERVUS(NEVE) 'LQ(1)'
SET &DAWR=&TON
IF &DAWR=&STR()  THEN DO
   SET &UWR=0
   SET &PP=2
   QUIT
ELSE DO
   SET &UWR=15
   SET &PP=1
   QUIT
WRITE KURVENZAHL / &INL / =: ( < 51 )
WRITE > 0 : KURVEN EINZELN EINGEBEN ,
WRITE = 0 : KURVEN VOM ERSTEN BIS LETZTEN
WRITE < 0 : ERSTE , ERSTE + ABSTAND , ... , +
ERSTE + (ABS(&INL)-1)*ABSTAND
READ &ANS
IF &STR(&ANS)¬ =&STR() THEN SET &INL=&ANS
IF &INL < = 0 THEN DO
   WRITENR ERSTE KURVE / &KE /  =:
   READ &ANS
   IF &STR(&ANS)¬ =&STR() THEN SET &KE=&ANS
   IF &INL = 0 THEN WRITENR LETZTE KURVE / &KL / =:
   IF &INL < 0 THEN WRITENR KURVENABSTAND / &KL / =:
```

```
   READ &ANS
   IF &STR(&ANS)¬ = &STR() THEN SET &KL = &ANS
   QUIT
SET &KAMEL = &STR(&KAMEL &INL &KE &KL &DR)
EX SERVUS(BEKI) 'MEM(KOPIER)'
IF &INL > 0 THEN DO
   SET &L = 0
   SET &K = 0
   DO WHILE &K < 5
      SET &K = &K + 1
      SET &KB = &K*10
      SET &LU = &STR()
      DO WHILE &L < &KB
         IF &L > = &INL THEN GOTO EDI
         SET &L = &L + 1
         WRITENR &L-TE KURVE =: ( "ENTER" = ENDE )
         READ &NU
         IF &NU = &STR() THEN  GOTO EDI
         SET &LU = &LU&STR(,)&NU
         SET &LAUF&K = &LU
         QUIT
      QUIT
   EDI: IF &NU = &STR() THEN SET &L = &L-1
   QUIT
WRITE BEARBEITUNG DES AUFTRAGES :
EX SERVUS(FEJ)
EDIT &N3..CNTL OLD NUM
0120 //COPY  EXEC F7CLG,USER = 'INR105.SERVUS.LOAD'
0130 //C.SYSPRINT DD DUMMY
0140 //C.SYSIN DD *
0150       PROGRAM KOPIER
0160       CALL COPYDA
0170       STOP
0180       END
0190 //*
0191 //L.SYSPRINT DD DUMMY
0200 //L.SYSIN DD *
0210  INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
0220  ENTRY KOPIER
0230 //*
0300 //G.FT11F001 DD DSN = &DARE,DISP = SHR
IF &UWR ¬ = 0 THEN +
0310 //G.FT15F001 DD DSN = &DAWR,DISP = SHR
0320 //G.SYSPRINT DD SYSOUT = *
0330 //G.SYSIN DD *
0400  &&LAKOPI IRE = &URE,IWR = &UWR,MODUS = '**OLD***',KPRI = &PP,
0410  LAX = 0,XMIST = 0.,XMAST = 0.,XEDST = 0.,
0420  LAF = 0,FMIST = 0.,FMAST = 0.,FEDST = 0.,
IF &INL > 0 THEN DO
   0421  LAUF = &LAUF1,
   IF &L > 10 THEN  0422  &LAUF2,
   IF &L > 20 THEN  0423  &LAUF3,
   IF &L > 30 THEN  0424  &LAUF4,
   IF &L > 40 THEN  0425  &LAUF5,
   0429  INLIST = &L, &&END
   QUIT
IF &INL < = 0 THEN +
```

```
   0421  LAUF=&KE,&KL,INLIST=&INL, &&END
1999  &&LAKOPI IRE=0, &&END
2000 //
EX SERVUS(SUSA) '&N3.'
FIN: EXIT
```

12.2.5 Die Kommandoprozedur DATAS

Mit DATAS kann man SERVUS-Dateien im Vordergrund lesen und kopieren.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdatei :

- tso000.MEMORY(DATAS)

Die Prozedurliste:

```
PROC 0 IR1(00) IR2(00) IR3(00) IW(00)
CONTROL PROMPT MAIN MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE
WRITE PROZEDUR      DATAS
WRITE EINE SERVUS-DATEI WIRD GESICHTET ( UND KOPIERT )
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(DATAS)'
SET &SYSDVAL=&KAMEL
READDVAL &N1 &N2 &PR2 &D2 &PR3 &D3 &PRR &DR
SET &KAMEL=&STR()
SET &N3 = &STR(DATA)
WRITE LESEDATEI NR "15" :
EX SERVUS(NEVE) 'LQ(1)'
SET &DAT1=&TON
IF &DAT1 = &STR() THEN GOTO FIN
SET &IR1=&STR(15)
WRITE LESEDATEI NR "16" :
SET &N1=&PR2
SET &N2=&D2
EX SERVUS(NEVE) 'LQ(1)'
SET &DAT2=&TON
IF &DAT2 = &STR() THEN DO
   SET &KAMEL = &KAMEL &PR3 &D3
   SET &DAT#=&STR()
   GOTO WRI
   END
SET &IR2=&STR(16)
WRITE LESEDATEI NR "17" :
SET &N1=&PR3
SET &N2=&D3
EX SERVUS(NEVE) 'LQ(1)'
SET &DAT3=&TON
```

```
IF &DAT3 ¬ = &STR() THEN  SET &IR3=&STR(17)
WRI: WRITE SCHREIB-DATEI "26" :
SET &N1=&PRR
SET &N2=&DR
EX SERVUS(NEVE) 'LQ(1)'
SET &DATR=&TON
IF &DATR ¬ = &STR() THEN  SET &IW=&STR(26)
BEG: EX SERVUS(BEKI) 'MEM(DATAS)'
WRITE BEGINN DER PROZEDUR
ALLOC DA('&DAT1') F(FT15F001) SHR REU
IF &DAT2 ¬ = &STR() THEN ALLOC DA('&DAT2') F(FT16F001) SHR REU
IF &DAT3 ¬ = &STR() THEN ALLOC DA('&DAT3') F(FT17F001) SHR REU
IF &DATR ¬ = &STR() THEN ALLOC DA('&DATR') F(FT26F001) SHR REU
SET &SYSDVAL=&STR(&IR1 &IR2 &IR3 &IW)
CALL 'INR105.SERVUS.LOAD(DATAS)' '&SYSDVAL '
FIN: WRITE PROZEDUR DATAS ABGESCHLOSSEN.
EXIT
```

12.2.6 Die Kommandoprozedur LOESCH

Die Prozedur stellt die Steuerkarten eines Lösch-Auftrages - im Dialog mit dem Benutzer - zusammen . Der Benutzer erhält dabei Gelegenheit, eine Sicherheitskopie der zu bearbeitenden SERVUS-Datei im Massenspeicher abzulegen.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(FEJ)
- tso000.SERVUS.CLIST(SUSA)

Benötigte Hilfsdatei :

- tso000.MEMORY(LOESCH)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE
WRITE PROZEDUR      LOESCH
WRITE KURVEN AUS EINER SERVUS DATEI WERDEN GESTRICHEN.
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(LOESCH)'
SET &SYSDVAL=&KAMEL
READDVAL. &N1 &N2 &KE &KL &NUMX
SET &KAMEL=&STR()
SET &N3=&STR(DATA)
WRITE DATEI :
EX SERVUS(NEVE) 'LQ(1)'
IF &N1 = &STR(#) THEN GOTO FIN
WRITE KURVENZAHL =: ( < 51 )
WRITE > 0 : KURVEN EINZELN EINGEBEN ,
```

```
WRITE = 0 : KURVEN VOM ERSTEN BIS LETZTEN
READ &ANS
IF &STR(&ANS)¬ = &STR() THEN SET &INL = &ANS
IF &INL < = 0 THEN DO
   GR: WRITE ERSTE UND LETZTE KURVE / &KE , &KL / =:
   WRITE ( "ENTER" = JA | E = ERSTE | L = LETZTE KURVE )
   READ &ANS
   IF &ANS = &STR() THEN GOTO BEG
   IF &ANS = &STR(E) THEN DO
      WRITENR ERSTE KURVE / &KE / =:
      READ &ANS
      IF &ANS¬ = &STR() THEN SET &KE = &ANS
      QUIT
   IF &ANS = &STR(L) THEN DO
      WRITENR LETZTE KURVE / &KL / =:
      READ &ANS
      IF &ANS¬ = &STR() THEN SET &KL = &ANS
      QUIT
   GOTO GR
   QUIT
BEG: WRITE LETZTE KURVE , DIE BLEBEN SOLL , NUMMAX / &NUMX / =:
READ &ANS
IF &ANS¬ = &STR() THEN SET &NUMX = &ANS
SET &KAMEL = &KAMEL &KE &KL &NUMX &DR
EX SERVUS(BEKI) 'MEM(LOESCH)'
WRITE EINE KOPIE VON &TON
WRITE IM MASSENSPEICHER ABLEGEN ? ( "ENTER" = JA ) =:
READ &ANS
IF &ANS = &STR() THEN DO
   IF &SUBSTR(1:1,&TON) = &STR(I) THEN BS '&TON'
   IF &SUBSTR(1:1,&TON) = &STR(T) THEN TS '&TON'
   QUIT
IF &INL > 0 THEN SET &MAXL = &INL
IF &INL < =0 THEN SET &MAXL = &KL-&KE + 1
IF &INL < =0 THEN SET &NU = &KE-1
SET &L = 0
SET &K = 0
DO WHILE &K < 5
   SET &K = &K + 1
   SET &KB = &K*10
   SET &LU = &STR()
   DO WHILE &L < &KB
      IF &L > = &MAXL THEN GOTO EDI
      SET &L = &L + 1
      IF &INL > 0 THEN DO
         WRITENR &L-TE KURVE =: ( "ENTER" = ENDE )
         READ &NU
         IF &NU = &STR() THEN GOTO EDI
         QUIT
      ELSE SET &NU = &NU + 1
      SET &LU = &LU&STR(,)&NU
      SET &LAUF&K = &LU
      QUIT
   QUIT
EDI: IF &NU = &STR() THEN SET &L = &L-1
WRITE BEARBEITUNG DES AUFTRAGES :
EX SERVUS(FEJ)
```

```
EDIT &N3..CNTL OLD NUM
0120 //ERASE EXEC F7CLG,USER='INR105.SERVUS.LOAD'
0130 //C.SYSPRINT DD DUMMY
0140 //C.SYSIN DD *
0150       PROGRAM LOESCH
0160       CALL ERASE
0170       STOP
0180       END
0190 //*
0191 //L.SYSPRINT DD DUMMY
0200 //L.SYSIN DD *
0210  INCLUDE SYSLIB(SERDIO,COPYDA,FENST4,NEXT)
0220  ENTRY LOESCH
0230 //*
0300 //G.FT11F001 DD DSN=&TON,DISP=SHR
0310 //G.FT15F001 DD UNIT=SYSDA,DCB=DCB.VBS,SPACE=(CYL,(20,10))
0320 //G.SYSPRINT DD SYSOUT=*
0330 //G.SYSIN DD *
0400  &&LALOES IRE=11,IWR=15,KPRI=1,FAKTOR=0.0,
0401  NIX=&LAUF1,
IF &L > 10 THEN  0402  &LAUF2,
IF &L > 20 THEN  0403  &LAUF3,
IF &L > 30 THEN  0404  &LAUF4,
IF &L > 40 THEN  0405  &LAUF5,
0409  NUMMAX=&NUMX, &&END
1999  &&LALOES IRE=0, &&END
2000 //
LIST 300 2000
EX SERVUS(SUSA) '&N3.'
FIN: EXIT
EXIT
```

12.2.7 Die Kommandoprozedur DRUDA

Die Prozedur stellt die Steuerkarten eines Druck-Auftrages - im Dialog mit dem Benutzer - zusammen.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(FEJ)
- tso000.SERVUS.CLIST(SUSA)

Benötigte Hilfsdatei :

- tso000.MEMORY(DRUDA)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE
WRITE PROZEDUR      DRUDA
```

```
WRITE FUNKTIONEN EINER SERVUS DATEI WERDEN AUSGEDRUCKT .
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(DRUDA)'
SET &SYSDVAL=&STR(&KAMEL)
READDVAL &N1 &N2 &N3 &INL &KE &KL
SET &KAMEL=&STR()
WRITE DATEI :
EX SERVUS(NEVE)
IF &N1 = &STR(#) THEN GOTO FIN
WRITE KURVENZAHL / &INL / =: ( < 51 )
WRITE > 0 : KURVEN EINZELN EINGEBEN ,
WRITE = 0 : KURVEN VOM ERSTEN BIS LETZTEN
WRITE < 0 : ERSTE , ERSTE + ABSTAND , ... , +
ERSTE + (ABS(&INL)-1)*ABSTAND
READ &ANS
IF &STR(&ANS)¬=&STR() THEN SET &INL=&ANS
IF &INL <= 0 THEN DO
   WRITENR ERSTE KURVE / &KE / =:
   READ &ANS
   IF &STR(&ANS)¬=&STR() THEN SET &KE=&ANS
   IF &INL = 0 THEN WRITENR LETZTE KURVE / &KL / =:
   IF &INL < 0 THEN WRITENR KURVENABSTAND / &KL / =:
   READ &ANS
   IF &STR(&ANS)¬=&STR() THEN SET &KL=&ANS
   QUIT
SET &KAMEL=&STR(&KAMEL &INL &KE &KL )
EX SERVUS(BEKI) 'MEM(DRUDA)'
IF &INL > 0 THEN DO
   SET &L=0
   SET &K=0
   SET &B=0
   DO WHILE &K < 6
      SET &K=&K+1
      SET &LU=&STR()
      SET &KB=&K*10
      DO WHILE &L<&KB
         IF &L >= &INL THEN GOTO EDI
         SET &L=&L+1
         S1: WRITENR &L-TE KURVE =: +
         ( "ENTER"= ENDE | 0 = NEUES BLATT )
         READ &NU
         IF &STR(&NU)= &STR() THEN  GOTO EDI
         IF &STR(&NU)= &STR(0) THEN DO
            SET &B=0
            GOTO S1
            QUIT
         IF   > 0 && &SUBSTR(1,&NU) ¬= &STR(-) THEN +
         SET &NU=&STR(-&STR(&NU))
         IF &STR(&LU)=&STR()  THEN SET &LU=&STR(&NU)
         ELSE          SET &LU=&STR(&LU,&STR(&NU))
         SET &B=&B+1
         SET &LAUF&K=&STR(&LU)
         QUIT
      QUIT
   EDI: IF &STR(&NU) = &STR() THEN SET &L=&L-1
   QUIT
```

```
WRITE BEARBEITUNG DES AUFTRAGES :
EX SERVUS(FEJ)
EDIT &N3..CNTL OLD NUM
0120 //DRUCK EXEC F7CLG,USER='INR105.SERVUS.LOAD'
0130 //C.SYSPRINT DD DUMMY
0140 //C.SYSIN DD *
0150       PROGRAM DRUDA
0160       CALL PAGEFF
0170       STOP
0180       END
0190 //*
0191 //L.SYSPRINT DD DUMMY
0200 //L.SYSIN DD *
0210  INCLUDE SYSLIB(SERDIO,COPYDA,NEXT)
0220  ENTRY DRUDA
0230 //*
0300 //G.FT11F001 DD DSN=&TON,DISP=SHR
0320 //G.SYSPRINT DD SYSOUT=*
0330 //G.SYSIN DD *
0400  &&LADRUC IRE=11,KPRI=0,MODPRI='DN.EINX1',
IF &INL > 0 THEN DO
   0421  LAUF=&LAUF1,
   IF &K > 1 THEN  0422  &LAUF2,
   IF &K > 2 THEN  0423  &LAUF3,
   IF &K > 3 THEN  0424  &LAUF4,
   IF &K > 4 THEN  0425  &LAUF5,
   IF &K > 5 THEN  0425  &LAUF6,
   0429  INLIST=&L, &&END
   QUIT
IF &INL < = 0 THEN +
   0421  LAUF=&KE,&KL,INLIST=&INL, &&END
1999  &&LADRUC IRE=0, &&END
2000 //
EX SERVUS(SUSA) '&N3.'
FIN: EXIT
```

12.2.8 Die Kommandoprozedur FIGUR

FIGUR hilft - im Rahmen eines Dialogs - dem Benutzer, geeignete Funktionen einer SERVUS-Datei auszuwählen und sie als Kurvenschar abzubilden. Die SERVUS-Datei muß "GRA4" normiert sein.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)
- tso000.SERVUS.CLIST(SERVIN)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdateien :

- tso000.MEMORY(FIGUR)
- tso000.memory(lafigu)
- tso000.MEMORY(tttt)
- tso000.MEMORY(eeee)

- tso000.MEMORY(xxxx)
- tso000.MEMORY(tttt)

Die Prozedurliste:

```
PROC 0 VOL(bat00x)
CONTROL PROMPT MAIN MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON AP
WRITE
WRITE PROZEDUR      FIGUR
WRITE DARSTELLUNG VON FUNKTIONEN ALS KURVENSCHAREN.
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(FIGUR)'
SET &SYSDVAL=&KAMEL
READDVAL &AP &N1 &N2 &PRB &BH
EX SERVUS(TUKOR)
SET &KAMEL=&AP
WRITE FUNKTIONS-DATEI :
SET &N3 = &STR(DATA)
EX SERVUS(NEVE) 'LQ(1)'
IF &N1 = &STR(#) THEN GOTO FIN
SET &DAT1=&TON
WRITE "BUCH" ZUM BILDSPEICHERN :
SET &N1 = &PRB
SET &N2 = &BH
SET &N3 = &STR(#)
EX SERVUS(NEVE) 'LQ(1) CQ(1)'
SET &BUCH=&TON
EX SERVUS(BEKI) 'MEM(FIGUR)'
EX SERVUS(SERVIN) 'FIGUR'
IF &N1 = &STR(#) THEN GOTO FIN
WRITE BEGINN DER PROZEDUR
ALLOC DA('&DAT1') F(FT15F001) SHR REU
IF &BUCH = &STR() THEN GOTO PI
IF &SYSDSN('&BUCH')¬=OK THEN DO
   EX SERVUS(ADA) '&BUCH LIN(0) WO(&VOL) KA(10)'
   WRITE BUCH '&BUCH' ALLOKIERT.
   END
ALLOC DA('&BUCH') F(TRACEGS7) SHR REU
PI: SET &SYSDVAL = &STR(&AP &KAMEL &N1)
CALL 'INR105.SERVUS.LOAD(FIGUR)' '&SYSDVAL '
FIN: WRITE PROZEDUR FIGUR BEENDET.
EXIT
```

12.2.9 Die Kommandoprozedur DDFIG

DDFIG hilft - im Rahmen eines Dialogs - dem Benutzer, geeignete Funktionen einer SERVUS-Datei auszuwählen und sie als Funktionsoberfläche abzubilden. Die SERVUS-Datei muß "GRA4" normiert sein.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)

- tso000.SERVUS.CLIST(SERVIN)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdateien :

- tso000.MEMORY(FIGUR)
- tso000.memory(laddfi)
- tso000.MEMORY(tttt)
- tso000.MEMORY(eeee)
- tso000.MEMORY(xxxx)
- tso000.MEMORY(tttt)

Die Prozedurliste:

```
PROC 0 VOL(bat00x)
CONTROL PROMPT MAIN MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON AP
WRITE
WRITE PROZEDUR      DDFIG
WRITE DARSTELLUNG VON FUNKTIONEN ALS KURVENSCHAREN.
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(FIGUR)'
SET &SYSDVAL=&KAMEL
READDVAL &AP &N1 &N2 &PRB &BH
EX SERVUS(TUKOR)
SET &KAMEL=&AP
WRITE FUNKTIONS-DATEI :
SET &N3 = &STR(DATA)
EX SERVUS(NEVE) 'LQ(1)'
IF &N1 = &STR(#) THEN GOTO FIN
SET &DAT1=&TON
WRITE "BUCH" ZUM BILDSPEICHERN :
SET &N1 = &PRB
SET &N2 = &BH
SET &N3 = &STR(#)
EX SERVUS(NEVE) 'LQ(1) CQ(1)'
SET &BUCH=&TON
EX SERVUS(BEKI) 'MEM(FIGUR)'
EX SERVUS(SERVIN) 'DDFIG'
IF &N1 = &STR(#) THEN GOTO FIN
WRITE BEGINN DER PROZEDUR
ALLOC DA('&DAT1') F(FT15F001) SHR REU
IF &BUCH = &STR() THEN GOTO PI
IF &SYSDSN('&BUCH')¬=OK THEN DO
   EX SERVUS(ADA) '&BUCH LIN(0) WO(&VOL) KA(10)'
   WRITE BUCH '&BUCH' ALLOKIERT.
   END
ALLOC DA('&BUCH') F(TRACEGS7) SHR REU
PI: SET &SYSDVAL = &STR(&AP &KAMEL &N1)
CALL 'INR105.SERVUS.LOAD(DDFIG)' '&SYSDVAL '
FIN: WRITE PROZEDUR DDFIG BEENDET.
EXIT
```

12.2.10 Die Kommandoprozedur TEXIMA

TEXIMA wandelt Texte zu GS-Bilder um. Die zu verbildende Texte werden der Prozedur durch die MEMORY-Glieder tttt, eeee, ... zugeführt. Die Bilder können DIN-Hoch oder -Querformat haben.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)
- tso000.SERVUS.CLIST(SERVIN)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdateien :

- tso000.MEMORY(TEXIMA)
- tso000.memory(latexi)
- tso000.MEMORY(tttt)
- tso000.MEMORY(eeee)
- tso000.MEMORY(xxxx)
- tso000.MEMORY(tttt)

Die Prozedurliste:

```
PROC 0 VOL(bat00x)
CONTROL PROMPT MAIN MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON AP
WRITE
WRITE PROZEDUR      TEXIMA
WRITE TEXTDATEIEN WERDEN ZUR BILDER VERARBEITET.
WRITE
SET &EA=0
EX SERVUS(BEKI) 'MEM(TEXIMA)'
SET &SYSDVAL=&KAMEL
READDVAL &AP &N1 &N2
EX SERVUS(TUKOR)
SET &KAMEL=&AP
WRITE "BUCH" ZUM BILDSPEICHERN :
SET &N3 = &STR(#)
EX SERVUS(NEVE) 'LQ(1) CQ(1)'
IF &N1 = &STR(#) THEN GOTO FIN
SET &BUCH=&TON
EX SERVUS(BEKI) 'MEM(TEXIMA)'
EX SERVUS(SERVIN) 'TEXIMA'
IF &N1 = &STR(#) THEN GOTO FIN
IF &N1 < 1 THEN DO
   WRITE KEIN TEXTDATEI ALLOKIERT !
   GOTO FIN
   END
WRITE BEGINN DER PROZEDUR
IF &BUCH = &STR() THEN GOTO PI
IF &SYSDSN('&BUCH')¬ =OK THEN DO
   EX SERVUS(ADA) '&BUCH LIN(0) WO(&VOL) KA(10)'
   WRITE BUCH '&BUCH' ALLOKIERT.
   END
ALLOC DA('&BUCH') F(TRACEGS7) SHR REU
```

```
PI: SET &SYSDVAL = &STR(&AP &KAMEL &N1)
CALL 'INR105.SERVUS.LOAD(TEXIMA)' '&SYSDVAL '
FIN: WRITE PROZEDUR TEXIMA BEENDET.
EXIT
```

12.2.11 Die Kommandoprozedur SVLOAD.

SVLOAD dient dazu, aus den Glieder der SERVUS-Quelldatei Vordergrund-Prozeduren herzustellen.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(TAGOK)
- tso000.SERVUS.CLIST(GLIEDER)

Benötigte Hilfsdatei :

- keine

Name : tso000.SERVUS.CLIST(SVLOAD)

```
PROC 0 PROG(TEMP)
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE PROZEDUR      SVLOAD
WRITE ERSTELLUNG EINES LOAD - MODULS FUER DIE SERVUS-PAKET
WRITE
SET &TON = &STR(TSO105.SERVUS.FORT)
IF &SYSDSN('&TON') ¬ = OK THEN DO
   WRITE &TON &SYSDSN('&TON') !
   GOTO FIN
   QUIT
SET &M1 = &STR(IDA)
WIE: EX SERVUS(TAGOK)
IF &M1 = &STR(!) THEN GOTO FIN
SET &TON = &TON(&M1)
SET &GLIED = &M1
WRITE BEGINN DER PROZEDUR
SET &COS = &STR('SYS7.FORTLIB' 'SYS2.FORTLIB' 'SYS2.GS7' )
CONCAT ( 'INR105.SERVUS.LOAD' 'TSOSYS.TRACEGS7' &COS )
COPY &TON &PROG..FORT
FORT77 &PROG. NOGO OBJECT(&PROG..OBJ) PRINT(*) LANGLVL(77)
LINK &PROG..OBJ LET PRINT(*) LOAD('INR105.SERVUS.LOAD(&GLIED)')
WRITE MODUL 'INR105.SERVUS.LOAD(&GLIED.)' GELADEN
ERASE &PROG..FORT
ERASE &PROG..OBJ
FIN: EXIT
```

12.3 G.S.-Prozeduren

12.3.1 Die G.S.-Prozedur BRAUS

BRAUS hilft - im Rahmen eines Dialogs - dem Benutzer, Bilder eines GS-Buches auszuwählen und sie durch einen der Stapel- Zeichengeräte zeichnen zu lassen.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)
- tso000.SERVUS.CLIST(ALOGS)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(TAGOK)
- tso000.SERVUS.CLIST(GLIEDER)
- tso000.SERVUS.CLIST(FEJ)
- tso000.SERVUS.CLIST(SUSA)

Benötigte Hilfsdateien :

- tso000.MEMORY(BRAUS)
- tso000.MEMORY(BOOKS)

Die Prozedurliste:

```
PROC 0 NB(0) IZ1(0) IZ2(0) IZ3(0) IZ4(0)
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 BIN TON AP
WRITE
WRITE GS-PROZEDUR BRAUS
WRITE BILDER , DIE IN EINEM GS-BUCH GESPEICHERT SIND
WRITE WERDEN AN EINEM STAPEL-PLOTTER GEZEICHNET.
SET &EA=0
EX SERVUS(ALOGS)
SET &GOBLIN=&KAMEL
SET &SYSDVAL=&KAMEL
READDVAL &SIZE &GSB &UFB &SF &ZUB
IF &SIZE=&STR(0) THEN EXIT
SET &EA=0
EX SERVUS(BEKI) 'MEM(BRAUS)'
SET &L = &LENGTH(&GSB)
SET &PRJ = &SUBSTR(1:6,&GSB)
SET &BCH = &SUBSTR(8:&L,&GSB)
SET &SYSDVAL=&KAMEL
READDVAL &GER &BIN &QU &DIN &FORM
WRITE ZEICHENGERAET / &GER / =:
WRITE   "VER" :  VERSATEC
WRITE   "XYN" :  XYNETICS
WRITE   "VCP" :  VERSACOLOR  FARBIG
WRITE   "VMP" :  VERSACOLOR  SCHWARZ-WEISS
READ &ANS
IF &ANS ¬= &STR() THEN  SET &GER = &SUBSTR(1:3,&ANS)
IF &GER = &STR(VER) THEN DO
   SET &KG=1
   SET &QU=&STR(V)
```

```
   QUIT
IF &GER = &STR(XYN) THEN DO
   SET &KG=2
   WRITE AUSFUEHRUNG / &QU / =: ( "ENTER"=JA | X=KULI | Y=TUSCHE )
   READ &ANS
   IF &ANS ¬ = &STR() THEN SET &QU = &ANS
   QUIT
IF &GER = &STR(VCP) THEN SET &KG=3
IF &GER = &STR(VMP) THEN SET &KG=4
IF &GER ¬ = &STR(XYN) THEN SET &ESP=&STR(GS7,PLOU=&QU)
ELSE                  SET &ESP=&STR(GS7)
SET &IR = 2
WRITE BILDNAME IN DIE ZEICHNUNG EINTRAGEN ? ( "ENTER"=JA )
READ &ANS
IF &ANS ¬ = &STR() THEN SET &IR = 1
WRITE BEGINN DER PROZEDUR
SET &TN1=&STR()
SET &TN2=&STR()
SET &TN3=&STR()
SET &TN4=&STR()
SET &TN5=&STR()
SET &TDI=&STR()
SET &TFO=&STR()
SET &TPX=&STR()
SET &TPY=&STR()
BI: WRITE NAME DES BILDES :
SET &TON=&GSB
SET &LAB=&BIN
EX SERVUS(TAGOK)
IF &BIN=&STR(!) THEN DO
   SET &BIN=&LAB
   GOTO FIA
   QUIT
BF: WRITENR BILDFENSTER DIN-A / &DIN / =:
   READ &ANS
   IF &ANS¬ =&STR() THEN DO
   SET &DIN=&ANS
   IF &DATATYPE(&DIN)¬ =NUM | &DIN<1 | &DIN>6  THEN GOTO BF
   QUIT
ST: WRITENR STELLUNG / &FORM / =: ( H=HOCH | Q=QUER )
READ &ANS
IF &ANS¬ =&STR() THEN SET FORM=&SUBSTR(1,&ANS)
IF &FORM¬ =H && &FORM¬ = Q THEN GOTO ST
IF &FORM=H THEN SET QUER=0
    ELSE    SET QUER=1
WRITE BILDKONTROLLE AM SCHIRM
INITGS -&SIZE
OW &AP
RPIB &BIN B1
SET &EFE=&STR(A4)&FORM
DWI 0 0 &EFE
ROUT
ENDGS
WRITENR AUSGEBEN ? ( "ENTER"=JA | ALLES ANDERE=NEIN ) =:
READ &ANS
IF &ANS¬ =&STR() THEN GOTO BI
SET NB=&NB+1
```

```
   SET &IS1=&IZ1
   SET &IS2=&IZ2
   SET &IS3=&IZ3
   SET &IS4=&IZ4
   SET &I = &IZ4
   IF &I > &IZ3  THEN SET &I=&IZ3
   IF &I > &IZ2  THEN SET &I=&IZ2
   IF &I > &IZ1  THEN SET &I=&IZ1
   SET &I = &I+1
L1: IF &DIN=6 | &DIN=5 THEN SET &MU=0
   IF &DIN=4 | &DIN=3 THEN SET &MU=1
   IF &DIN<=2        THEN SET &MU=3
   SET &IZM=50
   IF &GER=&STR(XYN) THEN SET &IZM=7
L2: SET &NU=&I+&MU
   IF &GER=&STR(XYN) && &NU > &IZM  THEN  GOTO VL
   IF &IZ1>=&I    THEN GOTO L5
   IF &DIN=6      THEN GOTO L4
   IF &IZ2>=&I    THEN GOTO L5
   IF &DIN>3      THEN GOTO L3
   IF &IZ3>=&I | &IZ4>=&I THEN GOTO L10
   SET &IZ4=&NU
   SET &IZ3=&NU
L3: SET &IZ2=&NU
L4: SET &J=1
   SET &IZ1=&NU
   GOTO S1
L5: IF &DIN<4      THEN GOTO L10
   IF &IZ2>=&I    THEN GOTO L7
   IF &DIN=6      THEN GOTO L6
   IF &IZ3>=&I    THEN GOTO L7
   SET &IZ3=&NU
L6: SET &J=2
   SET &IZ2=&NU
   GOTO S1
L7: IF &IZ3>=&I    THEN GOTO L9
   IF &DIN=6      THEN GOTO L8
   IF &IZ4>=&I    THEN GOTO L9
   SET &IZ4=&NU
L8: SET &J=3
   SET &IZ3=&NU
   GOTO S1
L9: IF &DIN<6 | &IZ4>=&I    THEN GOTO L10
   SET &J=4
   SET &IZ4=&NU
   GOTO S1
L10: SET &I=&I+1
   GOTO L2
VL: WRITE NICHT GENUG PLATZ !
   SET &NB=&NB-1
   SET &IZ1=&IS1
   SET &IZ2=&IS2
   SET &IZ3=&IS3
   SET &IZ4=&IS4
   SET &I=-1
S1: IF &GER¬= &STR(XYN) THEN GOTO S3
   IF &I < 0 THEN GOTO BI
```

```
   WRITE Z1/Z1M = &IZ1/&IZM , Z2/Z2M = &IZ2/&IZM ,
   WRITE Z3/Z3M = &IZ3/&IZM , Z4/Z4M = &IZ4/&IZM .
S3: WRITE  &NB.. BILD = &BIN VERLEGT
   IF &NB > 5  THEN GOTO S4
SET &TN1=&TN1&STR(')&BIN&STR(',)
   GOTO S9
S4: IF &NB > 10 THEN GOTO S5
SET &TN2=&TN2&STR(')&BIN&STR(',)
   GOTO S9
S5: IF &NB > 15 THEN GOTO S6
SET &TN3=&TN3&STR(')&BIN&STR(',)
   GOTO S9
S6: IF &NB > 20 THEN GOTO S7
SET &TN4=&TN4&STR(')&BIN&STR(',)
   GOTO S9
S7: SET &TN5=&TN5&STR(')&BIN&STR(',)
S9: SET &TFO=&TFO&QUER&STR(,)
SET &TDI=&TDI&DIN&STR(,)
SET &TPX=&TPX&I&STR(,)
SET &TPY=&TPY&J&STR(,)
IF &DIN=1 THEN GOTO FIA
IF &IZ1 > = &IZM && &IZ2 > = &IZM && +
&IZ3 > = &IZM && &IZ4 > = &IZM THEN DO
      WRITE DAS BLATT IST VOLL !
      GOTO FIA
      QUIT
IF &NB<24 THEN GOTO BI
WRITE DIE HOECHSTMOEGLICHE BILDZAHL (24) IST ERREICHT !
FIA: SET &KAMEL = &GOBLIN
EX SERVUS(ALOGS)
SET &KAMEL = &GER &BIN &QU &DIN &FORM
EX SERVUS(BEKI) 'MEM(BRAUS)'
IF &NB=0 THEN GOTO FIN
WRITE AUFTRAGSBEARBEITUNG :
EX SERVUS(FEJ) 'R(4000)'
EDIT &N3..CNTL OLD NUM
 120 //PICTURE  EXEC F7CLG,PARM.C='LANGLVL(77)',
 130 // PARM.L='MAP,LIST',PLOT=&ESP
 140 //C.SYSIN DD *
 190 C     HAUPTPROGRAMM ZUM AUSGABE VON BIS ZU 24 GESPEICHERTEN
 200 C     GS-BILDERN AUF EINEM DER ZEICHENGERAETE IN
 210 C     DIN-A-FORMAT.     STAND JUNI ,1988 .
 220       DIMENSION X(2),Y(2),U(2),V(2),ZHF(2),ZPF(2),FF(2),FT(2)
 230       =,FY(2),FG(4),GG(4),DAL(7),INFO(8),BIN(24),NID(24)
 240       =,KBF(24),KPX(24),KPY(24)
 250       CHARACTER*8 INFO,BCH,BIN,FG,FF,FT, GG*4,PRJ*6,BIM*9
 260       =,TXT*26,FTA*25,FY*16,FYE*45,FFA*22,FFE*38
 270       COMMON /LGS/ IRET
 280       DATA ZUF/.103585/, ZPF/2.5,-1./, ZHF/2.5E-2,9.5E-3/
 290       =,ZOLL/2.54/, DAL/33.1062,23.4096,16.5531
 300       =,11.7048,8.27654,5.8524,4.13827/
 310       DATA FG/'VERSATEC','XYNETICS','VERS.COL','VERS.UNI'/
 320       =,GG/'VER_','XYN_','VCP_','VMP_'/
 330       =,FTA/'(7H FORMAT,13X,7H= DIN A,'/
 340       =,FT/'I2,2H Q)','I2,2H H)'/
 350       =,FY/'(8H X0 , Y0,12X,','(8H DX , DY,12X,'/
 360       =,FYE/'3H= (,F8.3,2H ,,F8.3,2H ),5X,11H( CM , CM ) )'/,FFA
```

```
370      =/'(18H0* * * FEHLER BEI '/, FF/',5H RPIB',',5H OUT '/
380      =,FFE/',7H . BILD,A9,9H . IRET =,Z8,6H * * *)'/
390      DATA PRJ/'&PRJ'/,BCH/'&BCH'/,KG/&KG/,NB/&NB/,+
(BIN(I),I=1,&NB)
400      =/&SUBSTR(1:&LENGTH(&TN1),&TN1)
IF &TN2¬ =&STR() THEN +
410      =,&SUBSTR(1:&LENGTH(&TN2),&TN2)
IF &TN3¬ =&STR() THEN +
420      =,&SUBSTR(1:&LENGTH(&TN3),&TN3)
IF &TN4¬ =&STR() THEN +
430      =,&SUBSTR(1:&LENGTH(&TN4),&TN4)
IF &TN5¬ =&STR() THEN +
440      =,&SUBSTR(1:&LENGTH(&TN5),&TN5)
450      =/,IR/&IR/
460      =,(NID(I),I=1,&NB)/&SUBSTR(1:&LENGTH(&TDI),&TDI)/
470      =,(KBF(I),I=1,&NB)/&SUBSTR(1:&LENGTH(&TFO),&TFO)/
480      =,(KPX(I),I=1,&NB)/&SUBSTR(1:&LENGTH(&TPX),&TPX)/
490      =,(KPY(I),I=1,&NB)/&SUBSTR(1:&LENGTH(&TPY),&TPY)/
500      CALL JOBINF(INFO)
510      INFO(3)(1:1) = ' '
520      INFO(4)(4:8) = '    '
530      INFO(6)(1:2) = ' '
540      WRITE(6,'(10H DATUM :  ,3A8,2X,2A8,11H , '//FG(KG)
550      =         //',9H-PLOTTER.)')
560      =         INFO(3),INFO(4),INFO(6),INFO(7),INFO(8)
570      WRITE(6,'(23H BILDER AUS DEM BUCH  +,A8,1H.,A8,1H+ )')
580      =          PRJ,BCH
590      CALL INITFG(IRET,-&SIZE)
600      CALL OW(GG(KG))
610      DO 111 I=1,24
620      BIM='________'
630      DO 113 J=1,8
640      IF(BIN(I)(J:J) .EQ. ' ')                GO TO 114
650      BIM(J:J)=BIN(I)(J:J)
660  113 CONTINUE
670  114 CALL RPIB(BIM,'BUCH_')
680      IF(IRET .EQ. 0)        GO TO 120
690      WRITE(6,FFA//FF(1)//FFE)  BIN(I),IRET
700         GO TO 110
710  120 X(1)=0.787402
720      Y(1)=0.787402
730      IF(NID(I) .GT. 1)      GO TO 130
740      IF(KG .EQ. 2)          THEN
750         DX=DAL(1)
760         DY=DAL(2)
770         IF(KBF(I) .EQ. 1)  WINK=0.0
780         IF(KBF(I) .NE. 1)  WINK=90.
790         GO TO 140
800      ELSE
810         NID(I)=2
820         GO TO 130
830      ENDIF
840  130 X(1)=(KPX(I)-1)*DAL(6)*(1.+ZUF)+X(1)
850      Y(1)=(KPY(I)-1)*DAL(7)*(1.+ZUF)+Y(1)
860      IF(MOD(NID(I),2) .NE. 0) THEN
870         DX=DAL(NID(I)+1)
880         DY=DAL(NID(I))
```

```
      IF(KBF(I) .NE. 1)  WINK=0.0
      IF(KBF(I) .EQ. 1)  WINK=90.
   ELSE
      DX=DAL(NID(I))
      DY=DAL(NID(I)+1)
      IF(KBF(I) .EQ. 1)  WINK=0.0
      IF(KBF(I) .NE. 1)  WINK=90.
   ENDIF
140 WRITE(6,'(9H0BILDNAME,11X,1H=,A9,I29)') BIN(I),I
   IF(KBF(I) .EQ. 1)     WRITE(6,FTA//FT(1)) NID(I)
   IF(KBF(I) .NE. 1)     WRITE(6,FTA//FT(2)) NID(I)
   WRITE(6,FY(1)//FYE) ZOLL*X(1),ZOLL*Y(1)
   WRITE(6,FY(2)//FYE) ZOLL*DX,ZOLL*DY
   DU=DX*ZUF
   DV=DY*ZUF
   X(2)=X(1)+DX
   Y(2)=Y(1)+DY
   IF(WINK .NE. 0.)      THEN
      WRITE(6,'(35H DAS BILD WURDE UM  90 GRAD GEDREHT)')
      CALL ROT(0.5*DX,0.5*DX,WINK)
      AH=DX*0.025
      ZH=DX*ZHF(IR)
      XT=X(2)+ZPF(IR)*ZH
      YT=Y(2)-21.6*ZH
      CALL DWI(X(1),Y(1),X(2),Y(2))
   ELSE
      CALL SHI(0.0,0.0,0.0,DV)
      AH=DY*0.025
      ZH=DY*ZHF(IR)
      YT=Y(1)+DV-ZPF(IR)*ZH
      XT=X(2)-21.6*ZH
      CALL DWI(X(1),Y(1)+DV,X(2),Y(2)+DV)
   ENDIF
   CALL OUT
   IF(IRET .NE. 0)   WRITE(6,FFA//FF(2)//FFE) BIN(I),IRET
   CALL DPI('*_')
   CALL PI('LABEL_')
   TXT=PRJ//'.'//BCH//'('//BIN(I)//')_'
   U(1)=X(1)+AH
   V(1)=Y(1)
   U(2)=X(2)+DU
   V(2)=Y(2)+DV
   X(2)=U(2)
   Y(2)=V(2)-AH
   CALL SCO
   CALL VEC(X,Y,U,V,2)
   CALL STR0(XT,YT,TXT,ZH,WINK)
   CALL ECO
   CALL DWI(X(1),Y(1),U(2),V(2))
   CALL OUT
   IF(IRET .NE. 0)  WRITE(6,FFA//FF(2)//FFE) 'LABEL',IRET
110 IF(NB .EQ. I)        GO TO 900
111 CALL DPI('LABEL_')
900 CALL ENDGS(MAXS)
   WRITE(6,'(27H0GROESSTER SPEICHERBEDARF =,I4
   =        ,8H K BYTES)') MAXS
   STOP
```

```
1460      END
1490 //L.SYSPRINT DD SYSOUT=*
1500 //**STEPLIB DD DISP=SHR,DSN=SYS7.VSLOAD
IF &GER ¬= &STR(XYN) THEN DO
   SET &IZM=&IZ1
   IF &IZ2 > &IZM THEN  SET &IZM=&IZ2
   IF &IZ3 > &IZM THEN  SET &IZM=&IZ3
   IF &IZ4 > &IZM THEN  SET &IZM=&IZ4
   SET &XMX = 7*&IZM
   IF &XMX<50  THEN  SET &XMX=50
   QUIT
IF &GER = &STR(XYN) THEN DO
   SET &IDENT=&QU&SUBSTR(4:6,&PRJ)&N2
   1550 //G.PLOTTAPE DD UNIT=T0800,LABEL=(,NL),DCB=DEN=2,+
   VOL=SER=&IDENT
   QUIT
ELSE DO
   1510 //**VRFOUT  DD DUMMY
   1520 //G.PLOTPARM DD *
   1530  &&PLOT LUNIT=26, &&END
   1540 //G.FT26F001 DD SYSOUT=*
   QUIT
   1600 //G.BUCH DD DSN=&GSB,DISP=SHR
   IF &UFB ¬= &STR(#) THEN +
   1610 //G.UFOLIB DD DSN=&UFB,DISP=SHR
   1620 //G.COMM DD SYSOUT=*
   2000 //
LIST 390 490
LIST 1520 2000
EX SERVUS(SUSA) '&N3.'
FIN: EXIT
```

12.3.2 Die G.S.-Prozedur IMMANI

IMMANI verwaltet GS-Bücher: Bilder werden unter die Lupe genommen , umgefärbt, verdreht, in eine GDF-Datei geschickt, gelöscht oder umbenannt.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)
- tso000.SERVUS.CLIST(ALOGS)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(TAGOK)
- tso000.SERVUS.CLIST(GLIEDER)

Benötigte Hilfsdateien :

- tso000.MEMORY(IMMANI)
- tso000.MEMORY(BOOKS)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG END(QUIT)
```

```
GLOBAL KAMEL EA N1 N2 N3 BIN TON AP
WRITE
WRITE GS-PROZEDUR IMMANI
WRITE VERWALTUNG EINES  GS-BUCHES :
WRITE #   BILDER UMBENENNEN ,
WRITE #   BILDER VOM ZOLL*ZOLL AUF CM*CM UMWANDELN ,
WRITE #   BILDER IN EINE GDF-DATEI ABLEGEN ,
WRITE #   NICHT MEHR BENOETIGTE BILDER LOESCHEN.
SET &EA=0
EX SERVUS(ALOGS)
SET &SYSDVAL=&KAMEL
SET &GOBLIN=&KAMEL
READDVAL &SIZE &GSB &UFB &SF &ZUB
IF &SIZE=&STR(0) THEN EXIT
SET &EA=0
EX SERVUS(BEKI) 'MEM(IMMANI)'
SET &SYSDVAL=&KAMEL
READDVAL &BIN &FORM &XLU &YLU &XRO &YRO
WRITE BEGINN DER PROZEDUR
BI: WRITE NAME DES BILDES :
SET &TON=&GSB
SET &LAB=&BIN
EX SERVUS(TAGOK)
IF &BIN=&STR(!) THEN DO
   SET &BIN=&LAB
   GOTO FIN
   QUIT
SET &V = &STR()
INITGS -&SIZE
OW &AP
RPIB &BIN B1
MP: LA
WRITE
WRITE VORGEHEN =: ! = NAECHSTES BILD
WRITE            G = GDF-ABLAGE
WRITE            N = UMBENENNEN
WRITE            W = FENSTER
WRITE            # = LOESCHEN
WRITE            C = GRUEN UMFAERBEN
WRITE            R = DREHEN
WRITE            S = ZOLL => CM UMRECHNEN
WRITE            Z = LUPE
READ &V
IF &V=&STR(!) THEN DO
   ENDGS
   GOTO BI
   QUIT
IF &V=&STR(C) | &V=&STR(R) | &V=&STR(Z) | &V=&STR(S) | +
&V=&STR(N) THEN GOTO CH
IF &V=&STR(#) THEN DO
   DPIB &BIN B1
   WRITE DAS BILD &BIN WURDE AUS DEM BUCH &GSB AUSRADIERT.
   QUIT
IF &V=&STR(G) THEN DO
   CW &AP
   OW GDF
   ROUT
```

```
   CW GDF
   OW &AP
   WRITE DAS BILD &BIN WURDE INS BUCH &ZUB. EIGETRAGEN.
   QUIT
IF &V=&STR(W) THEN DO
   WRITE FENSTERWAHL / &FORM / =:
   WRITENR ( "H" = DIN-H | "Q" = DIN-Q | SONST MAXIMAL )
   READ &ANS
   IF &ANS¬ =&STR() THEN SET &FORM=&ANS
   IF &FORM¬ =&STR(H) && &FORM¬ =&STR(Q) THEN GOTO IK
   SET &EFE=&STR(A4)&FORM
   DWI &XLU &YLU &EFE
   IK: ROUT
   QUIT
GOTO MP
CH: IF &V=&STR(C) THEN SAT C2
IF &V=&STR(R) THEN DO
   WRITENR DREHWINKEL ( IN GRAD ) =:
   READ &ANS
   SET &WI = &STR(&ANS)
   ROT 0 0 &WI
   QUIT
IF &V=&STR(N) THEN DO
   SET &LB=&LENGTH(&BIN)
   ER: WRITENR NEUE NAME / &BIN / =: ( "ENTER" = KEINE NEUE NAME )
   READ &ANS
   IF &ANS=&STR() THEN GOTO MP
   IF &LENGTH(&ANS) > &LB THEN DO
      WRITE NAMENSLAENGE > &LB ZEICHEN !
      GOTO ER
      QUIT
   SET &NEU=&ANS
   WRITENR NEUE NAME &NEU IN ORDNUNG ? ( "ENTER" = JA )
   READ &ANS
   IF &ANS¬ =&STR() THEN GOTO ER
   REN &BIN &NEU
   SET &BIN=&NEU
   QUIT
IF &V = &STR(S) THEN DO
   PWI
   SCA 0. 0. 1. 2.54 1. 2.54
   ROUT
   QUIT
IF &V=&STR(Z) THEN DO
   LU: WRITENR ECKE LINKSUNTEN : X / &XLU / =: , Y / &YLU / =:
   READ &AX &AY
   IF &STR(&AX) ¬ = &STR() THEN SET &XLU = &STR(&AX)
   IF &STR(&AY) ¬ = &STR() THEN SET &YLU = &STR(&AY)
   WRITE ECKE LINKSUNTEN = ( &XLU , &YLU )
   WRITENR ECKE RECHTSOBEN : X / &XRO / =: , Y / &YRO / =:
   READ &AX &AY
   IF &STR(&AX) ¬ = &STR() THEN SET &XRO = &STR(&AX)
   IF &STR(&AY) ¬ = &STR() THEN SET &YRO = &STR(&AY)
   WRITE ECKE RECHTSOBEN = ( &XRO , &YRO )
   API &BIN
   PWI &XLU &YLU &XRO &YRO
   QUIT
```

```
WRITE GEAENDERTES &BIN ABSPEICHERN ? ( # = JA ) =:
READ &ANS
IF &ANS = &STR(#) THEN DO
   SPIB &BIN B1
   WRITE DAS BILD &BIN WURDE IN &GSB. GESPEICHERT
   QUIT
GOTO MP
FIN: SET &KAMEL = &BIN &FORM &XLU &YLU &XRO &YRO
EX SERVUS(BEKI) 'MEM(IMMANI)'
SET &KAMEL = &GOBLIN
EX SERVUS(ALOGS)
EXIT
```

12.3.3 Die G.S.-Prozedur PICOR

PICOR überträgt die Strukturliste des GS-Bildes BILD in die gegliederte Datei 'inr000.PICOR(BILD)'.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(TAGOK)
- tso000.SERVUS.CLIST(GLIEDER)

Benötigte Hilfsdatei :

- tso000.MEMORY(PICOR)

Die Prozedurliste:

```
PROC 0 SIZE(999) VOL(bat00x)
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 BIN TON
SET &SED = &SYSUID..PICOR
IF &SYSDSN('&SED') ¬ = OK THEN +
EX SERVUS(ADA) '&SED LIN(80) KA(10) WO(&VOL)'
WRITE
WRITE GS-PROZEDUR PICOR
WRITE STRUKTUR EINES GS-BILDES WIRD IN DIE DATEI '&SED.' +
GESPEICHERT .
SET &EA = 0
EX SERVUS(BEKI) 'MEM(PICOR)'
SET &SYSDVAL = &KAMEL
READDVAL &N1 &N2 &BIN
SET &KAMEL = &STR()
WRITE GS-BUCH:
SET &N3 = &STR(#)
EX SERVUS(NEVE) 'LQ(1)'
IF &N1 = &STR(#) THEN GOTO FIE
WRITE BEGINN DER PROZEDUR
ALLOC FI(B1) DS('&TON') SHR REU
BI: WRITE NAME DES BILDES :
SET &LAB = &BIN
EX SERVUS(TAGOK)
```

```
IF &BIN=&STR(!) THEN DO
   SET &BIN=&LAB
   GOTO FIN
   QUIT
WRITENR BILD &BIN O.K. =: ( "ENTER" : JA )
READ &AN
IF &AN ¬ = &STR() THEN GOTO BI
SET &SEM=&SED(&BIN.)
ALLOC FILE(INTEX) DA('&SEM') SHR REU
SET &SYSOUTTRAP = 500
INITGS -&SIZE
RPIB &BIN B1
LS
OPENFILE INTEX  OUTPUT
SET I=2
ERROR
DO WHILE &I < = &SYSOUTLINE
   SET LIN = &STR(&&SYSOUTLINE&I)
   SET INTEX = &STR(&LIN)
   PUTFILE INTEX
   SET &I = &I+1
   QUIT
CLOSFILE INTEX
ENDGS
SET &SYSOUTTRAP = 0
SET &I=&I-6
WRITE DIE STRUKTUR DES BILDES &BIN ( &I. ZEILEN ) WURDE
WRITE IN DER DATEI '&SEM' GESPEICHERT.
GOTO BI
FIN: SET &KAMEL = &KAMEL &BIN
EX SERVUS(BEKI) 'MEM(PICOR)'
FREE FI(B1 INTEX)
WRITENR COMPRESS '&SED' ? =: ( "ENTER" = JA )
READ &ANS
IF &ANS=&STR() THEN DO
  COMPRESS '&SED'  NOLIST
  REDUCE  '&SED'
  QUIT
FIE: EXIT
```

12.3.4 Die G.S.-Prozedur PIXI

PIXI vereinfacht Eröffnung und Schließung einer GS-Sitzung.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)
- tso000.SERVUS.CLIST(ALOGS)
- tso000.SERVUS.CLIST(NEVE)

Benötigte Hilfsdatei :

- tso000.MEMORY(BOOKS)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON AP
WRITE GS-PROZEDUR      PIXI
WRITE EINSTIEG ZUM ZEICHNEN MIT DEM GS .
SET &EA=0
EX SERVUS(ALOGS)
SET &SYSDVAL=&KAMEL
READDVAL &SIZE
IF &SIZE=&STR(0) THEN GOTO FIN
WRITE BEGINN DER PROZEDUR
WRITE                                  ****
WRITE DIE PROZEDUR KANN MAN DURCH  EINTIPPEN VON   STOP   BEENDEN.
WRITE                                  ****
INITGS -&SIZE
OW &AP
DWI MAX
TERMIN STOP
ENDGS
EX SERVUS(ALOGS)
FIN: EXIT
```

12.3.5 Die G.S.-Prozedur PODEL

PODEL unterstützt den Benutzer beim ausradieren einzelner Elemente aus einem komplexen Objekts eines GS-Bildes.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)
- tso000.SERVUS.CLIST(ALOGS)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(TAGOK)
- tso000.SERVUS.CLIST(GLIEDER)
- tso000.SERVUS.CLIST(OBJEKT)

Benötigte Hilfsdateien :

- tso000.MEMORY(PONUM)
- tso000.MEMORY(BOOKS)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 BIN TON AP
WRITE
WRITE GS-PROZEDUR PODEL
WRITE PRIMITIVE OBJEKTE EINES COMPLEXEN OBJEKTS
WRITE WERDEN AUSRADIERT
SET &EA=0
EX SERVUS(ALOGS)
```

```
SET &GOBLIN = &KAMEL
SET &SYSDVAL = &KAMEL
READDVAL &SIZE &GSB &UFB &SF &ZUB
IF &SIZE = &STR(0) THEN EXIT
SET &EA = 0
EX SERVUS(BEKI) 'MEM(PONUM)'
SET &SYSDVAL = &KAMEL
READDVAL &BIN &PAN &JI &JF &FORM &MG &MH &XLU &YLU &XRO &YRO
SET &N3 = &PAN
SET &TON = &GSB
WRITE BEGINN DER PROZEDUR
SET &KAMEL = &FORM &MG &MH &XLU &YLU &XRO &YRO
NBO: EX SERVUS(OBJEKT)
SET &PAN = &N3
SET &SYSDVAL = &KAMEL
READDVAL &FORM &MG &MH &XLU &YLU &XRO &YRO
SET &EFE = &STR(A4)&FORM
INITGS -&SIZE
OW &AP
DWI 0 0 &EFE
RPIB &BIN B1
PWI &XLU &YLU &XRO &YRO
IF &PAN ¬ = &STR(#) THEN APA &PAN
APO
SAT C&MG
IR: SET &V = &STR()
WRITE
WRITE VORGEHEN =:    W : BILD ZEIGEN
WRITE                ! : ENDE
WRITE                O : OBJEKT / FENSTER AENDERN
WRITE                R : UNKONTROLLIERT RADIEREN
WRITE           SONST : KONTROLLIERT RADIEREN
READ  &V
IF &V = &STR(!) THEN GOTO FGS
IF &V = &STR(W) THEN GOTO AHA
IF &V = &STR(O) THEN GOTO FGS
EL: WRITE ELEMENTE VON / &JI / BIS / &JF / =:  =: ( < 1 : ENDE )
READ &A1 &A2
WRITE A1 = &A1 A2 = &A2
IF &A1 = &STR(0) THEN DO
   APO
   GOTO IR
   QUIT
IF &A1 ¬ = &STR() THEN SET &JI = &A1
IF &A2 = &STR() | &A2 < &A1 THEN SET &JF = &JI
ELSE SET &JF = &A2
WRITE JI = &JI , JF = &JF
IF &V = &STR(R) THEN GOTO RB
APO &JI &JF
SAT C&MH
ROUT
WRITE WEGRADIEREN ? =:  ( # : JA )
READ  &ANS
IF &ANS ¬ = &STR(#) THEN GOTO EL
RB: DPO &JI &JF
WRITE PRIMITIVE OBJEKTE &JI - &JF BESEITIGT
GOTO EL
```

```
AHA: LA
ROUT
WRITE GEAENDERTES &BIN ABSPEICHERN ? ( # = JA ) =:
READ &ANS
IF &ANS = &STR(#) THEN DO
   PWI
   SPIB &BIN B1
   WRITE DAS BILD &BIN WURDE IN &GSB. GESPEICHERT
   QUIT
GOTO IR
FGS: ENDGS
IF &V ¬ = &STR(!) THEN GOTO NBO
FIN: SET &KAMEL = &GOBLIN
EX SERVUS(ALOGS)
SET &KAMEL = &BIN &PAN &JI &JF &FORM &MG &MH &XLU &YLU &XRO &YRO
EX SERVUS(BEKI) 'MEM(PONUM)'
EXIT
```

12.3.6 Die G.S.-Prozedur PONUM

PONUM erleichtert die Identifikation einzelner Elemente in einem komplexen Objekt eines GS-Bildes durch umfärben der Elemente.

Benötigte Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(TUKOR)
- tso000.SERVUS.CLIST(ALOGS)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(TAGOK)
- tso000.SERVUS.CLIST(GLIEDER)
- tso000.SERVUS.CLIST(OBJEKT)

Benötigte Hilfsdateien :

- tso000.MEMORY(PONUM)
- tso000.MEMORY(BOOKS)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT MAIN MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 BIN TON AP
WRITE
WRITE GS-PROZEDUR PONUM
WRITE ELEMENTE EINES BILDTEILS
WRITE WERDEN UMGEFAERBT
WRITE UM IHREN ORDNUNGSZAHLEN ZU FINDEN.
SET &EA = 0
EX SERVUS(ALOGS)
SET &GOBLIN = &KAMEL
SET &SYSDVAL = &KAMEL
READDVAL &SIZE &GSB &UFB &SF &ZUB
IF &SIZE = &STR(0) THEN EXIT
SET &EA = 0
```

```
EX SERVUS(BEKI) 'MEM(PONUM)'
SET &NOTNAGL=&KAMEL
SET &SYSDVAL=&KAMEL
READDVAL &BIN &PAN &JI &JF &FORM &MG &MH &XLU &YLU &XRO &YRO
SET &N3=&PAN
SET &TON=&GSB
WRITE BEGINN DER PROZEDUR
SET &KAMEL=&FORM &MG &MH &XLU &YLU &XRO &YRO
NBO: EX SERVUS(OBJEKT)
SET &PAN=&N3
SET &SYSDVAL=&KAMEL
READDVAL &FORM &MG &MH &XLU &YLU &XRO &YRO
SET &EFE=&STR(A4)&FORM
INITGS -&SIZE
OW &AP
DWI 0 0 &EFE
RPIB &BIN B1
PWI &XLU &YLU &XRO &YRO
IF &PAN ¬ = &STR(#) THEN APA &PAN
APO
SAT C&MG
JOC: SET &V = &STR()
WRITE
WRITE VORGEHEN =:   W : BILD ZEIGEN
WRITE              ! : ENDE
WRITE              O : OBJEKT / FENSTER AENDERN
WRITE              E : EINZELELEMENTE UMFAERBEN
WRITE         SONST : REIHENWEISE UMFAERBEN
READ  &V
IF &V = &STR(!) | &V = &STR(O) THEN GOTO FGS
IF &V = &STR(W) THEN GOTO AHA
IF &V = &STR(E) THEN GOTO EL
WRITE ANFANG / &JI / =:
READ  &AI
IF &AI ¬ = &STR() THEN SET &JI=&AI
IF &JI < 1  THEN SET &JI = 1
WRITE ENDE / &JF / =:
READ  &AF
IF &AF ¬ = &STR() THEN SET &JF=&AF
IF &JF < &JI  THEN SET &JF = &JI + 1
WRITE ABSTAND / &JD / =:
READ  &AD
IF &AD ¬ = &STR() THEN SET &JD=&AD
IF &JD < 1  THEN SET &JD = 1
SET &K=0
SET &LF=&JI-1
SET &M=-1
DO WHILE K < 49
   SET &M=&M+1
   IF &M > = 7 THEN SET &M = &M-7
   SET &K=&K+1
   SET &LI=&LF+1
   SET &LF=&LI+&JD-1
   IF &LF > &JF THEN SET &LF = &JF
   WRITE &LI &STR( < L < ) &LF  &STR( , )  C&M
   APO &LI &LF
   SAT C&M
```

```
   IF &LF = &JF THEN GOTO CR
   QUIT
CR: ROUT
   APO &JI &JF
   SAT C&MG
   APO
GOTO JOC
EL: WRITE ELEMENTNUMMER / &JI / =: ( < 1 : ENDE )
READ  &ANS
IF &ANS = &STR() THEN GOTO N8
IF &ANS < 1 THEN DO
   ROUT
   APO
   GOTO JOC
   QUIT
IF &ANS ¬ = &STR() THEN SET &JI=&ANS
N8: APO &JI
SAT C&MH
GOTO EL
AHA: LA
ROUT
WRITE GEAENDERTES &BIN ABSPEICHERN ? ( # = JA ) =:
READ &ANS
IF &ANS=&STR(#) THEN DO
   PWI
   SPIB &BIN B1
   WRITE DAS BILD &BIN WURDE IN &GSB. GESPEICHERT
   QUIT
GOTO JOC
FGS: ENDGS
IF &V ¬ = &STR(!) THEN GOTO NBO
FIN: SET &KAMEL=&GOBLIN
EX SERVUS(ALOGS)
SET &KAMEL=&BIN &PAN &JI &JF &FORM &MG &MH &XLU &YLU &XRO &YRO
EX SERVUS(BEKI) 'MEM(PONUM)'
EXIT
```

12.4 Hilfsprozeduren

Die Hilfsprozeduren erledigen Teilaufgaben, die in vielen verschiedenen Prozeduren in derselben Weise auftreten.

Die Verständigung zwischen Hilfsprozedur und Prozedur erfolgt sowohl durch die Variablen im Kopf der Hilfsprozedur, als auch durch die GLOBAL-Variablen.

12.4.1 Die Hilfsprozedur ADA

ADA legt bei Bedarf unterschiedlich geartete Dateien an .

Die Eingabegrößen:

DSNAM ist die Name der Datei,
LIN steuert Format und Blocklänge der Datei,
KA ist die Anzahl der "directory-blocks" der (gegliederten) Datei,

WO ist die Name der Platte wohin die neue Datei hingelegt werden soll.

Die Prozedurliste:

```
PROC 1 DSNAM LIN(80) KA(0) WO(tso00x)
CONTROL PROMPT NOFLUSH MSG
IF &LIN = &STR(80) THEN +
ATTRIB JOKER LRECL(&LIN) BLKSIZE(3120) RECFM(F B)
IF &LIN = &STR(132) THEN +
ATTRIB JOKER LRECL(&LIN) BLKSIZE(3036) RECFM(F B)
IF &LIN = &STR(400) THEN +
ATTRIB JOKER LRECL(&LIN) BLKSIZE(3200) RECFM(F B)
IF &LIN = &STR(133) THEN +
ATTRIB JOKER LRECL(&LIN) BLKSIZE(3059) RECFM(F B A)
IF &LIN = &STR(180) THEN +
ATTRIB JOKER LRECL(&LIN) BLKSIZE(32760) RECFM(V S)
IF &LIN = &STR(X) THEN +
ATTRIB JOKER LRECL(&LIN) BLKSIZE(19069) RECFM(V B S)
IF &LIN = &STR(0) THEN +
ATTRIB JOKER LRECL(&LIN) BLKSIZE(19069) RECFM(U)
ALLOC DA('&DSNAM') VOLUME(&WO) NEW UNIT(DISK) +
SPACE(10,2) TRA DIR(&KA) USING(JOKER)
FREE ATTRLIST(JOKER)
EXIT
```

12.4.2 Die Hilfsprozedur ALOGS

ALOGS allokiert die Dateien , die bei einer G.S.-Sitzung in der Regel benötigt werden. Am Ende der Sitzung werden die Dateien wieder befreit.

ALOGS benötigt die Hilfsdatei

- tso000.MEMORY(BOOKS)

sowie die Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(TUKOR)

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT NOFLUSH MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON AP
IF &EA¬ =0 THEN GOTO K5
EX SERVUS(BEKI) 'MEM(BOOKS)'
SET &NOTNAGL=&KAMEL
SET &SYSDVAL=&KAMEL
READDVAL &AP &SIZE &PRO1 &BUCH1 &PROU &UFOS &SF &PRO2 &BUCH2
EX SERVUS(TUKOR)
ALLOC FI(COMM) DS(*)
ALLOC FI(ADMDEFS) DA('TSOSYS.ADMDEFS') SHR REU
IF &AP=&STR(GAT) THEN +
ALLOC FI(GAAPLIB) DS('TSOSYS.GAS.FONT') SHR REU
```

```
IF &AP=&STR(T15) THEN GSA
IF &AP¬=&STR(GAT) && &AP ¬= &STR(T15) THEN +
ALLOC FI(ADMSYMBL) DS('TSOSYS.GDDM.GDDMSYM') SHR REU
IF &AP=&STR(T25) THEN +
ALLOC FI(GSLINK) DS('TSO186.TGS.LOAD') SHR REU
SET &KAMEL=&AP &SIZE
WRITE GS BILDER-DATEI :
SET &N1=&PRO1
SET &N2=&BUCH1
SET &N3=&STR(#)
EX SERVUS(NEVE) 'LQ(1)'
IF &N1=&STR(#) THEN SET &SIZE=0
IF &SIZE=0 THEN DO
   SET &KAMEL=&NOTNAGL
   GOTO K3
   END
SET &GSB=&TON
ALLOC FI(B1) DA('&GSB') SHR REU
WRITE GS UFO-DATEI :
SET &N1=&PROU
SET &N2=&UFOS
SET &N3=&STR(LOAD)
EX SERVUS(NEVE) 'LQ(1)'
SET &UFB=&TON
IF &UFB¬=&STR() THEN ALLOC FI(UFOLIB) DS('&UFB') SHR REU
ELSE SET &UFB=&STR(#)
WRITE ZUSATZ DATEI / &SF / =: ( # = "KEINE" | G = "GDF"-DATEI )
READ &ANS
IF &ANS¬=&STR() THEN SET &SF=&ANS
IF &SF¬=&STR(#) && &SF¬=&STR(G) THEN SET &SF=&STR(B)
SET &KAMEL=&KAMEL &SF
IF &SF=&STR(#) THEN DO
   SET &KAMEL=&KAMEL &PRO2 &BUCH2
   GOTO K3
   END
SET &N1=&PRO2
SET &N2=&BUCH2
SET &N3=&STR(#)
EX SERVUS(NEVE) 'LQ(1)'
IF &TON=&STR() THEN SET &SF=&STR(#)
IF &SF=&STR(B) THEN ALLOC FI(B2)    DA('&TON') SHR REU
IF &SF=&STR(G) THEN ALLOC FI(ADMGDF) DA('&TON') SHR REU
K3: SET &GOBLIN=&SIZE &GSB &UFB &SF &TON
EX SERVUS(BEKI) 'MEM(BOOKS)'
SET &KAMEL=&GOBLIN
GOTO FIN
K5: FREE FI(B1,COMM,ADMDEFS)
SET &SYSDVAL=&KAMEL
READDVAL &SIZE &GSB &UFB &SF &ZUB
IF &AP=&STR(GAT) THEN FREE FI(GAAPLIB)
IF &AP¬=&STR(GAT) && &AP¬=&STR(T15) THEN FREE FI(ADMSYMBL)
IF &AP=&STR(T25) THEN FREE FI(GSLINK)
IF &UFB¬=&STR(#) THEN FREE FI(UFOLIB)
IF &SF¬=&STR(#) THEN DO
   IF &SF=&STR(B) THEN FREE FI(B2)
   IF &SF=&STR(G) THEN FREE FI(ADMGDF)
   WRITENR COMPRESS &ZUB ? =: ( "ENTER" = JA )
```

```
   READ &ANS
   IF &ANS = &STR() THEN DO
      COMPRESS '&ZUB' NOLIST
      REDUCE '&ZUB'
      SPACE '&ZUB'
      END
   END
WRITENR COMPRESS &GSB ? =: ( "ENTER" = JA )
READ &ANS
IF &ANS = &STR() THEN DO
      COMPRESS '&GSB' NOLIST
      REDUCE '&GSB'
      SPACE '&GSB'
   END
FIN: EXIT
```

12.4.3 Die Hilfsprozedur BEKI

BEKI liest - falls EA=0 ist - Eingangswerte einer Anzahl von Prozedurgrößen von der Datei tso000.MEMORY(MEM) und speichert - falls EA=1 ist - dieselbe Größen als Ausgangswerte in die Datei. Falls tso000.MEMORY(MEM) nicht vorhanden , legt BEKI sie an. Dazu benötigt BEKI die Hilfsprozedur tso000.SERVUS.CLIST(ADA) .

Die Prozedurliste:

```
PROC 0 MEM()
CONTROL PROMPT NOFLUSH MSG END(QUIT)
GLOBAL KAMEL EA
IF &SYSDSN('&SYSPREF..MEMORY(&MEM)') = OK THEN GOTO AL
IF &SYSDSN('&SYSPREF..MEMORY') = OK THEN GOTO CM
EX SERVUS(ADA) '&SYSPREF..MEMORY LIN(80) KA(10)'
WRITE '&SYSPREF..MEMORY' ALLOCATED
CM: IF &SYSDSN('&SYSPREF..MEMORY(&MEM)') ¬ = OK THEN DO
   WRITE REQUIRED MEMBER NOT FOUND , IT WILL BE CREATED .
   EDIT '&SYSPREF..MEMORY(&MEM)' NEW FORTGI NUM
   10 ????
   LIST
   UNNUM
   END SAVE
   QUIT
AL: ALLOC FILE(MEMORY) DA('&SYSPREF..MEMORY(&MEM)') SHR REU
IF &EA=1 THEN GOTO IN
WRITE LESEN AUS '&SYSPREF..MEMORY(&MEM)'
ERROR DO
   WRITE MEMBER IS EMPTY
   GOTO CL
   QUIT
OPENFILE MEMORY
GETFILE MEMORY
SET &KAMEL=&STR(&MEMORY)
CL: CLOSFILE MEMORY
SET &EA=1
FREE DA('&SYSPREF..MEMORY(&MEM)')
GOTO FIN
IN: SET &L = &LENGTH(&STR(&KAMEL))
```

```
OPENFILE MEMORY OUTPUT
SET &MEMORY=&SUBSTR(1:&L,&KAMEL)
PUTFILE MEMORY
CLOSFILE MEMORY
FREE DA('&SYSPREF..MEMORY(&MEM)')
WRITE SCHREIBEN IN '&SYSPREF..MEMORY(&MEM)'
FIN: EXIT
```

12.4.4 Die Hilfsprozedur FEJ

FEJ schreibt Jobkarten für CNTL-Prozeduren. Der Benutzer möge den Prozedurkopf mit den eigenen Angaben ausfüllen. Auch in der Karte 0111 - am Ende der Prozedur - soll man anstatt RM00x die zutreffende Nummer des Ausgabegerätes eintragen.

Mit der Eingabegröße **R** kann man den REGION-Parameter ändern, mit **NAME** kann man eine Name für die CNTL-Datei wählen.

FEJ benötigt die Hilfsprozeduren tso000.SERVUS.CLIST(ADA) und tso000.SERVUS.CLIST(BEKI).

Die Prozedurliste:

```
PROC 0 R(1000) KOR(NEW) NAME() +
USER(Benutzer) ACCOUNT('0abc,xyz,p9x9y')
CONTROL PROMPT NOFLUSH MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 M1 TON
WRITE ZUSAMMENSTELLUNG DER JOB-KARTE(-N)
SET &EA=0
EX SERVUS(BEKI) 'MEM(FEJ)'
SET &SYSDVAL=&KAMEL
READDVAL &N1 &N2 &N3 &US &AC
IF &NAME ¬= &STR() THEN SET &N3=&NAME
WRITE DRUCKQUALITAET / &N1 / =: ( H | A | W = TSO-ZEICHENSATZ )
READ &ANS
IF &ANS ¬= &STR() THEN SET &N1= &ANS
WRITE DRUCKORT / &OO / =: ( "ENTER"=NORM | O = RM003 )
READ &OO
WRITENR JOB-KENNZEICHNUNG / &N2 / =:
READ &ANS
IF &ANS ¬= &STR() THEN SET &N2=&ANS
IF &LENGTH(&N2) > 2 THEN SET &N2 = &SUBSTR(1:2,&N2)
IF &US ¬= &STR() THEN SET &USER=&US
WRITENR "USER" / &USER / =: ( "ENTER"=JA )
READ &ANS
IF &ANS ¬= &STR() THEN SET &USER=&ANS
IF &AC ¬= &STR() THEN SET &ACCOUNT=&AC
WRITE "ACCOUNT" / &ACCOUNT / =: ( "ENTER"=JA )
WRITE NEUE ZEICHENKETTE MIT HOCHKOMMATA ('0123,456,ABCDE') EINGEBEN !
READ &ANS
IF &ANS ¬= &STR() THEN SET &ACCOUNT=&ANS
WRITENR "LINES" ? =: ( "ENTER"=KEINE | 10 | ... )
READ &M1
IF &M1 ¬= &STR() THEN SET &OO=&STR()
IF &N3 = &STR() THEN SET &N3=&STR(CLIST)
OK: IF &SYSDSN('&SYSPREF..&N3..CNTL') = OK THEN DO
   WRITE DATEI '&SYSPREF..&N3..CNTL' BEREITS VORHANDEN !
```

```
   WRITE SOLLTE SIE UEBERSCHRIEBEN WERDEN ? =: +
   ( "ENTER" = JA | NEUE NAME )
   READ &ANS
   IF &ANS ¬ = &STR() THEN DO
      SET &N3 = &ANS
      GOTO OK
      QUIT
   ELSE DO
      EDIT &N3..CNTL OLD NONUM
      RENUM
      DEL 10 9990
      END SAVE
      SET &KOR = &STR(OLD)
      QUIT
   QUIT
SET &KAMEL = &STR(&N1 &N2 &N3 &USER &STR('&ACCOUNT') )
EX SERVUS(BEKI) 'MEM(FEJ)'
EDIT &N3..CNTL &KOR NUM
0100 //&SYSUID.&N2 JOB (&ACCOUNT),&USER,MSGLEVEL = (1,1),REGION = K,
0110 // NOTIFY = &SYSUID,MSGCLASS = &N1
IF &OO = &STR(O) THEN 0111 //*MAIN LINES = 3,ORG = RM003
IF &M1 ¬ = &STR() THEN 0111 &STR(//&STR(*MAIN LINES = &M1))
END SAVE
EXIT
```

12.4.5 Die Hilfsprozedur GLIEDER

Diese Hilfsprozedur listet die Glieder einer Datei "DSNAM" am Bildschirm. DSNAM muß der vollständige Name sein.

Die Prozedurliste:

```
PROC 1 DSNAM
CONTROL PROMPT NOFLUSH MSG
IF &SYSDSN('&DSNAM') ¬ = OK THEN DO
   WRITE DATEI '&DSNAM' GIBT ES NICHT !
   EXIT
   END
WRITE GLIEDER DER DATEI '&DSNAM'
SET &SYSOUTTRAP = 200
LISTDS '&DSNAM' MEMBERS
SET I = 7
SET M = 0
DO WHILE &I < = &SYSOUTLINE
   SET &M = &M + 1
   SET DSN = &&SYSOUTLINE&I
   WRITE &M &DSN
   SET &I = &I + 1
   END
EXIT
```

12.4.6 Die Hilfsprozedur NEVE

NEVE hilft dem Benutzer beim Zusammenstellen des Namens einer Datei und prüft, ob die genannte Datei vorhanden ist oder nicht.

Zu den Eingabegrößen LQ , CQ , SQ :

LQ wählt die Namensteile , die erfragt werden sollen , bei
LQ = 1 werden "Projekt" und "Name" erfragt , bei
LQ = 2 wird nur "Name" erfragt und bei
LQ = 3 werden nur "Name" und "Typ" ermittelt. Bei
LQ = 0 werden alle drei Namensteile erfragt.

CQ steuert die Datei-Kontrolle: bei
CQ = 0 wird überprüft, ob die Datei vorhanden sei , bei
CQ ≠ 0 erfolgt keine Kontrolle.

SQ steuert die Registratur : bei
SQ = 0 werden die erfragten Namensteilen in die "MEMORY" eingetragen , bei
SQ ≠ 0 dagegen nicht.

Die Prozedurliste:

```
PROC 0 LQ(0) CQ(0) SQ(0)
CONTROL PROMPT NOFLUSH MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON
SET &NOTA=&STR(MEMBER SPECIFIED, BUT DATASET IS NOT PARTITIONED)
IF &N1 = &STR() | &N1 = &STR(#) THEN SET &N1=&SYSPREF
IF &M1 = &STR() | &M1 = &STR(#) | +
&DATATYPE(&M1)¬ =&STR(CHAR) THEN SET &M1=&STR(TEMP)
PRJ: SET &TON = &STR()
IF &LQ > 1 THEN GOTO NAM
WRITENR PROJECT / &N1 / =: ( "ENTER"=JA | # = KEINE DATEI )
READ &A1
IF &A1 ¬ = &STR() THEN DO
   IF &A1 = &STR(#) THEN GOTO REG
   SET &N1 = &A1
   IF &N1 = &STR(I) THEN SET &N1 = &STR(&SYSUID)
   IF &N1 = &STR(T) THEN SET &N1 = &STR(&SYSPREF)
   END
NAM: WRITENR NAME / &N2 / =: ( "ENTER"(=JA) )
READ &AN
IF &AN ¬ = &STR() THEN SET &N2 = &AN
TYP: IF &LQ = 0 | &LQ = 3 THEN DO
   WRITENR TYP / &N3 / =: ( "ENTER"(=JA) | # (=TYP FEHLT) )
   READ &AN
   IF &AN ¬ = &STR() THEN SET &N3 = &AN
   END
IF &N3 ¬ = &STR(#) THEN SET &TON= &N1..&N2..&N3
            ELSE SET &TON= &N1..&N2
CTR: IF &CQ ¬ = 0 THEN GOTO REG
IF &SYSDSN('&TON')¬ =OK THEN DO
   WRITE &TON &SYSDSN('&TON') !
   WRITE
   WRITE KATALOG DER &N1 -DATEIEN :
   WRITE
```

```
   CAT &N1
   GOTO PRJ
   END
IF &SYSDSN('&TON(&M1)')=&NOTA THEN DO
   WRITE &SUBSTR(22:48,&NOTA)
   SET &M1=&STR(#)
   END
REG: IF &SQ ¬ = 0 THEN GOTO FF
IF &LQ < 2 THEN SET &KAMEL=&KAMEL &N1
SET &KAMEL=&KAMEL &N2
IF &LQ = 0 | &LQ = 3 THEN SET &KAMEL=&KAMEL &N3
FF: IF &A1 = &STR(#) THEN SET &N1=&A1
EXIT
```

12.4.7 Die Hilfsprozedur OBJEKT

OBJEKT legt einen Teil eines Bildes für weitere Bearbeitung fest. Die Hilfsprozedur unterstüzt die Wahl des Fensters, zweier Arbeitsfarben sowie die Vergrößerung eines Teilbildes.

Die Prozedurliste:

```
PROC 0 X1U(0.0) Y1U(0.0) X2O(11.7) Y2O(8.28)
CONTROL PROMPT NOFLUSH MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 PAN BIN TON GER
SET &SYSDVAL=&KAMEL
READDVAL &FORM &MG &MH &XLU &YLU &XRO &YRO
IF &XLU=&STR() THEN SET &XLU=&X1U
IF &YLU=&STR() THEN SET &YLU=&Y1U
IF &XRO=&STR() THEN SET &XRO=&X2O
IF &YRO=&STR() THEN SET &YRO=&Y2O
NBO: WRITE OBJEKT " &BIN..&PAN " IN ORDNUNG ?
WRITE    "ENTER" = JA
WRITE    "B" = NEUES BILD
WRITE    "T" = NEUES TEILBILD
WRITE    "C" = ARBEITSFARBEN WAHLEN
WRITE    "W" = FENSTER
READ &ANS
IF &ANS=&STR() THEN GOTO FIN
IF &ANS¬=&STR() THEN SET &N1=&ANS
IF &N1=&STR(B) THEN DO
   SET &LAB=&BIN
   EX SERVUS(TAGOK)
   IF &BIN=&STR(!) THEN DO
      SET &N1=&BIN
      SET &BIN=&LAB
      QUIT
   QUIT
IF &N1=&STR(!) THEN EXIT
IF &ANS=&STR(W) THEN GOTO KF
IF &ANS=&STR(T) THEN GOTO TBI
IF &ANS=&STR(C) THEN DO
   NC0: WRITENR GRUNDFARBE / &MG / =: ( 0 , ... , 7 )
   READ  &ANS
   IF &ANS > 7 THEN GOTO NC0
```

```
   IF &ANS ¬ = &STR() THEN SET &MG=&ANS
   WRITE GRUNDFARBE = C&MG
   NC1: WRITENR FARBE ZUM HERVORHEBEN / &MH / =: ( 0 , ... , 7 )
   READ &ANS
   IF &ANS > 7 THEN GOTO NC1
   IF &ANS ¬ = &STR() THEN SET &MH=&ANS
   WRITE HERVORHEBEN MIT DER FARBE C&MH
   QUIT
GOTO NBO
IF &ANS=&STR(Z) THEN DO
   KF: WRITE FENSTER IM AUGENBLICK :
   WRITE (&XLU,&YLU) - (&XRO,&YRO) , DIN-&FORM FORMAT
   WRITE IN ORDNUNG =:
   WRITE      "ENTER" = JA
   WRITE      "L" = LINKSUNTEN AENDERN
   WRITE      "R" = RECHTSOBEN AENDERN
   WRITE      "F" = FENSTERFORM AENDERN
   WRITE      "SONST" = NATUERLICHES FENSTER
   READ &ANS
   IF &ANS=&STR() THEN GOTO FIN
   IF &ANS¬=&STR(L) && &ANS¬=&STR(R) && &ANS¬=&STR(F) THEN DO
      SET &XLU = &STR()
      SET &YLU = &STR()
      SET &XRO = &STR()
      SET &YRO = &STR()
      GOTO FIN
      QUIT
   IF &ANS=&STR(F) THEN DO
      IK: WRITENR ( "H" = DIN-H | "Q" = DIN-Q )
      READ &ANS
      IF &ANS¬=&STR() THEN SET &FORM=&ANS
      IF &FORM¬=&STR(H) && &FORM¬=&STR(Q) THEN GOTO IK
      QUIT
   IF &ANS=&STR(L) THEN DO
      WRITENR ECKE LINKSUNTEN : X / &XLU / =: , Y / &YLU / =:
      READ &AX &AY
      IF &AX¬=&STR() THEN SET &XLU = &STR(&AX)
      IF &AY¬=&STR() THEN SET &YLU = &STR(&AY)
      QUIT
   IF &ANS=&STR(R) THEN DO
      WRITENR ECKE RECHTSOBEN : X / &XRO / =: , Y / &YRO / =:
      READ &AX &AY
      IF &AX¬=&STR() THEN SET &XRO = &STR(&AX)
      IF &AY¬=&STR() THEN SET &YRO = &STR(&AY)
      QUIT
   GOTO KF
   QUIT
TBI: WRITE NAME DES TEILBILDES / &PAN / =:
WRITE      "ENTER" : JA
WRITE           ? : BILD-STRUKTUR
WRITE           # : KEIN TEILBILD
READ &ANS
IF &ANS = &STR(?) THEN DO
   INITGS 500
   OW &GER
   RPIB &BIN B1
   LS
```

```
    ENDGS
    GOTO TBI
    QUIT
IF &ANS¬ = &STR() THEN  SET &PAN = &ANS
GOTO NBO
FIN: SET &KAMEL = &FORM &MG &MH &XLU &YLU &XRO &YRO
EXIT
```

12.4.8 Die Hilfsprozedur SERVIN

SERVIN bereitet die Herstellung eines SERVUS-Bildes vor.

Zuerst erfragt die Prozedur vom Benutzer verschiedene bildspezifische Eingabedaten (DIN-Format , KfK-Emblem , ...) sowie die Namen der für die Bildherstellung benötigten Hilfsdateien. Dann erfolgt das Anknüpfen dieser Dateien und die vorgesehene FORTRAN-Prozedur wird aufgerufen.

LANAM ist die Name der FORTRAN-Prozedur.

SERVIN benötigt die Hilfsdatei tso000.MEMORY(SERVIN) und folgende Hilfsprozeduren :

- tso000.SERVUS.CLIST(ADA)
- tso000.SERVUS.CLIST(BEKI)
- tso000.SERVUS.CLIST(NEVE)
- tso000.SERVUS.CLIST(TAGOK)
- tso000.SERVUS.CLIST(GLIEDER)

Die Prozedurliste:

```
PROC 1 LANAM
CONTROL PROMPT MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON
SET &DRA = &STR(LA&SUBSTR(1:4,&LANAM))
SET &EA = 0
EX SERVUS(BEKI) 'MEM(SERVIN)'
SET &SYSDVAL = &KAMEL
READDVAL &IDIN &FEDA &BILN &ABT &CALE +
&N1 &N2 &N3 &M1 &Q1 &Q2 &Q3 &Q4
WRITE FENSTER / &IDIN / =: ( 1 : DIN-HOCH | SONST : DIN QUER )
READ &ANS
IF &ANS ¬ = &STR() THEN SET &IDIN = &ANS
IF &IDIN ¬ = &STR(1) THEN SET &IDIN = &STR(0)
WRITE QUELLE DER RAHMENDATEN / &FEDA / =:
WRITE        0 : DATEI &DRA
WRITE     SONST : &LANAM-VORGABE
READ &ANS
IF &ANS ¬ = &STR() THEN SET &FEDA = &ANS
IF &FEDA¬ = &STR(0) THEN SET &FEDA = &STR(1)
BIN: WRITE VORGESEHENE BILDNAME / &BILN / =:
READ &ANS
IF &ANS ¬ = &STR() THEN SET &BILN = &ANS
SET &L = &LENGTH(&BILN)
IF &L > 8 THEN SET &BILN = &SUBSTR(1:8,BILN)
IF &L < 6 THEN DO
   WRITE ZU KURZ ! BITTE MINDESTENS 6 ZEICHEN .
   GOTO BIN
```

```
   END
WRITE BILDER MIT KFK-SYMBOL / &ABT / =: +
( "?" : NEIN | INSTITUTSKUERZEL )
READ &ANS
IF &ANS ¬ = &STR() THEN DO
   SET &L=&LENGTH(&ANS)
   IF &L > 4 THEN SET &L = 4
   SET &ABT=&SUBSTR(1:&L,&ANS)
   END
WRITE BILDER MIT DATUM / &CALE / =: ( "0" : JA | SONST NEIN )
READ &ANS
IF &ANS ¬ = &STR() THEN SET &CALE=&ANS
IF &CALE¬ = &STR(0) THEN SET &CALE=&STR(9)
SET &TEVE=&STR(&IDIN &FEDA &BILN &ABT &CALE)
SET &KAMEL=&TEVE
WRITE ANGABEN ZUR QUELLDATEI &DRA :
EX SERVUS(NEVE)
IF &N1 = &STR(#) THEN GOTO FIN
IF &M1 ¬ = &STR(#) THEN DO
   EX SERVUS(TAGOK)
   IF &M1 = &STR(!) THEN GOTO FIN
   SET &TON=&STR(&TON(&M1))
   END
ALLOC DA('&TON') F(FT10F001) SHR REU
SET &KAMEL = &STR(&KAMEL &M1)
WRITE ANGABEN ZUR ZUSAETZLICHEN TEXT-DATEIEN ABB1 ... ABB4 :
SET &TON=&SYSPREF..MEMORY
SET &I=0
SET &N=0
DO WHILE &I < 4
   SET &I=&I+1
   SET MM = &&Q&I
   SET &M1 = &MM
   EX SERVUS(TAGOK)
   IF &M1 = &STR(!) THEN GOTO FIE
   SET &FI = &STR(FT3)&STR(&I)&STR(F001)
   ALLOC DA('&TON(&M1)') F(&FI) SHR REU
   SET &KAMEL = &KAMEL &M1
   SET &N=&I
   END
FIE: EX SERVUS(BEKI) 'MEM(SERVIN)'
SET &N1=&N
SET &KAMEL=&TEVE
ALLOC DA(*) F(COMM) SHR REU
ALLOC FI(ADMDEFS) DA('TSOSYS.ADMDEFS') SHR REU
FIN: EXIT
```

12.4.9 Die Hilfsprozedur SUSA

SUSA speichert die fertiggestellte CNTL-Prozedur und auf Verlangen lässt sie den Auftrag ausführen.

Die Prozedurliste:

```
PROC 1 NAME
```

```
CONTROL PROMPT NOFLUSH MSG END(QUIT)
GLOBAL KAMEL EA N1 N2 N3 M1 TON
UNNUM
SET &TON = &SYSPREF..&N3..CNTL
SAVE '&TON'
WRITE GESPEICHERT ALS '&TON'
WRITE JOB ABGEBEN ? ( "ENTER"=JA ) =:
READ &ANS
IF &ANS = &STR() THEN  SUBMIT '&TON'
END NOSAVE
EXIT
```

12.4.10 Die Hilfsprozedur TAGOK

TAGOK hilft dem Benutzer beim aufsuchen eines Gliedes einer gegliederten Datei. Falls **CQ** = 0 ist, dann wird auch noch überprüft, ob das Glied vorhanden sei. TAGOK selber benötigt eine Hilfspozedur : tso000.SERVUS.CLIST(GLIEDER) .

Die Prozedurliste:

```
PROC 0 CQ(0)
CONTROL PROMPT NOFLUSH MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON
ANF: WRITE GLIED / &M1 / +
( "ENTER" = OK | ? = "LISTE" | ! = "ENDE" ) =:
READ &ANS
IF &ANS=&STR(?) THEN DO
   EX SERVUS(GLIEDER) '&TON'
   GOTO ANF
   END
IF &ANS¬ =&STR() THEN SET &M1 = &ANS
IF &ANS=&STR(!) THEN EXIT
IF &SYSDSN('&TON.(&M1)')¬ =OK && &CQ = 0 THEN DO
   WRITE NICHT VORHANDEN !
   GOTO ANF
   END
EXIT
```

12.4.11 Die Hilfsprozedur TUKOR

TUKOR unterstützt den Benutzer bei der Wahl eines graphischen Arbeitzplatzes.

Die Prozedurliste:

```
PROC 0
CONTROL PROMPT NOFLUSH MSG
GLOBAL KAMEL EA N1 N2 N3 M1 TON AP
SET &N3 = 0
IF &AP = 2 THEN SET &AP=&STR(327)
IF &AP = 3 THEN SET &AP=&STR(GAT)
IF &AP = 4 THEN SET &AP=&STR(TCX)
IF &AP = 5 THEN SET &AP=&STR(317)
IG: WRITE
WRITE GRAPHISCHER ARBEITSPLATZ / &AP / =:
```

```
WRITE "PCG" = IBM 5378 PC
WRITE "327" = IBM 3279 G | MEMOREX 2079
WRITE "GAT" = IBM 3277 GA
WRITE "T15" = TEKTRONIX 4015
WRITE "TCX" = TEKTRONIX CX 4109 A
WRITE "T25" = TEKTRONIX 4125
WRITE "317" = IBM 3179 G
READ &ANS
IF &ANS ¬= &STR() THEN SET &AP=&ANS
IF &AP = &STR(GAT) THEN SET &N3=3
IF &AP = &STR(PCG) | &AP = &STR(317) THEN SET &N3=5
IF &AP = &STR(327) THEN SET &N3=2
IF &AP = &STR(T15) | &AP = &STR(TCX) | +
&AP = &STR(T25) THEN SET &N3=4
IF &N3 < 2 | &N3 > 5 THEN DO
   WRITE GIBT'S NICHT !
   GOTO IG:
   END
EXIT
```

13.0 Literatur:

/1/ C. Frese, D. Lang, R. Pietschmann,
GML - Generalized Markup Language
Kurzanleitung für DCF-Benutzer,
KfK-Version 3
Unveröffentlichter Bericht 1988.

/2/ Fachausdrücke der Text- und Datenverarbeitung,
IBM Deutschland 1978.

/3/ K. Gogg, J. Buschmann u. a., GS-Handbuch.
Unveröffentlichter Bericht des KfK, Dez. 1982.
Stand Januar 1986.

/4/ C. Broeders, PLOTDAT - Programmbeschreibung,
Unveröffentlichter Bericht 1978.

/5/ M. Politzky, TRACEGS, Ein Programmsystem für die Zeichnungserstellung..., KFK 3237 ,1981.

/6/ J. Oehlschläger, Die Prozedur BIPLOT zur Plotter-Ausgabe von GS-Bildern, KFK 3748 ,1984.

/7/ H. Kleinheins, DDPLOT - Programmbeschreibung,
Unveröffentlichter Bericht 1978.

/8/ SIEMENS AKTIENGESELLSCHAFT, Operating System BS3000
FORTRAN77 User's Guide
Version 1, September 1982.
Chapt. 15. List-directed READ statement. P. 295

14.0 Abbildungen.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+  21. 4.89    THE GRAPHS WILL BE COPIED INTO THE DATASET    SZUSZOG .DATA     +
+                                                                              +
+                    STORAGE-NORM IS      "GRAPHIC0"                          +
+                                                                              +
+     DATE OF THE ALLOCATION:   21. 4.89      LAST SCRATCH:      21. 4.89     +
+                                                                              +
+     THE DATASET IS EMPTY                              TIME: 15H 56' 32"      +
+                                                                              +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


===>    1   GRAPH(   1/    1)  X : +   X    + - 1 -  +  F : +SIN(1*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.6378     ,  -1.000     < F <  1.000     , U=0.40

===>    2   GRAPH(   1/    2)  X : +   X    + - 1 -  +  F : +SIN(2*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)=-0.9825     ,  -1.000     < F <  1.000     , U=0.40

===>    3   GRAPH(   1/    3)  X : +   X    + - 1 -  +  F : +SIN(3*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.8757     ,  -1.000     < F <  1.000     , U=0.40

===>    4   GRAPH(   1/    4)  X : +   X    + - 1 -  +  F : +SIN(4*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)=-0.3665     ,  -1.000     < F <  1.000     , U=0.40

===>    5   GRAPH(   1/    5)  X : +   X    + - 1 -  +  F : +SIN(5*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)=-0.3111     ,  -1.000     < F <  1.000     , U=0.40
```

Abbildung 1. Funktionen eintragen.

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                               +
+     21. 4.89     LIST OF THE GRAPHS OF THE DATASET        SZUSZOG .DATA      +
+                                                                               +
+                         STORAGE-NORM IS      "GRAPHIC0"                       +
+                                                                               +
+     DATE OF THE ALLOCATION:   21. 4.89      LAST SCRATCH:      21. 4.89      +
+                                                                               +
+     THE DATASET CONTAINS     5 GRAPHS                 TIME: 16H 42' 36"      +
+                                                                               +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                               +
+   21. 4.89    THE GRAPHS WILL BE COPIED INTO THE DATASET    SZUSZOG .DATA    +
+                                                                               +
+                         STORAGE-NORM IS      "GRAPHIC0"                       +
+                                                                               +
+     DATE OF THE ALLOCATION:   21. 4.89      LAST SCRATCH:      21. 4.89      +
+                                                                               +
+     THE DATASET CONTAINS     5 GRAPHS                 TIME: 16H 42' 36"      +
+                                                                               +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 #      1   GRAPH(   1/    1)  X : +   X    + - 1 -  +  F : +SIN(1*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.6378     ,  -1.000     < F <  1.000     , U=0.40

===>    6   GRAPH(   2/    1)  X : +   X    + - 1 -  +  F : +SI2(1*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.4067     ,  -1.000     < F <  1.000     , U=0.40

 #      2   GRAPH(   1/    2)  X : +   X    + - 1 -  +  F : +SIN(2*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)=-0.9825     ,  -1.000     < F <  1.000     , U=0.40

===>    7   GRAPH(   2/    2)  X : +   X    + - 1 -  +  F : +SI2(2*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.9652     ,  -1.000     < F <  1.000     , U=0.40

 #      3   GRAPH(   1/    3)  X : +   X    + - 1 -  +  F : +SIN(3*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.8757     ,  -1.000     < F <  1.000     , U=0.40

===>    8   GRAPH(   2/    3)  X : +   X    + - 1 -  +  F : +SI2(3*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.7668     ,  -1.000     < F <  1.000     , U=0.40

 #      4   GRAPH(   1/    4)  X : +   X    + - 1 -  +  F : +SIN(4*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)=-0.3665     ,  -1.000     < F <  1.000     , U=0.40

===>    9   GRAPH(   2/    4)  X : +   X    + - 1 -  +  F : +SI2(4*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.1343     ,  -1.000     < F <  1.000     , U=0.40

 #      5   GRAPH(   1/    5)  X : +   X    + - 1 -  +  F : +SIN(5*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)=-0.3111     ,  -1.000     < F <  1.000     , U=0.40

===>   10   GRAPH(   2/    5)  X : +   X    + - 1 -  +  F : +SI2(5*X)+ - 1 -  +
X1= 0.0000E+00,X( 50)=  2.450     ,  0.0000E+00 < X <  2.500     , U=0.50
F1= 0.0000E+00,F( 50)= 0.9680E-01 ,  -1.000     < F <  1.000     , U=0.40
```

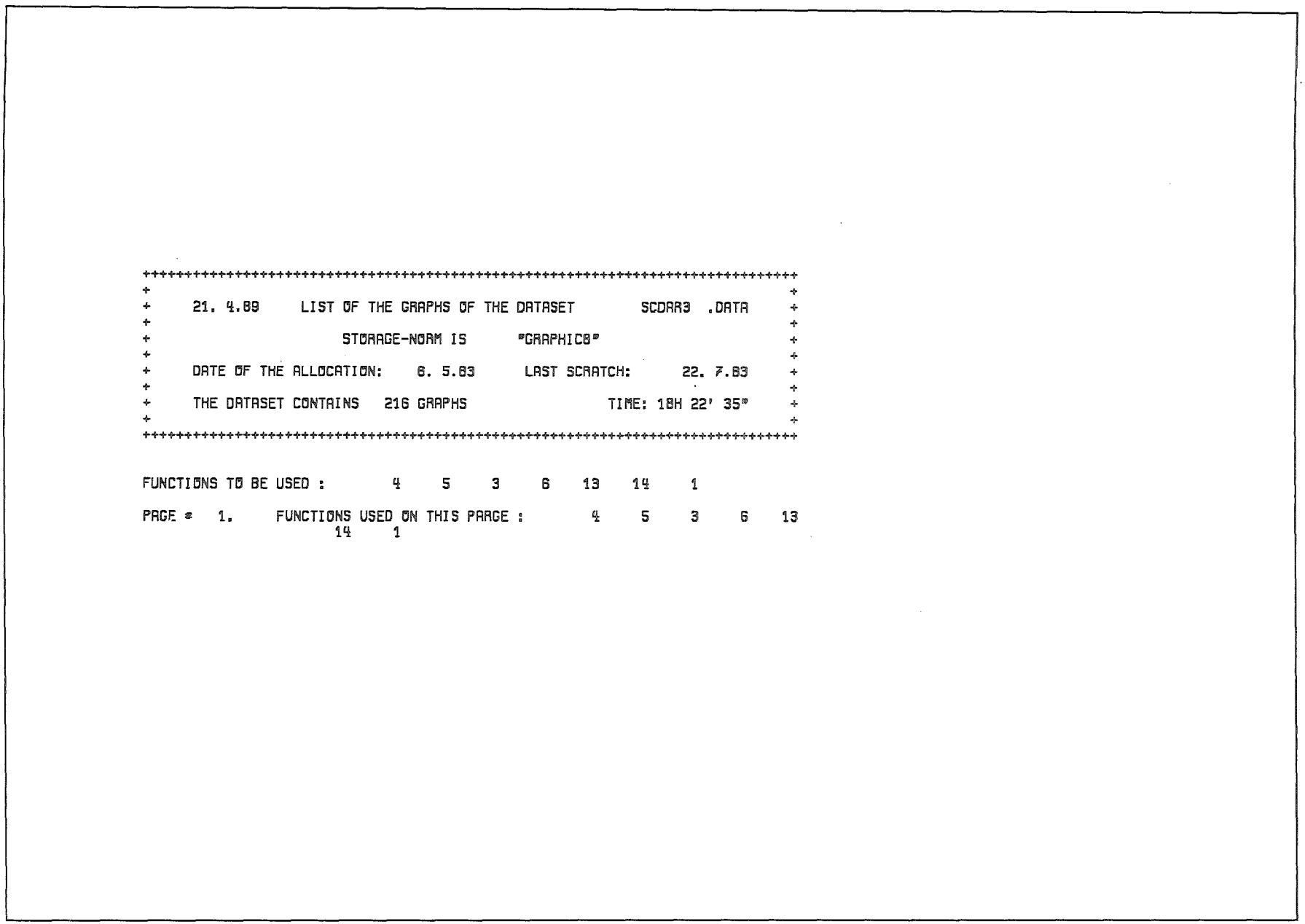
Abbildung 2. Lesen-Schreiben.

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+     21. 4.89     LIST OF THE GRAPHS OF A PLOTEASY DATASET                    +
+                                                                              +
+                          STORAGE-NORM IS      "PLOTEASY"                     +
+                                                                              +
+     THE DATASET CONTAINS    10 GRAPHS                  TIME: 16H 50' 40"     +
+                                                                              +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+   21. 4.89    THE GRAPHS WILL BE COPIED INTO THE DATASET    SZUSZOG .DATA    +
+                                                                              +
+                          STORAGE-NORM IS      "GRAPHICS"                     +
+                                                                              +
+     DATE OF THE ALLOCATION:   21. 4.89      LAST SCRATCH:      21. 4.89      +
+                                                                              +
+     THE DATASET CONTAINS    10 GRAPHS                  TIME: 16H 50' 40"     +
+                                                                              +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 #      1   GRAPH(   1/    1)  F : +SAILER  +
X1=  1017.    ,X( 12)=  1680.
F1=  2.531    ,F( 12)=  67.16

===>   11   GRAPH(   1/    1)  X : +-FEHLT- +-FEHLT- +  F : +SAILER  +-FEHLT- +
X1=  1017.    ,X( 12)=  1680.     ,  0.0000E+00 < X < 0.0000E+00 , U=0.00E+00
F1=  2.531    ,F( 12)=  67.16     ,  0.0000E+00 < F < 0.0000E+00 , U=0.00E+00

 #      2   GRAPH(   2/    1)  F : +SAILER  +
X1=  1697.    ,X( 13)=  2392.
F1=  70.12    ,F( 13)=  95.96

===>   12   GRAPH(   2/    1)  X : +-FEHLT- +-FEHLT- +  F : +SAILER  +-FEHLT- +
X1=  1697.    ,X( 13)=  2392.     ,  0.0000E+00 < X < 0.0000E+00 , U=0.00E+00
F1=  70.12    ,F( 13)=  95.96     ,  0.0000E+00 < F < 0.0000E+00 , U=0.00E+00

 #      3   GRAPH(   3/    1)  F : +SAILER  +
X1=  1030.    ,X( 14)=  1805.
F1=  3.790    ,F( 14)=  99.34

===>   13   GRAPH(   3/    1)  X : +-FEHLT- +-FEHLT- +  F : +SAILER  +-FEHLT- +
X1=  1030.    ,X( 14)=  1805.     ,  0.0000E+00 < X < 0.0000E+00 , U=0.00E+00
F1=  3.790    ,F( 14)=  99.34     ,  0.0000E+00 < F < 0.0000E+00 , U=0.00E+00

 #      4   GRAPH(   4/    1)  F : +SAILER  +
X1=  1866.    ,X(  6)=  2182.
F1=  116.3    ,F(  6)=  140.5

===>   14   GRAPH(   4/    1)  X : +-FEHLT- +-FEHLT- +  F : +SAILER  +-FEHLT- +
X1=  1866.    ,X(  6)=  2182.     ,  0.0000E+00 < X < 0.0000E+00 , U=0.00E+00
F1=  116.3    ,F(  6)=  140.5     ,  0.0000E+00 < F < 0.0000E+00 , U=0.00E+00

...  .....  ...............  ..........................................  +
...  .....  ...............  ..........................................  +
...  .....  ...............  ..........................................  +

 #     10   GRAPH(  10/    1)  F : +SAILER  +
X1=  1980.    ,X(  2)=  2452.
F1=  157.9    ,F(  2)=  202.3

===>   20   GRAPH(  10/    1)  X : +-FEHLT- +-FEHLT- +  F : +SAILER  +-FEHLT- +
X1=  1980.    ,X(  2)=  2452.     ,  0.0000E+00 < X < 0.0000E+00 , U=0.00E+00
F1=  157.9    ,F(  2)=  202.3     ,  0.0000E+00 < F < 0.0000E+00 , U=0.00E+00

+     FUNCTION NR.   11 NOT IN STOCK !                                        +
+                                  THE DATASET CONTAINS ONLY   10 GRAPHS       +
+                                                                              +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Abbildung 3. PLOTEASY-Datei.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+     21. 4.89      LIST OF THE GRAPHS OF THE DATASET        VIELBILD.DATA     +
+                                                                              +
+                          STORAGE-NORM IS       "GRAPHIC4"                    +
+                                                                              +
+     DATE OF THE ALLOCATION:     3. 5.84      LAST SCRATCH:      29. 5.87     +
+                                                                              +
+     THE DATASET CONTAINS   236 GRAPHS                  TIME: 10H  2' 48"     +
+                                                                              +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


FUNCTIONS TO BE USED :  ALL IN THE RANGE     1 -    99

 #      1   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : + P PIN  + *1.E+ 6+
X1= 0.5000    ,X(100)=  99.50     ,  0.0000E+00 < X <  100.0     , U= 10.
F1=  89.64    ,F(100)=  89.64     ,   89.00     < F <  150.0     , U= 10.

 #      2   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DM(W<S) + *1.E- 9+
X1= 0.5000    ,X(100)=  99.50     ,  0.0000E+00 < X <  100.0     , U= 10.
F1= 0.0000E+00,F(100)= 0.0000E+00 ,  -1.600     < F < 0.8200     , U=0.10

 #      3   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(RO,G)+    K   +
X1= 0.5000    ,X(100)=  99.50     ,  0.0000E+00 < X <  100.0     , U= 10.
F1= 0.3168E-07,F(100)= 0.3378E-07 , -0.1900     < F < 0.1100     , U=0.10E-01

 #      4   GRAPH(   3/   85)  X : + HOEHE  +   CM   +  F : +DW(S<F) + *1.E+ 3+
X1= 0.5000    ,X(100)=  99.50     ,  0.0000E+00 < X <  100.0     , U= 10.
F1=  2.470    ,F(100)=  2.470     ,   1.300     < F <  7.600     , U= 1.0

.....   ..  ..............  ....................  ....................
.....   ..  ..............  ....................  ....................
.....   ..  ..............  ....................  ....................

 #     96   GRAPH(   3/   95)  X : + HOEHE  +   CM   +  F : +DW(G<S) +    K   +
X1= 0.5000    ,X(100)=  99.50     ,  0.0000E+00 < X <  100.0     , U= 10.
F1= 0.2079E-01,F(100)= 0.2089E-01 ,  0.0000E+00 < F < 0.3500     , U=0.50E-01

 #     97   GRAPH(   3/   95)  X : + HOEHE  +   CM   +  F : +DY(G<S) + *1.E- 6+
X1= 0.5000    ,X(100)=  99.50     ,  0.0000E+00 < X <  100.0     , U= 10.
F1= 0.0000E+00,F(100)= 0.0000E+00 ,  -3.500     < F <  3.500     , U=0.50

 #     98   GRAPH(   3/   95)  X : + HOEHE  +   CM   +  F : +DY(RY,G)+ *1.E- 6+
X1= 0.0000E+00,X(100)=  99.00     ,  0.0000E+00 < X <  99.00     , U= 10.
F1= 0.0000E+00,F(100)= 0.0000E+00 ,  -3.500     < F <  3.100     , U=0.50

 #     99   GRAPH(   3/   95)  X : + HOEHE  +   CM   +  F : +E (G)   + *1.E- 3+
X1= 0.5000    ,X(100)=  99.50     ,  0.0000E+00 < X <  100.0     , U= 10.
F1=  44.53    ,F(100)=  48.87     ,  0.0000E+00 < F <  100.0     , U= 10.

              FUNCTIONS     1 -    99 LISTED.
```

Abbildung 4. Katalog-Auszug.

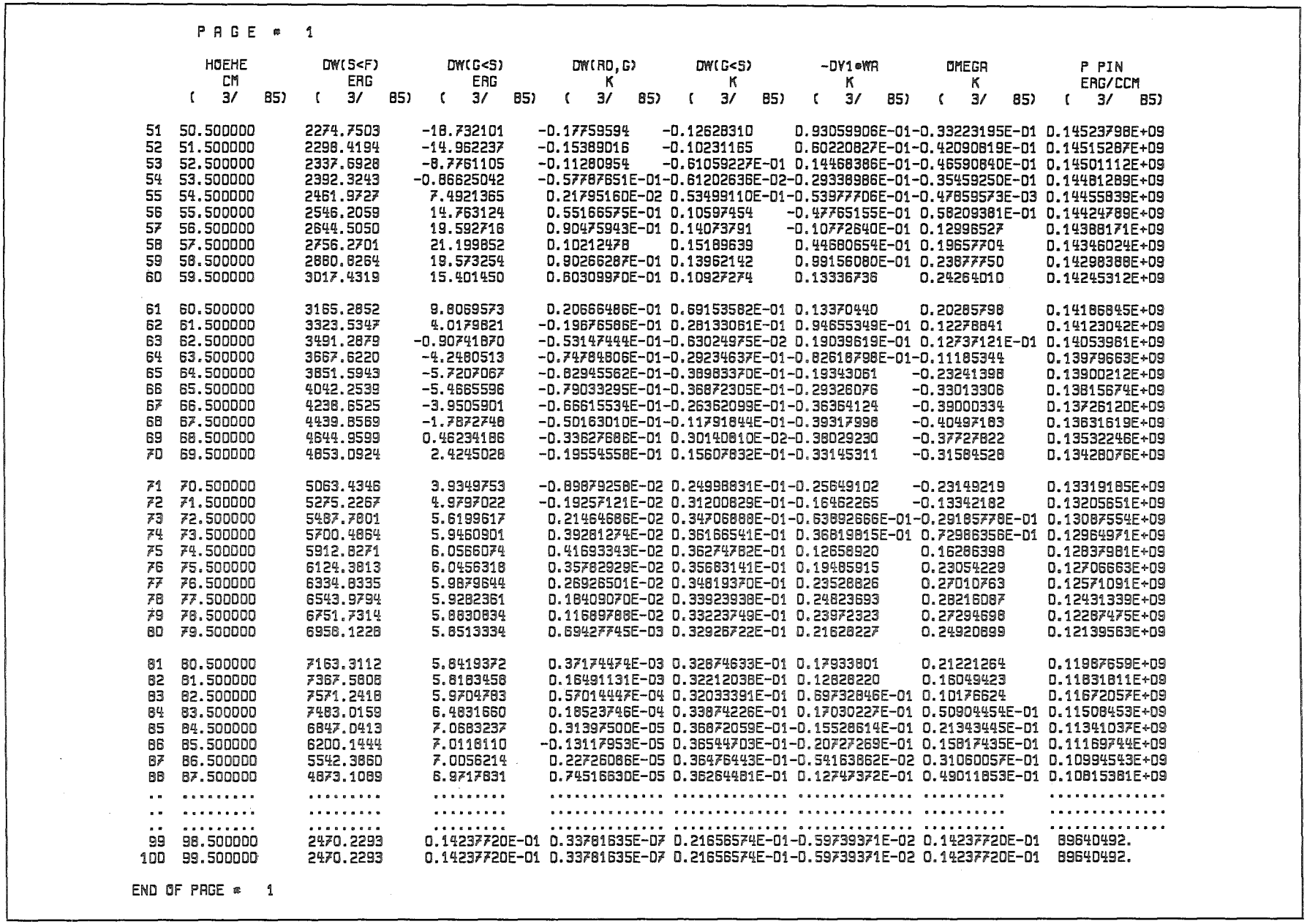


```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                             +
+     21. 4.89     LIST OF THE GRAPHS OF THE DATASET        SCDAA3  .DATA     +
+                                                                             +
+                       STORAGE-NORM IS      "GRAPHIC8"                       +
+                                                                             +
+     DATE OF THE ALLOCATION:    8. 5.83      LAST SCRATCH:      22. 7.83     +
+                                                                             +
+     THE DATASET CONTAINS   216 GRAPHS                TIME: 18H 22' 35"     +
+                                                                             +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

FUNCTIONS TO BE USED :        4     5     3     6    13    14     1

PAGE =   1.     FUNCTIONS USED ON THIS PAGE :        4     5     3     6    13
                                   14     1
```

Abbildung 5. Ausdruck-Spiegel-I.

PAGE # 1

	HOEHE CM (3/ B5)	DW(S<F) ERG (3/ B5)	DW(G<S) ERG (3/ B5)	DW(RD,G) K (3/ B5)	DW(G<S) K (3/ B5)	-DV1*WR K (3/ B5)	OMEGA K (3/ B5)	P PIN ERG/CCM (3/ B5)
1	0.50000000	2470.2293	0.14848984E-01	0.31683518E-07	0.21579710E-01	-0.57410204E-02	0.14848984E-01	89640492.
2	1.5000000	2470.2293	0.14848984E-01	0.31683518E-07	0.21579710E-01	-0.57410204E-02	0.14848984E-01	89640492.
3	2.5000000	2470.2293	4.8128792	0.31683518E-07	0.21579710E-01	-0.57410204E-02	0.15838690E-01	89640492.
4	3.5000000	2147.3559	4.3036059	-0.54984986E-06	0.19853007E-01	-0.20258648E-01	-0.40564087E-03	91354232.
5	4.5000000	1854.6601	4.5328658	-0.16669238E-05	0.21610241E-01	-0.40578645E-01	-0.18968404E-01	94204098.
6	5.5000000	1589.6315	4.4854673	-0.40117483E-05	0.22129569E-01	-0.60933873E-01	-0.38804303E-01	96402027.
7	6.5000000	1350.3041	5.7439081	-0.62139314E-05	0.29673326E-01	-0.71904621E-01	-0.42231295E-01	98558729.
8	7.5000000	2043.5987	6.7680780	-0.13458868E-05	0.35060226E-01	-0.61340373E-01	-0.26280146E-01	0.10057001E+09
9	8.5000000	2779.2999	6.7587089	0.3312117DE-05	0.35050893E-01	-0.3257424 4E-01	0.24766493E-02	0.10253158E+09
10	9.5000000	3493.9357	6.8503968	0.79047019E-05	0.35561728E-01	-0.11989703E-02	0.34362758E-01	0.10444820E+09
11	10.500000	4190.9663	6.9179210	0.98057399E-05	0.35948496E-01	0.16477233E-01	0.52425729E-01	0.10632178E+09
12	11.500000	4873.1089	6.9711096	0.75266890E-05	0.36260993E-01	0.12879560E-01	0.49140553E-01	0.10815381E+09
13	12.500000	5542.3860	7.0052583	0.23118515E-05	0.36474559E-01	-0.53200604E-02	0.31154498E-01	0.10994543E+09
14	13.500000	6200.1444	7.0097696	-0.12799624E-05	0.36534153E-01	-0.20583536E-01	0.15950617E-01	0.11169744E+09
15	14.500000	6847.0413	7.0797057	0.29339222E-05	0.36930900E-01	-0.15888835E-01	0.21042065E-01	0.11341037E+09
16	15.500000	7482.9440	6.5205651	0.17753111E-04	0.34071231E-01	0.15508295E-01	0.48579527E-01	0.11508453E+09
17	16.500000	7570.7308	6.0141751	0.55939715E-04	0.32269865E-01	0.86139167E-01	0.10040903	0.11672057E+09
18	17.500000	7366.9993	5.8312644	0.16437041E-03	0.32283764E-01	0.1277455 7	0.16002933	0.11831811E+09
19	18.500000	7162.7354	5.8411555	0.37180484E-03	0.32870414E-01	0.17824918	0.21211960	0.11987659E+09
20	19.500000	6957.5529	5.8526207	0.69430374E-03	0.32933825E-01	0.21615927	0.24909309	0.12139563E+09
21	20.500000	6751.1679	5.8840481	0.11694298E-02	0.33229390E-01	0.23970335	0.27293274	0.12287475E+09
22	21.500000	6543.4228	5.9293474	0.18421319E-02	0.33930624E-01	0.24831491	0.28224553	0.12431339E+09
23	22.500000	6334.2844	5.9892389	0.26949143E-02	0.34827128E-01	0.23542454	0.27025167	0.12571091E+09
24	23.500000	6123.8404	6.0470604	0.3581328 7E-02	0.35691880E-01	0.19499700	0.23068888	0.12706663E+09
25	24.500000	5912.2952	6.0580528	0.41722459E-02	0.36283785E-01	0.12669160	0.16297536	0.12837981E+09
26	25.500000	5699.9645	5.9473338	0.39295844E-02	0.36174470E-01	0.36877114E-01	0.73051584E-01	0.12964971E+09
27	26.500000	5487.2690	5.6207804	0.21451725E-02	0.34712272E-01	-0.63876491E-01	-0.29164219E-01	0.13087554E+09
28	27.500000	5274.7274	4.9799225	-0.19307024E-02	0.31202493E-01	-0.16464607	-0.13344358	0.13205651E+09
29	28.500000	5062.9480	3.9344770	-0.89972561E-02	0.24995881E-01	-0.25655554	-0.23155966	0.13319185E+09
30	29.500000	4852.6197	2.4232265	-0.19568574E-01	0.15599744E-01	-0.33155080	-0.31595106	0.13428076E+09
31	30.500000	4644.5021	0.46043001	-0.33645496E-01	0.30016405E-02	-0.36039788	-0.37739624	0.13532246E+09
32	31.500000	4439.4151	-1.7893182	-0.50181331E-01	-0.11805412E-01	-0.39326044	-0.40506585	0.13631619E+09
33	32.500000	4238.2278	-3.9519189	-0.66628690E-01	-0.26371147E-01	-0.36367501	-0.39004616	0.13726120E+09
34	33.500000	4041.8473	-5.4663028	-0.79035282E-01	-0.36870812E-01	-0.29324983	-0.33012064	0.13815674E+09
35	34.500000	3851.2071	-5.7184558	-0.82933400E-01	-0.38968272E-01	-0.19339641	-0.23236468	0.13900212E+09
36	35.500000	3667.2551	-4.2440778	-0.74760181E-01	-0.29207467E-01	-0.82582806E-01	-0.1117902 7	0.13979663E+09
37	36.500000	3490.9423	-0.90252481	-0.53115790E-01	-0.62685403E-02	0.19067742E-01	0.12799201E-01	0.14053961E+09
38	37.500000	3323.2115	4.0227674	-0.19645166E-01	0.28166699E-01	0.94676054E-01	0.12284275	0.14123042E+09
..	.....	.....	.....	.....	.....	.....	.....	.....
..	.....	.....	.....	.....	.....	.....	.....	.....
..	.....	.....	.....	.....	.....	.....	.....	.....
49	48.500000	2274.7212	-18.736815	-0.17762695	-0.12631398	0.93052711E-01	-0.33261274E-01	0.14523798E+09
50	49.500000	2266.8343	-19.970591	-0.18506996	-0.13388402	0.10476699	-0.29117029E-01	0.14526636E+09



PAGE # 1

	HOEHE CM (3/ 85)	DW(S<F) ERG (3/ 85)	DW(G<S) ERG (3/ 85)	DW(RD,G) K (3/ 85)	DW(G<S) K (3/ 85)	-DV1*WR K (3/ 85)	OMEGA K (3/ 85)	P PIN ERG/CCM (3/ 85)
51	50.500000	2274.7503	-18.732101	-0.17759594	-0.12628310	0.93059906E-01	-0.33223195E-01	0.14523798E+09
52	51.500000	2298.4194	-14.962237	-0.15389016	-0.10231165	0.60220827E-01	-0.42090819E-01	0.14515287E+09
53	52.500000	2337.6928	-8.7761105	-0.11280954	-0.61059227E-01	0.14468386E-01	-0.46590840E-01	0.14501112E+09
54	53.500000	2392.3243	-0.86625042	-0.57787651E-01	-0.61202636E-02	-0.29338986E-01	-0.35459250E-01	0.14481289E+09
55	54.500000	2461.9727	7.4921365	0.21795160E-02	0.53499110E-01	-0.53977706E-01	-0.47859573E-03	0.14455839E+09
56	55.500000	2546.2059	14.763124	0.55166575E-01	0.10597454	-0.47765155E-01	0.58209381E-01	0.14424789E+09
57	56.500000	2644.5050	19.592716	0.90475943E-01	0.14073791	-0.10772640E-01	0.12996527	0.14388171E+09
58	57.500000	2756.2701	21.199852	0.10212478	0.15189639	0.44680654E-01	0.19657704	0.14346024E+09
59	58.500000	2880.8264	19.573254	0.90266287E-01	0.13962142	0.89156080E-01	0.23877750	0.14298388E+09
60	59.500000	3017.4319	15.401450	0.80309970E-01	0.10927274	0.13336736	0.24264010	0.14245312E+09
61	60.500000	3165.2852	9.8069573	0.20666486E-01	0.69153582E-01	0.13370440	0.20285798	0.14186845E+09
62	61.500000	3323.5347	4.0179821	-0.19676586E-01	0.28133061E-01	0.94655349E-01	0.12278841	0.14123042E+09
63	62.500000	3491.2879	-0.90741870	-0.53147444E-01	-0.63024975E-02	0.19039619E-01	0.12737121E-01	0.14053961E+09
64	63.500000	3667.6220	-4.2480513	-0.74784806E-01	-0.29234637E-01	-0.82618798E-01	-0.11185344	0.13979663E+09
65	64.500000	3851.5943	-5.7207067	-0.82945562E-01	-0.38983370E-01	-0.19343061	-0.23241398	0.13900212E+09
66	65.500000	4042.2539	-5.4665596	-0.79033295E-01	-0.36872305E-01	-0.29326076	-0.33013306	0.13815674E+09
67	66.500000	4238.6525	-3.9505901	-0.66615534E-01	-0.26362099E-01	-0.36364124	-0.39000334	0.13726120E+09
68	67.500000	4439.8569	-1.7872748	-0.50163010E-01	-0.11791844E-01	-0.39317998	-0.40497183	0.13631619E+09
69	68.500000	4644.9599	0.46234186	-0.33627686E-01	0.30140810E-02	-0.38029230	-0.37727822	0.13532246E+09
70	69.500000	4853.0924	2.4245028	-0.19554558E-01	0.15607832E-01	-0.33145311	-0.31584528	0.13428076E+09
71	70.500000	5063.4346	3.9349753	-0.89879258E-02	0.24998831E-01	-0.25649102	-0.23149219	0.13319185E+09
72	71.500000	5275.2267	4.9797022	-0.19257121E-02	0.31200829E-01	-0.16462265	-0.13342182	0.13205651E+09
73	72.500000	5487.7801	5.6199617	0.21464686E-02	0.34706888E-01	-0.63892666E-01	-0.29185778E-01	0.13087554E+09
74	73.500000	5700.4864	5.9460901	0.39281274E-02	0.36166541E-01	0.36819815E-01	0.72986356E-01	0.12964971E+09
75	74.500000	5912.8271	6.0566074	0.41693343E-02	0.36274782E-01	0.12658920	0.16286398	0.12837981E+09
76	75.500000	6124.3813	6.0456318	0.35782929E-02	0.35683141E-01	0.19485915	0.23054229	0.12706663E+09
77	76.500000	6334.8335	5.9879644	0.26926501E-02	0.34819370E-01	0.23528826	0.27010763	0.12571091E+09
78	77.500000	6543.9794	5.9282361	0.18409070E-02	0.33923938E-01	0.24823693	0.28216087	0.12431339E+09
79	78.500000	6751.7314	5.8830834	0.11689788E-02	0.33223749E-01	0.23972323	0.27294698	0.12287475E+09
80	79.500000	6958.1228	5.8513334	0.69427745E-03	0.32926722E-01	0.21628227	0.24920699	0.12139563E+09
81	80.500000	7163.3112	5.8419372	0.37174474E-03	0.32874633E-01	0.17933801	0.21221264	0.11987659E+09
82	81.500000	7367.5808	5.8183458	0.16491131E-03	0.32212038E-01	0.12828220	0.16049423	0.11831811E+09
83	82.500000	7571.2418	5.9704783	0.57014447E-04	0.32033391E-01	0.69732846E-01	0.10176624	0.11672057E+09
84	83.500000	7483.0159	6.4831660	0.18523746E-04	0.33874226E-01	0.17030227E-01	0.50904454E-01	0.11508453E+09
85	84.500000	6847.0413	7.0683237	0.31397500E-05	0.36872059E-01	-0.15528614E-01	0.21343445E-01	0.11341037E+09
86	85.500000	6200.1444	7.0118110	-0.13117953E-05	0.36544709E-01	-0.20727269E-01	0.15817435E-01	0.11169744E+09
87	86.500000	5542.3860	7.0056214	0.22726086E-05	0.36476443E-01	-0.54163862E-02	0.31060057E-01	0.10994543E+09
88	87.500000	4873.1089	6.9717831	0.74516630E-05	0.36264481E-01	0.12747372E-01	0.49011853E-01	0.10815381E+09
..	.....	.....	.....	.....	.....	.....	.....	.....
..	.....	.....	.....	.....	.....	.....	.....	.....
..	.....	.....	.....	.....	.....	.....	.....	.....
99	98.500000	2470.2293	0.14237720E-01	0.33781635E-07	0.21656574E-01	-0.59739371E-02	0.14237720E-01	89640492.
100	99.500000	2470.2293	0.14237720E-01	0.33781635E-07	0.21656574E-01	-0.59739371E-02	0.14237720E-01	89640492.

END OF PAGE # 1

Abbildung 7. Ausdruck-Spiegel-III.

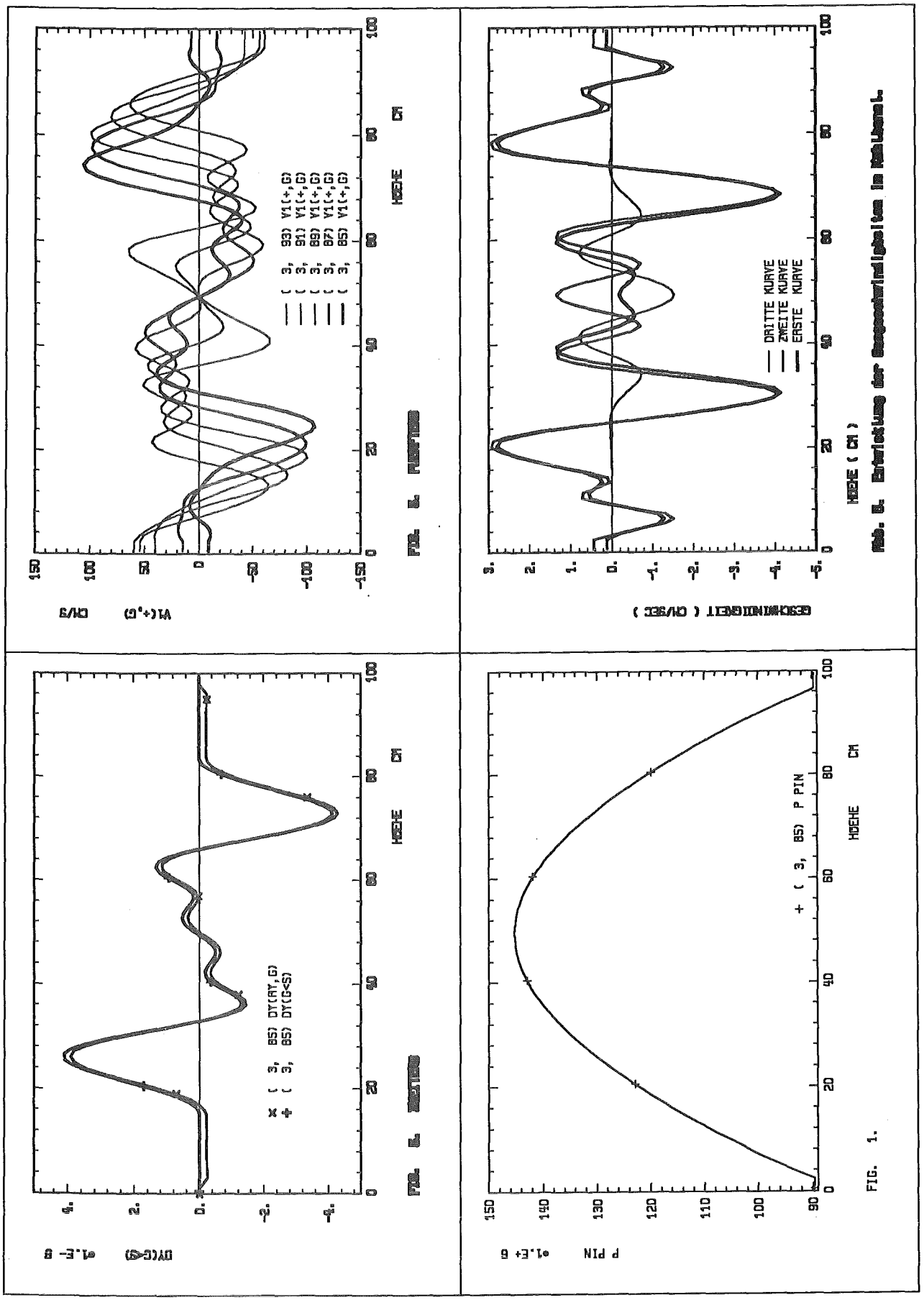


Abbildung 8. Funktionen als Kurvenschar.

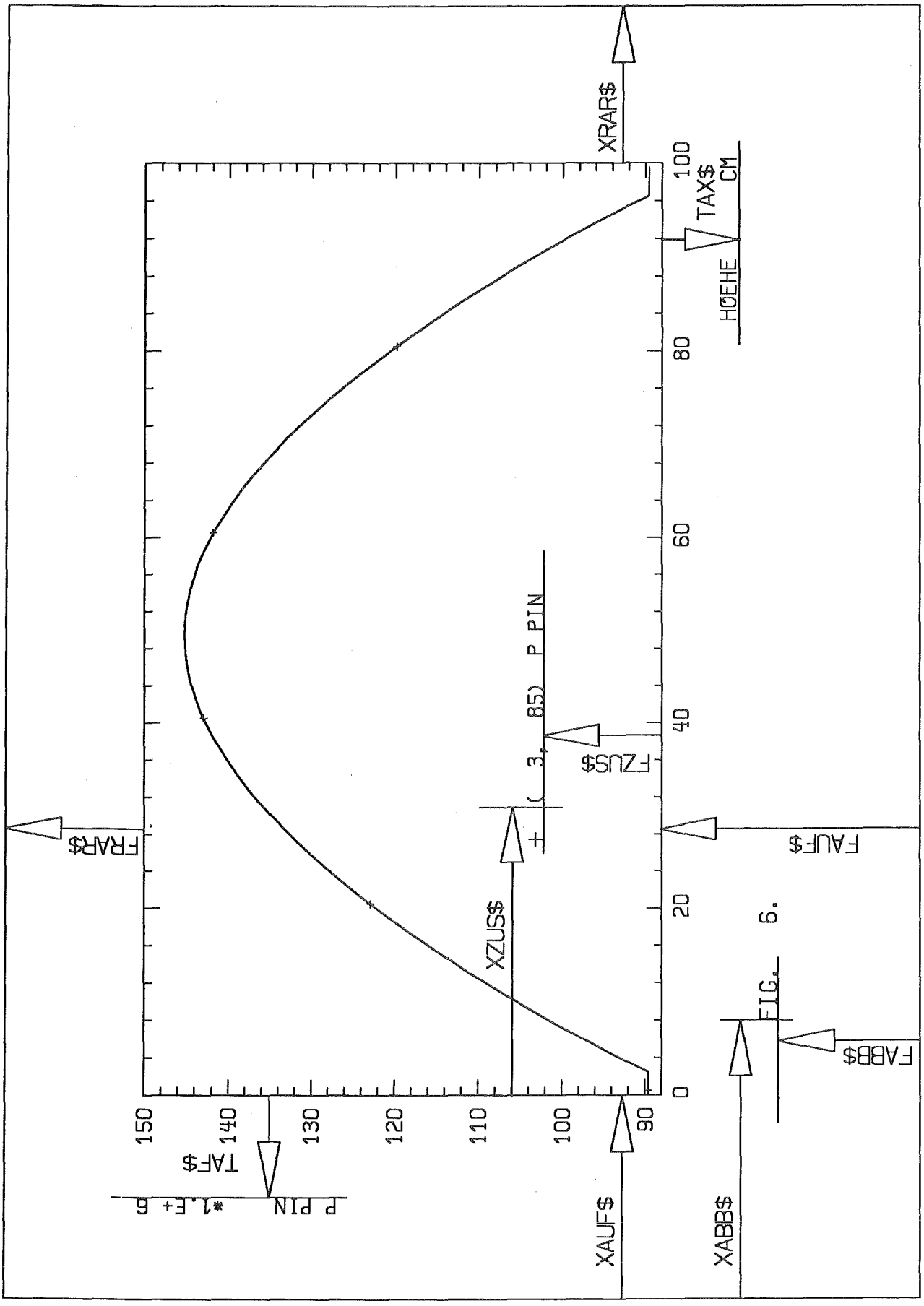


Abbildung 9. Aufbau des Bildrahmens.

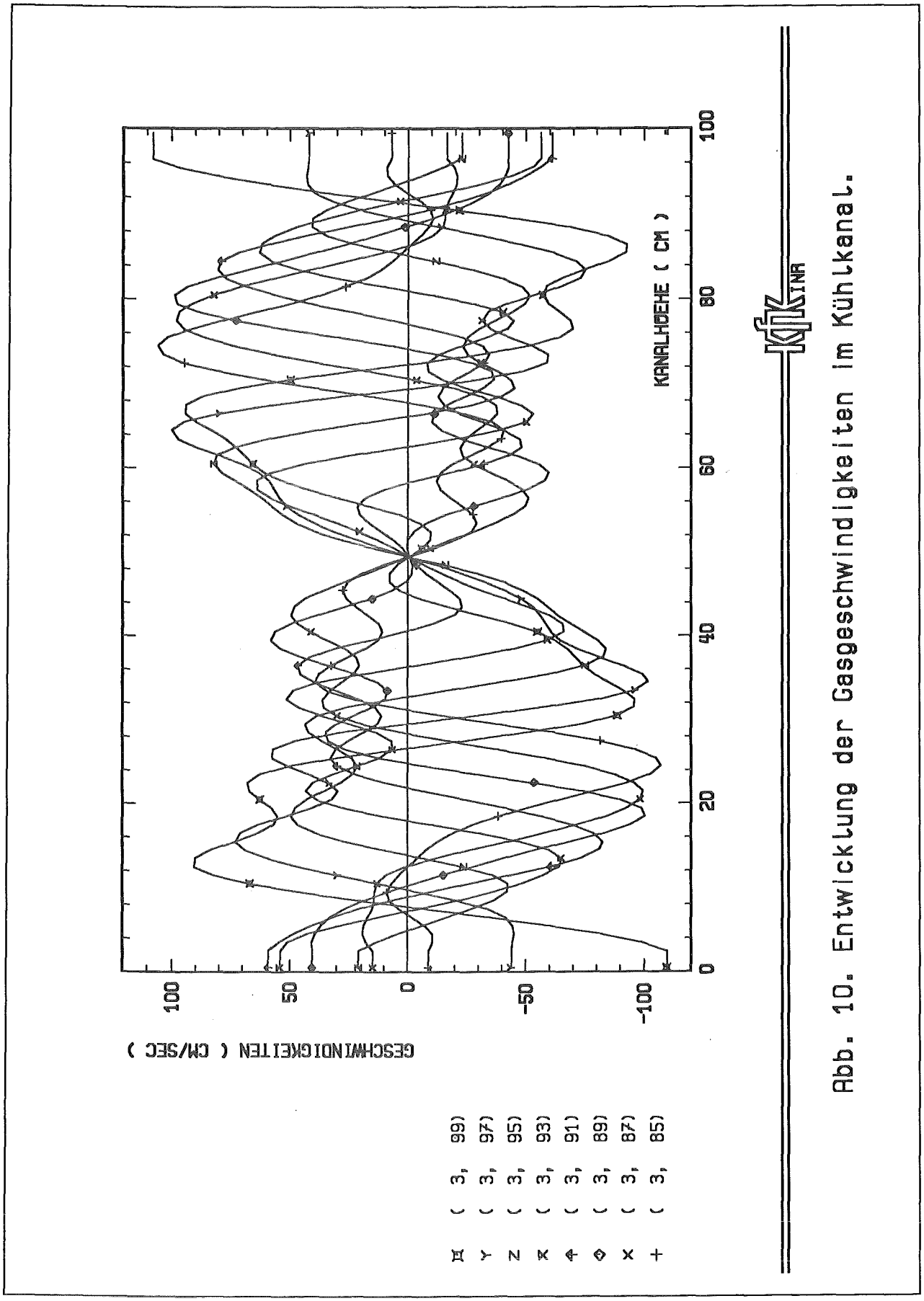


Abbildung 10. Entwicklung der Gasgeschwindigkeiten im Kühlkanal

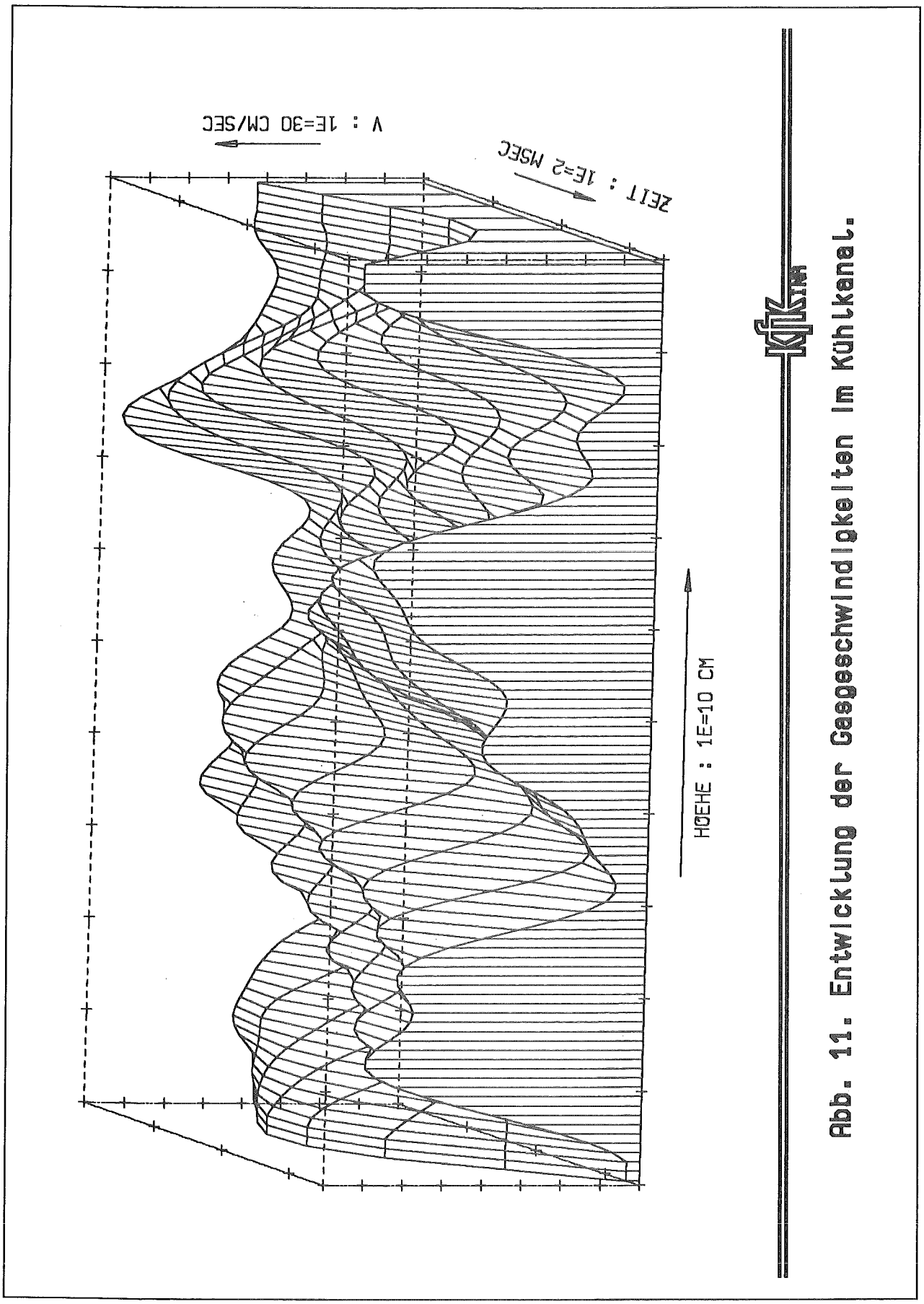


Abbildung 11. Entwicklung der Gasgeschwindigkeiten im Kühlkanal

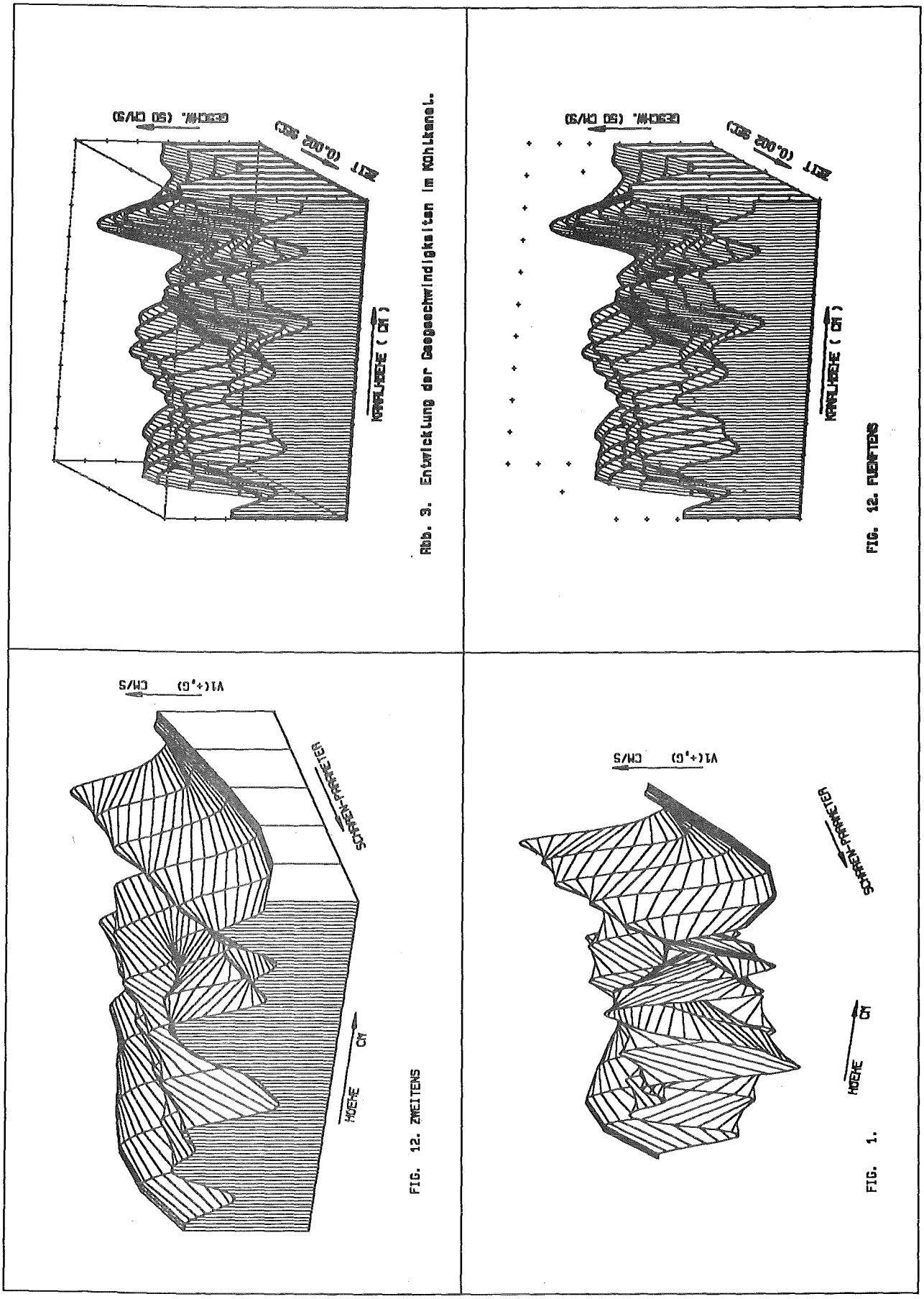


Abbildung 12. Funktionen als Oberfläche I.

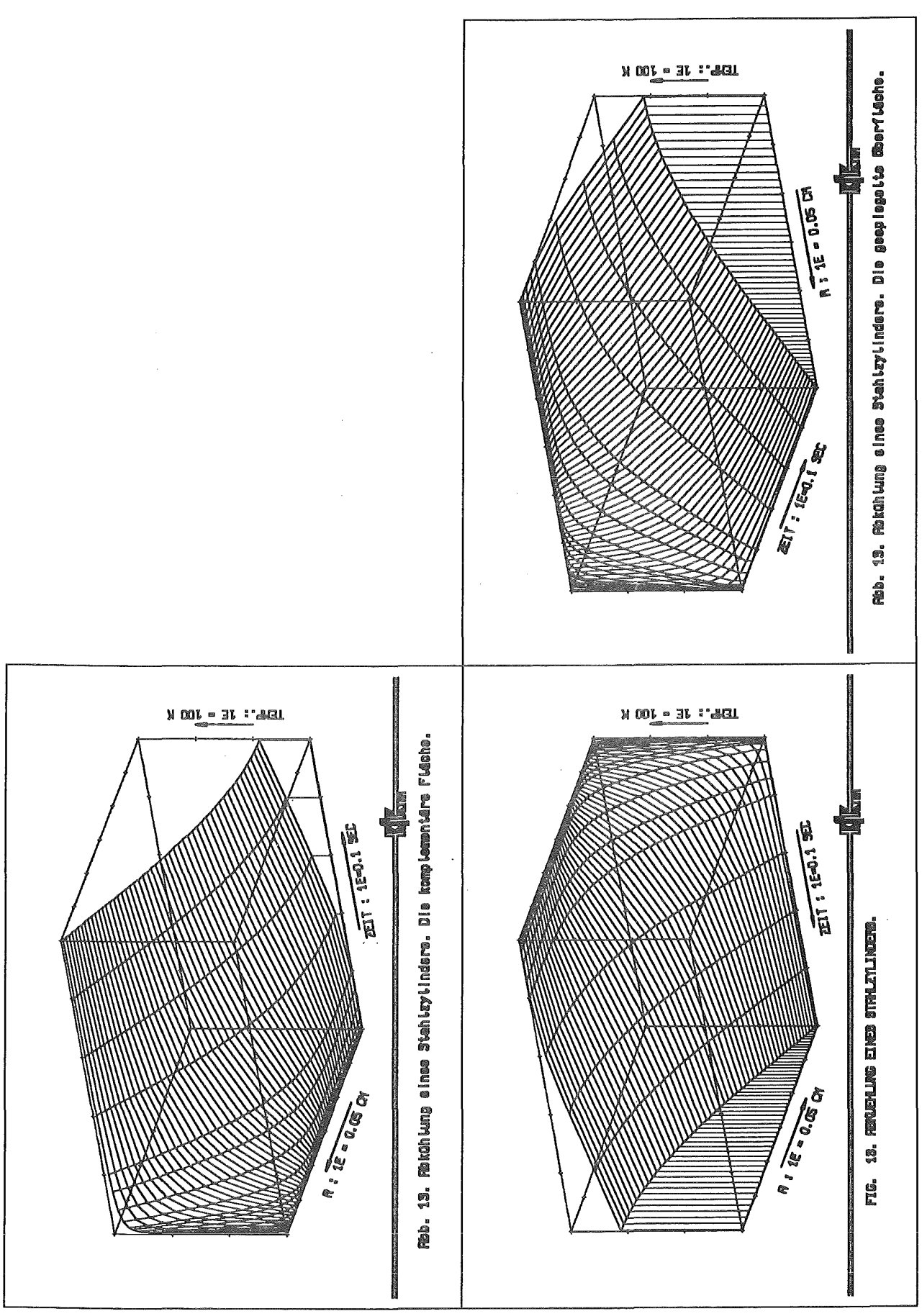


Abbildung 13. Funktionen als Oberfläche II.

```
DATA SET NAME: TSO105.MEMORY

GENERAL DATA:                          CURRENT ALLOCATION:
   Volume serial:         TSO00C          Allocated tracks:        1
   Device type:           3380            Allocated extents:       1
   Organization:          PO              Maximum dir. blocks:     5
   Record format:         FB
   Record length:         80          CURRENT UTILIZATION:
   Block size:            3120            Used tracks:             1
   1st extent tracks:     1               Used extents:            1
   Secondary tracks:      1               Used dir. blocks:        2


LIBRARY - TSO105.MEMORY    MEMBERS

       NAME       PROCEDURE

       ABB1       "FIGUR" & "DDFIG" & "TEXIMA"
       ABB2       "FIGUR" & "DDFIG" & "TEXIMA"
       ABB3       "FIGUR" & "DDFIG" & "TEXIMA"
       ABB4       "FIGUR" & "DDFIG" & "TEXIMA"
       BRAUS      "BRAUS"
       CORE       "CORE"
       DATAS      "DATAS" & "ENDE" & "FUNKIN"
       DRUDA      "DRUDA"
       FEJ        "FEJ"
       FIGUR      "FIGUR" & "DDFIG"
       IMMANI     "IMMANI"
       KOPIER     "KOPIER"
       LADDFI     "DDFIG"
       LAFIGU     "FIGUR"
       LATEXI     "TEXIMA"
       LOESCH     "LOESCH"
       PICOR      "PICOR"
       BOOKS      "BRAUS" "IMMANI" "PIXI" "PODEL" & "PONUM"
       PONUM      "PODEL" & "PONUM"
       SERVIN     "FIGUR" & "DDFIG" & "TEXIMA"
       TEXIMA     "TEXIMA"
```

Anhang B. Aufbau der Datei INR105.SERVUS.LOAD

1. Glied : IDAJOB
   Bestimmung : Eine SERVUS-Datei wird neu angelegt oder umbenannt.
   Enthält die Routine : IDAJOB

2. Glied : ENDE
   Bestimmung : Schreibt eine Endzeile in eine SERVUS-Datei
   Enthält die Routine : ENDE

3. Glied : FUNKIN
   Bestimmung : Erzeugt und schreibt Funktionen in eine SERVUS-Datei
   Enthält die Routine : FUNKIN

4. Glied : SERDIO
   Bestimmung : Schreibt Funktionen in eine Datei ("SDIN") bzw. liest Funktionen einer Datei ("SDEX").
   Enthält die Routinen :

   SERDIO
   SDINPY
   SDING4
   SDING8
   SDINF8
   SDEXPY
   SDEXG4
   SDEXG8
   SDEXF8
   PDEXPY
   SDTEXT
   SERNEX
   SERNIN
   NOPA

5. Glied : DATAS
   Bestimmung : Eine Datei wird im Dialogbetrieb gelesen/kopiert.
   Enthält die Routinen :

   DATAS
   NO

6. Glied : COPYDA
   Bestimmung : Eine Datei wird kopiert oder ausgedruckt, Funktionen einer Datei werden gelöscht, zusammengefügt, neugeordnet, transponiert, mit neuen Kenngrößen versehen, in ihren dekadischen Logarithmen umgerechnet , usw.
   Enthält die Routinen :

   COPYDA
   PAGEFF
   ERASE
   UNION
   MIX
   INVERT
   TRANSP
   NEWNUM
   LOGFUN

7. Glied : NEXT

Bestimmung : Sucht aus die nächste Funktion, Verteilt die Ausgabe auf mehreren Seiten.
Enthält die Routinen :

NEXT
CIFFRA
PAGINA
ORDNE
MONOT4
MONOT8
ISPER

8. Glied : FENST4
Bestimmung : Korrigiert die Funktionen beim Kopieren.
Enthält die Routinen :

FENST4
FENST8
MASTR4
MASTR8
VALS4
VALS8
CIRCA

9. Glied : FIGUR
Bestimmung : Funktionen einer Datei werden als Kurven gezeigt (Dialogbetrieb).
Enthält die Routine : FIGUR

10. Glied : FIGURB
Bestimmung : Funktionen einer Datei werden als Kurven gezeigt (Stapelbetrieb).
Enthält die Routine : FIGURB

11. Glied : PAPIER
Bestimmung : Papieraufteilung, Netzherstellung, Achsengestaltung bei der Bildherstellung.
Enthält die Routinen :

PAPIER
FENCO
AXON
CORDAT
FUNKK

12. Glied : CORTEX
Bestimmung : Textkorrektur im Dialogverfahren, Eingabeübernahme von einer Kommandoprozedur, Funktionsausgabe am Bildschirm, Beginn und Ende eines Bildes.
Enthält die Routinen :

CORTEX
NESZE
LEVEL
TABUL
FUTTER
PINIT
FINIP
GSNAME

JELE
FORMEX

13. Glied : DDFIG
Bestimmung : Funktionen einer Datei werden als Fläche gezeigt (Dialogbetrieb).
Enthält die Routine : DDFIG

14. Glied : DDFIGB
Bestimmung : Funktionen einer Datei werden als Fläche gezeigt (Stapelbetrieb).
Enthält die Routine : DDFIGB

15. Glied : WUERFL
Bestimmung : Hilfsroutinen für die Flächen-Darstellungen .
Enthält die Routinen :

WUERFL
MIRROR
AREA
DDPLO1
SICHT
STUECK
AXTEX
FAULA
DDAXIS
KANTE
MARKI

16. Glied : TEXIMA
Bestimmung : Texte werden zur GS-Bilder verarbeitet. (Dialogbetrieb).
Enthält die Routine : TEXIMA