KfK 3803
August 1984

Anleitung zur Benutzung des Programmsystems SEDAP

F. Katz, W. Olbrich
Institut für Reaktorentwicklung

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Reaktorentwicklung

KfK 3803

Anleitung zur Benutzung des Programmsystems SEDAP

F. Katz
W. Olbrich

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

Zusammenfassung

SEDAP (System for Experimental Data Processing) ist ein vielfältig einsetzbares Programmsystem zur Verarbeitung und Reduktion experimentell gewonnener Daten. SEDAP wurde in FORTRAN 77 programmiert.

Das SEDAP-Handbuch enthält alle notwendigen Informationen, die zur Anwendung von SEDAP erforderlich sind. Es umfaßt eine vollständige Beschreibung aller dem Anwender zur Verfügung stehenden Fähigkeiten des Systems mit Beispielen und Fehlermeldungen.

SEDAP - User Manual

Abstract

SEDAP (System for Experimental Data Processing) provides the scientist with a powerful tool to process various digital data which are sampled during an experiment. SEDAP is a software package based upon FORTRAN 77.

The SEDAP-Manual contains all informations that are necessary for the application of SEDAP. It gives a complete description of all user accessible system functions, examples and error messages.

INHALTSVERZEICHNIS

ALLGEMEINE EINFUEHRUNG

Die vorliegende Programmbeschreibung bezieht sich auf den Stand des Programmes am Tagesdatum der Titelseite. Sie ist nur auf eine reine Benutzung des Systems ausgerichtet. Erklärungen über die Organisation des Programmes erfahren Sie, wenn Sie das Kapitel SEDAP ORGANISATION oder eine der folgenden Veröffentlichungen lesen.

1. KFK-Nachrichten 3/71, SEDAP rechnergestützte Auswertung von Experimenten
2. Paper IFIPS-Congress 71, SEDAP a Systematic Approach to the Processing of Experimental Data.
3. KFK-Bericht Nr. 1594 SEDAP (Audoux, Katz, Olbrich, Schlechtendahl)

lesen.

Einige Bemerkungen bezüglich der Datenauswertung. Um diesen Prozeß an der IBM sinnvoll durchzuführen, sind auf den nächsten Seiten die dazugehörigen Job Control Karten kurz beschrieben. Bei Anwendung von SEDAP auf anderen Rechnern müssen die Programme selbstverständlich der dort installierten Job Control Sprache angepaßt werden. Das compilierte Programm befindet sich auf einem Datenträger und kann mit Hilfe der beschriebenen Steuer-Karten aufgerufen werden. Lesen Sie bitte auch das Kapitel LEISTUNGSGRENZEN VON SEDAP in diesem Handbuch genau durch.

Sollte während der Auswertung ein Fehler auftreten, so kommt, wenn er auf der Programmebene liegt, eine Fehlerdiagnostik, die im Klartext den Fehler beschreibt. Der Benutzer kann dann meistens ohne Rückfrage seine Befehle berichtigen und einen neuen Job starten. Es ist aber auch möglich, einen Fehler zu erhalten, der schwerwiegender Natur ist, wie z.B. die Completion Codes 0C4, 0C5 oder D37. Wenn dies der Fall ist, bitten wir den Benutzer zunächst nicht die Programmberatung aufzusuchen, sondern mit der fehlerhaften Liste zu einem erfahrenen SEDAP-Anwender oder zu einem der Verfasser zu kommen. Gleiches gilt auch für die Erstellung eventuell nötiger Anschlußprogramme.

SEDAP ORGANISATION

Die Erläuterungen in diesem Kapitel beziehen sich auf die graphische Darstellung der SEDAP Organisation im Anhang dieses Handbuches.

SEDAP enthält einen Steuerteil (BOSS), der die eingegebenen Befehle liest, interpretiert und die Ausführung der Befehle steuert. Die von SEDAP bearbeiteten Dateien, die im allgemeinen Messignale , sog. Meßrecords, repräsentieren, werden aus Gründen der Effektivität auf einem Direktzugriffsspeicher verwaltet (LAGER). Informationen über diese Meßrecords (Blockzahl, Frequenz usw.) werden in Tabellen eingetragen (DEPOT).

SEDAP enthält verschiedene, auf modularer Basis aufgebaute Routinen zur Ausführung der Befehle (OPERATOREN). Dateneingabe für SEDAP ist das DUMP-Format. Außerdem können bestimmte Meßsignale (Sinus, Kosinus, Konstante, Viereckimpuls usw.) erzeugt werden (DAGEN). Ausgabemöglichkeit ist die Druck- oder die Plot-Ausgabe.

Ferner können die Meßrecords (LAGER) mit ihren spezifischen Informationen (DEPOT) auf Band oder Platte gerettet werden (DUMP).

LEISTUNGSGRENZEN VON SEDAP

SEDAP unterliegt folgenden FORTRAN- und programmierspezifischen Leistungsgrenzen:

a) Während einer SEDAP-Ausführung sind max. 512 verschiedene Meßrecords zu verarbeiten (Tabellengröse von DEPOT).
b) Ein Befehl kann max. 99999 Blöcke a 512 Werten verarbeiten.
c) Als Raffungsfaktor kann ein Wert von 1 bis 100 angegeben werden.

Abweichungen von b) und c) sind bei den jeweiligen Befehlen angegeben.

IBM-JOB-CONTROL-KARTEN

SEDAP ist katalogisiert und kann durch folgende Job-Control-Karten aufgerufen werden:

1. Fall: HOLE (DUMP) Beispiel

```
//  EXEC  SEDAP
//G.FT02F001 DD UNIT=1600,VOL=SER=DV0999,LABEL=(1,SL),DSN=DATA1,
//     DISP=(OLD,PASS),DCB=(BLKSIZE=3303,RECFM=VBS)
//G.SYSIN  DD  *
         SEDAP Eingabekarten
//*
```

Bei einem DUMP-Lauf ist nur der DISP-Parameter in DISP=(NEW,PASS) zu ändern.
IM Kapitel SEDAP-Anschlußprogramm wird gezeigt, wie eine für SEDAP lesbare Datei erzeugt werden kann.

2. Fall: SEDAP Lauf mit Plotausgabe auf dem Versatec-Plotter

```
//  EXEC   SEDAP,PLOT=VER
//G.SYSIN  DD  *
       SEDAP Eingabekarten
// EXEC SVPLOT
```

weitere Möglichkeiten:

```
//  EXEC   SEDAP,PLOT=AGF,PLFILE=dsname      AGF-Plotfile
//  EXEC   SEDAP,PLOT=XYN,BAND=xnnnjn        Xyneticsplot
//  EXEC   SEDAP,PLOT=STA,BAND=xnnnjn        Statosplot
           dsname muß katalogisiert sein
           xnnnjn siehe IBM-Benutzerhandbuch
```

3. Fall: Anschluß eines EXTEND-Programmes an SEDAP

```
//STEP1  EXEC  F7C
//C.SYSIN  DD  *
          EXTEND Subroutine
//*
//STEP2  EXEC  SEDAP
//L.SYSIN  DD  DSN=&&LKSET,DISP=(OLD,DELETE)
//G.SYSIN  DD  *
         SEDAP Eingabekarten
//*
```

4. Fall: SEDAP Lauf mit Meßdaten auf einem 9 Spur-Magnetband,
wie sie im IRE erfaßt werden.

```
//  EXEC   SEDAP
//G.FT21F001 DD UNIT=T1600,LABEL=(1,SL),DSN=Filename,
//           DISP=(OLD,PASS),VOL=SER=Bandname
//G.SYSIN  DD  *
      SEDAP Eingabekarten
//*
```

Nachfolgende Auflistung gibt die katalogisierte SEDAP Prozedur wieder:

```
//*********************************************************************
//SEDAP    PROC PLOT=VER,PLFILE=NULLFILE,BAND=NULLFI,BLOCK=5000
//         DISK=SYSDA,LIB=HARWELL
//L        EXEC PGM=IEWL,COND=(4,LT),PARM='LIST,MAP'
//SYSLIB   DD   DSN=SYS7.FORTLIB,DISP=SHR
//         DD   DSN=SYS2.FORTLIB,DISP=SHR
//         DD   DSN=LOAD.IRE,DISP=SHR
//         DD   DSN=SYS2.&LIB,DISP=SHR
//SYSLIN   DD   DSN=DATA.IRE(SEDAP77),DISP=SHR
//         DD   DSN=DATA.IRE(SED&PLOT),DISP=SHR
//         DD   DDNAME=SYSIN
//LOAD     DD   DSN=LOAD.IRE,DISP=SHR
//AGF      DD   DSN=LOAD.IRE,DISP=SHR
//CALCOMP  DD   DSN=SYS2.CALCOMP,DISP=SHR
//STATOS   DD   DSN=SYS2.STATOS,DISP=SHR
//XYNETICS DD   DSN=SYS2.XYNETICS,DISP=SHR
//VERSATEC DD   DSN=SYS2.VERSATEC,DISP=SHR
//SYSUT1   DD   UNIT=&DISK,SPACE=(3303,(150)),DCB=BLKSIZE=3303
//SYSLMOD  DD   DSN=&&GOSET(MAIN),UNIT=&DISK,DCB=BLKSIZE=3303,
//         SPACE=(3303,(150,,1),RLSE),DISP=(,PASS)
//SYSPRINT DD   SYSOUT=N
//G        EXEC PGM=*.L.SYSLMOD,COND=(5,LT)
//FT01F001 DD   DSN=TSO745.SEDAP.USER.DATA,DISP=SHR
//FT05F001 DD   DDNAME=SYSIN
//FT06F001 DD   SYSOUT=A,DCB=(BLKSIZE=1995,LRECL=133,RECFM=FBA)
//FT08F001 DD   UNIT=&DISK,SPACE=(TRK,(50,10)),
//         DCB=(BLKSIZE=9442,LRECL=32000,RECFM=VBS)
//FT09F001 DD   DSN=&PLFILE,DISP=SHR
//FT15F001 DD   UNIT=&DISK,DISP=(NEW,DELETE),SPACE=(1680,(20,1)),
//         DCB=(BLKSIZE=1680,LRECL=80,RECFM=FB)
//FT40F001 DD   UNIT=&DISK,DISP=(NEW,DELETE),SPACE=(2048,(&BLOCK))
//PLOTTAPE DD UNIT=T0800,LABEL=(,NL),DSN=&BAND.LE,DCB=DEN=2,
//         VOL=(,RETAIN,SER=(&BAND))
//PLOTWK01 DD   UNIT=&DISK,SPACE=(TRK,(9,19))
//PLOTWK02 DD   UNIT=&DISK,SPACE=(TRK,(9,19))
//PLOTLOG  DD   SYSOUT=A
//VECTR1   DD   UNIT=&DISK,DISP=(MOD,PASS),SPACE=(CYL,(10,5)),
//         DSN=&&VECTR1
//VECTR2   DD   UNIT=&DISK,DISP=(MOD,PASS),SPACE=(CYL,(10,5)),
//         DSN=&&VECTR2,DCB=BLKSIZE=18000
//SEDAP    PEND
//*********************************************************************
```

Sollte aus irgend welchen Gründen eine oder mehrere dieser DD-Karten überschrieben werden, so ist darauf zu achten, daß die Reihenfolge und die Lage der überschriebenen Karten von Wichtigkeit ist.

Die FT40 legt eine temporäre Direct-Access-Datei für das Lager an, deren Initialisierung an der IBM etwa 60 DM kostet. Falls genügend Platz vorhanden ist, ist es zu empfehlen diese Datei permanent anzulegen. (Bei 5000 Blöcken circa 500 Spuren). Siehe hierzu Befehl SEDA.

DIE SEDAP BEFEHLSSTRUKTUR

TITEL-KARTEN

Die ersten vier Karten, die SEDAP in der Programmeingabe erwartet und welche immer vorhanden sein müssen, sind:

Karte 1 - Systemidentifikation
Karte 2 - erste Kommentarkarte
Karte 3 - zweite Kommentarkarte
Karte 4 - Lagerinitialisierung

1. Karte = Titelkarte:

in den Spalten 1 bis 5 muß das Wort SEDAP stehen. Dann folgt von Spalte 11 bis 18 ein beliebiger Text, der vom Benutzer angegeben wird. Dieser Text wird im Großformat an den Anfang der Ausgabeliste geschrieben.

Beispiel:

```
SEDAP      NSK-TEST
```

2.+3.Karte = Kommentarkarten:

Diese Karten ermöglichen dem Benutzer, seinen Auftrag im Klartext zu dokumentieren, um später evtl. Anhaltspunkte über Versuchsart, Datum des Versuches und seines Zeitpunktes zu haben. Der Inhalt beider Karten erscheint unter dem Großtext auf Seite 1 der Ausgabeliste. Diese Kommentarkarten müssen immer vorhanden sein, da sonst das Programm die nachfolgenden Befehlskarten für Kommentare ansieht. Wenn kein Text gewünscht wird, sollten hier also zwei Leerkarten stehen.
Diese Karten dürfen in der Spalte 1 nicht mit dem Zeichen > beginnen.

Beispiel:

```
NSK-Versuch vom 16.November 1971
Die Daten stehen auf dem Magnetband NSK, File 3
```

4. Karte = Initialisierungskarte für das Lager:

Diese Karte muß unbedingt an dieser Stelle stehen. Eine genaue Beschreibung dieser Karte siehe Befehl SEDA

Nun folgen die Befehlskarten

.
.
.
.
.

FORMAT DER BEFEHLSEINGABE

Die Befehle werden im Kartenformat eingegeben und haben folgenden Aufbau:

Bef	Nam1	Nam2	Nam3	I1	I2	I3	X1	X2	X3
____	____	____	____	____	____	____	________	________	________

Erklärungen:

Bef	Spalte 1- 4	Hier wird einer der im Handbuch beschriebenen Befehle angegeben.
Nam1	Spalte 11-14,	
Nam2	Spalte 16-19,	
Nam3	Spalte 21-24	kennzeichnen die Namen der zu verarbeitenden Dateien, der neu erzeugten Dateien, sowie Modifikationen von Befehlen.
I1	Spalte 26-30,	
I2	Spalte 31-35,	
I3	Spalte 36-40	sind Integerzahlen,die rechtsbündig eingegeben werden. I1 und I2 kennzeichnen Anfang und Ende des zu bearbeitenden Dateibereichs, die Bedeutung von I3 hängt vom Befehl ab.
X1	Spalte 41-50,	
X2	Spalte 51-60,	
X3	Spalte 61-70	sind Gleitkommazahlen. Ihre Bedeutung hängt vom Befehl ab.
	Spalte 71-80	nicht benutzt.

ALPHABETISCHE LISTE DER BEFEHLE

Befehl	Kurzerklärung
ADDI	Addieren zweier Dateien
AX+B	Linearverschiebung
BEFA	Freq. Spektrum in Polar-Koordinaten
BILD	Lagerzustand
DAGE	Erzeugung von Daten und Funktionen
DEFX	Definieren einer X-Achse
DEFY	Definieren einer Y-Achse
DIFF	Differenzieren einer Datei
DIKO	Komplexe Division
DIVI	Dividieren zweier Dateien
DUMP	Retten von Daten auf externem Datenspeicher
FANA	Fourier-Analyse
FANT	Antitransformation
FILI	Smoothing Verfahren erster Ordnung 3 Punkte
FIL2	Smoothing Verfahren erster Ordnung 5 Punkte
FIL3	Smoothing Verfahren dritter Ordnung
FIL4	Variables Filter erster Ordnung
FOUT	Fourier Transformation
HAFU	Hanning Funktion
HOLE	Holen einer im SEDAP Format beschriebenen Datei
INSI	Integration nach Simpson
INSW	Integration mit Switch
INTR	Integration nach der Trapez-Regel
KETT	Verketten von Dateien
KOKO	Konjugiert komplexe Multiplikation
LEDI	Ermittlung der spektralen Leistungsdichte
LOGA	Logarithmieren von Dateien
MUKO	Komplexe Multiplikation
MULT	Multiplizieren zweier Dateien
MWEF	Mittelwertbildung einer Datei
MWES	Es wird der errechnete Mittelwert abgezogen
NGET	Holen von mit der NOVA-Anlage erzeugten Daten
NOPR	Setzen der NO-PRINT Option
PBHE	Print Block horizontal im E-Format
PBHF	Print Block Horizontal im F-Format
PBVE	Print Block vertikal im E-Format
PBVF	Print Block vertikal im F-Format
PLOT	Plotten einer Datei
POTI	Potenzieren der Werte von Dateien
PRNT	Setzen der PRINT Option
RENA	Umbenennung einer Datei

Befehl	Kurzerklärung
SEDA	Dimensionierung des Programmes
S002	Sortieren 2
S004	Sortieren 4
S008	Sortieren 8
S016	Sortieren 16
S032	Sortieren 32
S064	Sortieren 64
SUBT	Subtraktion zweier Dateien
STOP	Ende der Eingabedaten
TWOR	Umrechnung von Thermospannungen in Celsius
WERT	Adressierung von Einzeldaten (Index|Zeit)
XTSD	Benutzerroutine (SEDAP Erweiterung)
ZERS	Löschen einer Datei
ZUST	Liste aller z.Z. vorhandenen Befehle

LISTE DER MODIFIER

KONS - benutzt bei dem Befehl DAGE Generierung einer Konstanten
AX+B - benutzt bei dem Befehl DAGE Generierung einer Rampe
SINF - benutzt bei dem Befehl DAGE Generierung einer Sinusfunktion
COSF - benutzt bei dem Befehl DAGE Generierung einer Cosinusfunktion
VIER - benutzt bei dem Befehl DAGE Generierung einer Viereckfunktion
RAND - benutzt bei dem Befehl DAGE Generierung einer Funktion mit Hilfe von Zufallszahlen
ZEIT - benutzt bei dem Befehl WERT Angabe der Begrenzung in Zeiteinheiten

DUMP - benutzt bei dem Befehl SEDA falls Fehler auftraten, Dump auf Band
TEMP - benutzt bei dem Befehl SEDA temporäres Lager
INIT - benutzt bei dem Befehl SEDA neues, stationäres Lager
ALT* - benutzt bei dem Befehl SEDA altes, stationäres Lager
ALT* - benutzt bei dem Befehl PLOT es soll auf vorhandenem Plot eine weitere Kurve gezeichnet werden
GRID - benutzt bei dem Befehl PLOT es soll ein zusätzliches Gitter in das Bild gezeichnet werden
ALLE - benutzt bei den Befehlen DUMP HOLE BILD ZERS

MOD1 - kann bei dem Befehl XTSD benutzt werden
MOD2 - kann bei dem Befehl XTSD benutzt werden
MOD3 - kann bei dem Befehl XTSD benutzt werden
MOD4 - kann bei dem Befehl XTSD benutzt werden

ADDI

```
**************
*            *
*    ADDI    *
*            *
**************
```

addiert zwei Dateien und speichert die Ergebnisdatei in das Lager

Befehlsform:

```
ADDI       DATA DATB DATC   IB1  IB2   IR
____       ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    . |
```

DATA ist der Name der ersten Datei, welche addiert werden soll
DATB ist der Name der zweiten Datei, welche verarbeitet werden soll
DATC ist der Name der Ergebnisdatei nach der Addition (C=A+B)
IB1 ist der erste Block des verlangten Dateisegmentes
IB2 ist der letzte Block des verlangten Dateisegmentes
IR ist der Raffungsfaktor mit dem die Inputwerte gerafft werden sollen

Beispiel:

```
ADDI       CH22 CH23 TEMP     1    6    2
____       ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    . |
```

Addiere die 6 Blöcke (1 bis 6) der Datei CH22 zu den 6 Blöcken der Datei CH23 und speichere die Ergebnisdatei, welche 3 Blöcke groß ist, da IR = 2, unter dem Namen TEMP in das Lager.

Bemerkung:

ADDI kann auch komplexe Werte addieren.

AX+B

```
**************
*            *
*    AX+B    *
*            *
**************
```

ermöglicht die Linearverschiebung einer Datei gemäß der Gleichung Y = AX + B und speichert das Ergebnis in das Lager.

Befehlsform:

```
AX+B      DATX      DATY   IB1  IB2   IR     A      B
____      ____ ____ ____ *   |*   |*   |*    .   |*   .   |*   . |
```

DATX	ist der Name der Eingabedatei (X in Gleichung)
DATY	ist der Name der Ergebnisdatei (Y in Gleichung)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit dem die Inputwerte gerafft werden sollen
A	ist der Koeffizient A der Gleichung
B	ist der Koeffizient B der Gleichung

Beispiel:

```
AX+B      CH11      NC11     1   10    5   10.0       3.0
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

multipliziert die ersten 10 Blöcke der Datei CH11 mit dem Faktor 10.0 und addiert den Wert 3.0 zum Produkt. Dann wird die Summe von zwei Blöcken, da IR = 5, in das Lager unter den Namen NC11 abgespeichert.

Bemerkung:

ist A = 0.0, dann erzeugt AX+B eine Datei mit den konstanten Werten von B
ist B = 0.0, dann multipliziert AX+B eine Konstante mit allen Werten der Datei
ist A = 1.0, dann addiert AX+B eine Konstante zu allen Werten der Datei.
ist A = 1.0 und B = 0.0, so wird von dieser Datei eine Kopie gemacht, ohne daß die Transfer-Routinen aktiv werden. Durch Angabe eines Raffungsfaktors kann man somit eine Datei aufteilen.

BEFA

```
**************
*            *
*    BEFA    *
*            *
**************
```

transformiert die mit der komplexen Fouriertransformation berechneten Spektren von kartesischen Koordinaten (real/imaginär) in Polarkoordinaten (Betrag-Phase)

Befehlsform:

```
BEFA       KOSP       BEPH    IB1  IB2
____       ____ ____  ____ *   |*   |*   |*    .   |*    .   |*    . |
```

KOSP	ist der Name der Eingangsdatei
BEPH	ist der Name der Ergebnisdatei. Sie enthält das Spektrum in Betrag-Phase Darstellung
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

```
BEFA       KOSP       BEFH      1   10
____       ____ ____  ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Berechnet für die ersten 10 Blöcke der Datei KOSP Betrag und Phasenwerte und speichert sie unter dem Namen BEPH paarweise ins Lager.

Bemerkung:

Nach dem Befehl BEFA sollte immer der Befehl SORTIERE mit SO02 folgen, wenn die Werte in einem PLOT dargestellt werden sollen.
Ein Raffungsfaktor kann bei diesem Befehl nicht angegeben werden.

BILD

```
**********
*        *
*  BILD  *
*        *
**********
```

listet in übersichtlicher Form den Inhalt des Lagers.
Zwei Befehlsvariationen sind möglich:

Befehlsform:

```
BILD
____ ____ ____ ____      *  |*  |*  |*   .  |*   .  |*   . |
```

oder

```
BILD      ALLE
____ ____ ____ ____      *  |*  |*  |*   .  |*   .  |*   . |
```

Der Befehl BILD listet die Namen aller Dateien, welche sich im Lager befinden. Ferner liefert die Liste die absoluten Positionen aller Dateien innerhalb des Lagers, also jeweils Anfangswert und Endwert, sowie die Anzahl der Werte pro Datei. Durch die Angabe ALLE werden noch zusätzlich die ersten acht Werte einer Datei ausgedruckt.

Bemerkung:

Die Verwendung dieses Befehles ist besonders zu empfehlen, da er dem Benutzer den Verarbeitungsablauf in SEDAP klar wiedergibt.

DAGE

```
**************
*            *
*    DAGE    *
*            *
**************
```

Datenerzeugung

Befehlsform:

```
DAGE          TYPE DAT?   IB1  IB2          X1          X2          X3
____          ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    . |
```

TYPE gibt den Typ des Signales an, welches erzeugt weden soll. Diese Angabe kann durch folgende Modifier ersetzt werden:

- KONS es wird eine Konstante erzeugt, welche durch die Abtastfrequenz X1 und durch die Amplitude X3 gekennzeichnet ist. X2 wird nicht verwendet.
- AX+B Es wird eine Rampe erzeugt, welche durch die Abtastfrequenz X1 gekennzeichnet ist. X3 gibt den ersten Punkt des S Signales an. X3+X2 den zweiten Punkt. Der n-te Punkte wird dann X3+X2(N-1) sein. Die Werte X2 und X3 können positiv oder negativ sein.
- SINF Es wird ein Sinus erzeugt mit der Abtastfrequenz X1, einer Sinusfrequenz X2 und der Amplitude X3. X1 und X2 müssen immer angegeben werden und folgenden Bedingungen genügen: X1 >= 2 * X2. X3 wird allgemein positiv angegeben. Der Benutzer kann aber durch eine negative Angabe eine Verschiebung um 180 Grad erreichen.
- COSF Es wird ein Cosinus erzeugt. Es gelten die gleichen Bestimmungen wie bei der Sinusfunktion.
- VIER erzeugt eine alternierende Viereckfunktion mit der Abtastfrequenz X1, einem Wiederholungsfaktor X2(HZ) und einer Amplitude X3. Die erste Hälfte des Vierecks ist gleich +X3, die zweite gleich -X3. X1 und X2 müssen genau angegeben werden. Ist X3 negativ, so ist die Funktion um 180 Grad verschoben
- RAND erzeugt Zufallszahlen. Die Frequenz der Werte wird X1 sein, die Amplitude liegt zwischen 0.0 und X3. X3 kann auch negativ sein. Die Angabe X2 bleibt unbeachtet.

DAT? ist der Name der erzeugten Datei.Das Fragezeichen soll auf die Dualität der Datei hinweisen:

- der Name kann neu sein. Es wird dann die erzeugte Datei unter diesem Namen im Lager gespeichert. In diesem Fall ist IB1 = 1 und IB2 der letzte Block der Datei, welcher erzeugt werden soll.

Fortsetzung siehe nächste Seite

DAGE - FORTSETZUNG

```
**************
*            *
*    DAGE    *
*            *
**************
```

- der Name kann schon bekannt sein, d.h. die Datei ist im Lager bereits vorhanden. In diesem Fall werden die erzeugten Signal-Daten auf die bereits vorhandenen Daten im Bereich der Dateisegmentgrenzen IB1 und IB2 aufaddiert. Die Dateisegmentgrenzen müssen innerhalb der bereits innerhalb der bereits vorhandenen Datei liegen. Die Angabe der Abtastfrequenz ist ohne Bedeutung. Sie ist durch die vorhandene Datei vorab festgelegt.

Beispiel

```
DAGE        AX+B RAMP     1   10       512.0       1.0       1.0
____        ____ ____ ____ *  |*   |*   |*    .   |*    .   |*    . |
```

Eine neue Datei mit 10 Blöcken und dem Namen RAMP wird erzeugt. Die Abtastfrequenz ist 512.0 Hz. Der erste Wert der Datei ist 1.0, der zweite 2.0 und der letzte 5120.

```
DAGE        SINF SINE     1    1       100.0       10.0       5.0
____        ____ ____ ____ *  |*   |*   |*    .   |*    .   |*    . |
```

Eine neue Datei mit der Größe von einem Block und der Abtastfrequenz von 100HZ soll erzeugt werden und unter dem Namen SINE ins Lager gespeichert werden. Die Sinusfrequenz ist 10 Hz. Dies bedeutet, daß 10 komplette Sinusperioden innerhalb 100 Punkten liegen. Die Amplitude ist 5.0.

```
DAGE        KONS MIXD     1     5       400.0                  10.0     .
____        ____ ____ *   |*   |*         .   |*    .    |*     . |
DAGE        VIER MIXD     2     2       400.0        8.5        1.0
____        ____ ____ *   |*   |*         .   |*    .    |*     . |
DAGE        RAND MIXD     4     4       400.0         .         1.0
____        ____ ____ *   |*   |*         .   |*    .    |*       |
```

Hier wird zuerst eine Datei erzeugt von 5 Blöcken Größe. mit den konstanten Werten 10.0, und der Abtastfrequenz 400.0 Hz. Auf den 2.Block wird dann eine Viereckfunktion mit der Amplitude 1.0 und auf den Block 4 werden Zufallszahlen, welche zwischen 0.0 und 1.0 liegen addiert.

Bemerkung

Eine Angabe des Raffungsfaktors hat bei diesem Befehl keinen Sinn.

DEFX

definiert die X-Achse einer Zeichnung.

Befehlsform:

```
DEFX      DATA TEXT        IB1  IB2   IR  XMIN       XMAX     XLANG
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, die die X-Achse bilden soll. Fehlt diese Angabe, dann ist die X-Achse durch die den Werten zugeordnete Zeit definiert
TEXT	ist ein Modifier mit dessen Hilfe unter die X-achse ein Text geschrieben werden kann. Dieser Text wird auf einer Kommentarkarte, die direkt hinter der DEFX-Karte liegen muß, angegeben. Fehlt TEXT, darf keine Kommentarkarte folgen. Der Text kann max. 60 Zeichen lang sein.
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen
XMIN	ist der kleinste Abszissenwert
XMAX	ist der größte Abszissenwert Wenn die X-Achse die Zeitachse ist, werden diese Werte in sec angegeben.
XLANG	gibt die Länge der X-Achse in cm an. Fehlt diese Angabe, so wird die Achse 20 cm lang.

Beispiel:

```
DEFX                                    11.5      11.7      10.0
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

In ein Diagramm mit einer 10 cm langen X-Achse werden alle Werte gezeichnet, die innerhalb dem geschlossenen Intervall von 11.5 und 11.7 liegen.

```
DEFX      TEMP             1    6    2
____      ____ ____ ____*   |*   |*   |*    .   |*    .   | . |
```

Die X-Achse des nachfolgenden Plottes wird durch die 3 Blöcke (IR=2) der Datei TEMP gebildet und in einen 20 cm langen Plot gezeichnet.

Fortsetzung siehe nächste Seite

DEFX - FORTSETZUNG

```
**************
*            *
*    DEFX    *
*            *
**************
```

Bemerkung:

Dieser Befehl kann maximal 20 Blöcke verarbeiten (1 Block = 512 Werte). Durch sinnvolle Angaben der XMIN und XMAX Werte ist es möglich, mehr als 20 Blöcke zu zeichnen, wenn die X-Achse die Zeitachse ist. Die Teilung der X-Achse erfolgt alle 2.5 cm und kann nicht verändert werden.

DEFY

```
**************
*            *
*    DEFY    *
*            *
**************
```

definiert die y-Achse eines Plottes.

Befehlsform:

```
DEFY          TEXT                          YMIN      YMAX     YHOEHE
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

TEXT ist ein Modifier mit dessen Hilfe neben die Y-achse ein Text geschrieben werden kann. Dieser Text wird auf einer Kommentarkarte, die direkt hinter der DEFY-Karte liegen muß, angegeben. Fehlt TEXT, darf keine Kommentarkarte folgen. Der Text kann max. 60 Zeichen lang sein.
YMIN ist der kleinste Ordinatenwert
YMAX ist der größte Ordinatenwert
YHOEHE ist die Höhe der y-Achse in cm. Fehlt diese Angabe, so wird 15.0 cm angenommen.

Beispiel:

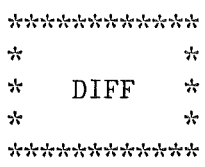
```
DEFY                                    -5.0      105.6      50.0
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

In ein Diagramm mit einer 50 cm langen Y-Achse werden alle Werte gezeichnet, die in dem geschlossenen Intervall von -5.0 bis 105.6 liegen.

Bemerkung:

Siehe PLOT Befehl und DEFX Befehl.
Die Teilung der Y-Achse erfolgt alle 2.5 cm und kann nicht verändert werden.

DIFF



differenziert die Werte einer Datei und speichert das Ergebnis in das Lager.

Befehlsform:

```
DIFF      DATA      DATB   IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche differenziert werden soll
DATB	ist der Name der differenzierten Datei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

```
DIFF      SPID      ACCE     1    9    3
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die ersten 9 Blöcke der Datei SPID werden differenziert und die entstandenen 3 Ergebnisblöcke, da IR=3, werden unter dem Namen ACCE in das Lager gespeichert.

Bemerkung:

Minimale Punktezahl ist bei diesem Befehl 3 Punkte.

DIKO

```
**************
*            *
*    DIKO    *
*            *
**************
```

erledigt die komplexe Division zweier Dateien und speichert das komplexe Ergebnis in das Lager.

Befehlsform:

```
DIKO       DAXA DAXB DAXC   IB1  IB2
____       ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DAXA	ist der Name der komplexen Datei, welche durch DAXB dividiert werden soll
DAXB	ist der Name der komplexen Datei, durch die die Datei DAXA dividiert wird
DAXC	ist der Name der komplexen Ergebnisdatei (C=A/B)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

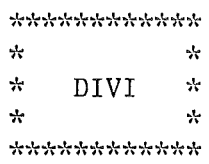
```
DIKO       SPKI WEIG QUOT     2    3
____       ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die Blöcke 2 und 3 der Datei SPKI werden durch die Blöcke 2 und 3 der Datei WEIG dividiert. Die beiden Ergebnisblöcke werden dann unter dem Namen QUOT im Lager gespeichert.

Bemerkung:

Die Angabe eines Raffungsfaktors ist nicht erlaubt (komplexe Operation).
Siehe Sonderbemerkungen für die Verarbeitung von synchronen Dateien im SEDAP KfK-Bericht.
Man muß berücksichtigen, daß in diesem Fall ein SEDAP Block aus 256 Real- und 256 Imaginärwerten besteht.

DIVI



dividiert zwei Dateien und speichert das Ergebnis in das Lager

Befehlsform:

```
DIVI      DATA DATB DATC   IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche dividiert werden soll
DATB	ist der Name der Datei, durch welche dividiert wird
DATC	ist der Name der Ergebnisdatei nach der Division (C=A/B)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

```
DIVI      CH15 CH16 RATE     2   21   10
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Dividiere CH15 durch CH16 (20 Blöcke) und speichere die beiden Ergebnisblöcke da IR=10 , unter dem Namen RATE in das Lager.

Bemerkung:

DIVI kann keine komplexe Daten dividieren. Siehe Befehl DIKO. Falls die Werte der Datei DATB = 0.0 sind und somit ein DIVIDE CHECK auftreten kann, werden diese Werte = 1.0 gesetzt. Die Anzahl der aufgetretenen Nullen wird am Ende der Ausgabeliste für diesen Befehl in Form von Warnungen aufgelistet.

DUMP

```
**************
*            *
*    DUMP    *
*            *
**************
```

rettet eine oder mehrere Dateien aus dem Lager auf externe Einheiten (Band oder Platte); zwei Befehlsformen sind möglich:

Befehlsform:

```
DUMP      DATA              IB1  IB2 IFIL
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |

DUMP           ALLE                   IFIL
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche gedumpt werden soll
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IFIL	ist die Filenummer und referiert auf den FT-Namen der DD-KARTE im G-Step
ALLE	ist eine Angabe, die bewirkt, daß der gesamte Lagerinhalt gedumpt wird.

Beispiel:

```
DUMP      TE25              1    9   22
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Dumpe die Datei TE25 von Block 1 bis Block 9 auf einen temporären Dataset mit der Filenummer //G.FT22F001 DD .....

```
DUMP           ALLE                   22
____      ____ ____ ____*    |*   |*   |*    .   |*    .   |*    . |
```

Dumpe alle sich im Lager befindenden Dateien auf einen Dataset mit der Filenummer //G.FT22F001 DD .....

Bemerkung

Nach dem Dumpbefehl muß eine Kommentarkarte folgen, die nicht mit dem Zeichen > in Spalte 1 beginnen darf.
Dieser Befehl kann als Schnittstelle des SEDAP-Systemes zu anderen Programmen verstanden werden, oder einfach die Speicherung von SEDAP Ergebnissen auf längere Zeit darstellen. Durch den Befehl HOLE können diese Werte dem System wieder zugänglich gemacht werden. Das Format der durch den Befehl DUMP erzeugten Datei ist beim Befehl HOLE beschrieben.

FANA

```
**************
*            *
*    FANA    *
*            *
**************
```

berechnet aus den Koeffizienten C(K) eines mit der komplexen Fouriertransformation erzeugten Spektrums die Koeffizienten A(K) und B(K) der reellen SIN-COS Transformationen mit der Beziehung

$$A(COS) = 2 * REAL(C)$$
$$B(SIN) = -2 * IMAG(C)$$

Befehlsform:

```
FANA      COMP      COSI    IB1  IB2
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

COMP	ist der Name der Eingabedatei (komplexes Spektrum)
COSI	ist der Name der Ergebnisdatei (A und B paarweise)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

```
FANA      COMP      COSI      1    4
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Von den ersten 4 Blöcken der Datei COMP werden die Koeffizienten der COS und SIN - Fouriertransformation berechnet und paarweise (COS-SIN) in der Datei COSI abgespeichert.

Bemerkung:

Die Angabe eines Raffungsfaktors ist nicht möglich.
Die paarweise abgespeicherte SIN und COS-Reihe kann mit dem Befehl SO02 getrennt werden. Im Namen der getrennten Datei muß statt $$ das Zeichen FT stehen (siehe Befehl SOnn).

FANT

```
**************
*            *
*    FANT    *
*            *
**************
```

wird benutzt zur Rücktransformation eines komplexen Spektrums mit Hilfe der schnellen Fouriertransformation in den Zeitbereich mit der Beziehung:

$$Zeit(j) = \sum_{k=1}^{n} Spec(k) e^{(2i\pi(j-1)(k-1))}$$

$$i=\sqrt{(-1)} \; ; \; j = 1 \text{ bis } n$$

Befehlsform

```
FANT      SPEC      ZEIT    IB1  IB2
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

SPEC	ist der Name der Eingabedatei
ZEIT	ist der Name der Ausgabedatei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

```
FANT      SPEC      ZEIT      1    4
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die ersten 4 Blöcke der Datei SPEC werden transformiert und unter dem Namen ZEIT im Lager abgespeichert.

Bemerkung:

Dieser Befehl kann maximal 16 Blöcke verarbeiten (1 Block=512 Werte). Ein Raffungsfaktor kann nicht angegeben werden.
FANT erwartet, daß das Spektrum ursprünglich mit der schnellen Fouriertransformation (Befehle: FOUT, LEDI) erzeugt wurde. Da in SEDAP nur der notwendige Teil eines Spektrums transformiert wird, wird dieses vor der Rücktransformation restauriert, d.h. die Werte der Frequenzen Nyquistfrequenz bis Tastfrequenz werden wieder zugefügt (hermetische Symmetrie). Die Frequenz (0) erhält den Wert Null. Wird ein Spektrum nicht vollständig rücktransformiert, geht entweder die Information der hohen Frequenzen verloren (Tiefpass) oder das Ergebnis wird falsch, der erste Block des verlangten Dateisegmentes ist dann nicht der Anfangsblock des Records.

FILN

```
**************
*            *
*    FILn    *
*            *
**************
```

dient zum Glätten einer Datei
FILn ist der allgemeine Name und steht für die Befehle FIL1, FIL2 und FIL3.

Befehlsform:

```
FILn       DATA       DATC    IB1  IB2   IR
____       ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    . |
```

DATA	ist der Name der Datei, die gefiltert werden soll
DATC	ist der Name der Ergebnisdatei nach der Filterung
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

```
FILn       RAW1       SM01      1    2    2
____       ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    . |
```

Die beiden ersten Blöcke von RAW1 werden gefiltert und die Ergebnisdatei, welche ein Block groß ist, in das Lager mit dem Namen SM01 gespeichert.

Bemerkung:

FILn benutzt verschiedene Filteralgorithmen, welche im Teil 2 des KFK-Berichtes beschrieben sind. (Siehe auch Bemerkungen im Kapitel 3.3 des KFK Berichtes 1594.) Minimale Punktezahl bei FIL1 = 3, FIL2 = 5 und FIL3 = 5 Punkte.

FIL4

```
**************
*            *
*    FIL4    *
*            *
**************
```

glättet als Tiefpassfilter eine Datei mit einer vom Benutzer angegebenen Grenzfrequenz FREQ = 1./TAU gemäß der Differntialgleichung:
TAU * D (Y(T))/DT + Y (T) = X (T) (X=ungefiltert, Y=gefiltert).

Befehlsform:

```
FIL4      DATA      DATB    IB1  IB2   IR   FREQ
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Inputdatei
DATB	ist der Name der Ergebnisdatei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen
FREQ	ist die 'CUT-OFF' Frequenz des Filters

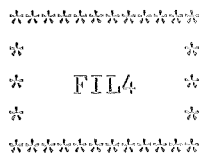
Beispiel:

```
FIL4      CH21      DA21      1   10    2   10.
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die ersten 10 Blöcke der Datei CH21 werden nach vorhergehender Raffung gefiltert, so daß sich ein Signal mit der oberen Grenzfrequenz von 10 Hz ergibt. Die fünf Blöcke der Ergebnisdatei DA21 werden in das Lager gespeichert.

Fortsetzung siehe nächste Seite

FIL4 - FORTSETZUNG



Bemerkung:
FIL4 simuliert ein sogenanntes RC-filter, d.h. ein Tiefpassfilter erster Ordnung. Während bei FIL1, FIL2 und FIL3 der Glättungseffekt immer der Abtastfrequenz angepaßt ist, kann bei FIL4 der Filtereffekt in weitem Umfang variabel gesteuert werden:

- Der Filtereffekt hängt von der Grenzfrequenz FREQ ab. FREQ ist der normalisierte Reziprokwert der Zeitkonstante (RC=TAU) des Filters, das heißt, wenn FREQ = 0.1, hat das Filter eine Zeitkonstante von 10 sec.
- Die Grundfrequenz muß kleiner als die Abtastfrequenz sein (andernfalls erfolgt Jobabbruch). Der Benutzer sollte bedenken, daß die effektive Tastfrequenz gleich der gemessenen Tastfrequenz dividiert durch den Raffungsfaktor ist.
- Eine zu große Zeitkonstante,d.h. eine zu kleine Grenzfrequenz FREQ kann den Informationsgehalt eines Signals zerstören. Z.B. erstreckt sich bei FREQ = 0.01 und einer Tastfrequenz von 50 Hz die Glättung über 10000 Punkte, da das Filter jeweils über zwei Zeitkonstanten (a 100 sec) wirkt.

Minimale Punktzahl = 3

FOUT

```
**************
*            *
*    FOUT    *
*            *
**************
```

Berechnet aus reellen Zeitsignalen mit Hilfe der schnellen Fouriertransformation komplexe Spektren, gemäß der Gleichung:

$$n \cdot Spec(k) = \sum_{j=1}^{n} Zeit(j) e^{(-2i\pi(j-1)(k-1))}$$

$$i=\sqrt{(-1)} \; ; \; k = 1 \text{ bis } n$$

Befehlsform:

```
FOUT       ZEIT       SPEC    IB1   IB2
____       ____ ____  ____  *   |*   |*   |*     .    |*     .    |*     . |
```

ZEIT	ist der Name der Eingabedatei (reelles Zeitsignal)
SPEC	ist der Name der Ergebnisdatei (komplexes Spektrum)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

```
FOUT       ZEIT       SPEC      1     5
____       ____ ____  ____  *   |*   |*   |*     .    |*     .    |*     . |
```

Die ersten 5 Blöcke der Datei ZEIT werden transformiert. Da die schnelle Fouriertransformation eine Menge von 2**M Werten erwartet, wird der Eingaberecord um 3 Blöcke (a 512 Werte), gefüllt mit Nullen, erweitert. Die transformierten Werte werden unter dem Namen SPEC (8 Blöcke) in das Lager gespeichert.

Bemerkung:
Dieser Befehl kann max. 16 Blöcke verarbeiten (1 Block = 512 Werte)
Ein Raffungsfaktor kann nicht angegeben werden.
Bei der schnellen Fouriertransformation sollte die Zahl der Werte N = 512 * (IB2 - IB1 + 1) eine ganzzahlige Potenz von 2 sein (N=2**M). Im anderen Fall werden so viele Nullen zugefügt bis N = 2**M.
Vor der Transformation wird der lineare Mittelwert des Signals eliminiert. Das komplexe Spektrum wird für den Frequenzbereich F = FTAST/N (Grundfrequenz) bis F = FTAST/2 (Nyquistfrequenz) ins Lager übertragen.
Durch das Zufügen von Nullen werden die komplexen Werte um den Faktor

Fortsetzung siehe nächste Seite

FOUT - FORTSETZUNG

```
**************
*            *
*    FOUT    *
*            *
**************
```

NIN/NM zu klein berechnet.
(NIN = 512 * (IB2 - IB1 + 1), NM = 2**M>=NIN)
Die korrekten Werte können mit dem Befehl AX+B (A = NM/NIN) erzeugt werden. Real- und Imaginärteil des komplexen Spektrums kann mit dem Befehl SO02 getrennt werden. Im Namen der getrennten Dateien muß dann statt $$ das Zeichen FT stehen. Ausführliche Beschreibung der schnellen Fouriertransformation und ihre Anwendung finden Sie in:
IEE Transactions on Audio and Electroacoustics, Vol. AU-15, No2, June 1967 (special issue) und
Webb C. Practical Use of the Fast Fourier Transform (FFT), Algorithm in Time-Series Analysis, Texas University Austin, Texas, AD713166,22 June 1970.

HAFU

```
**************
*            *
*    HAFU    *
*            *
**************
```

zur Glättung komplexer Spektren mit der sogenannten Hanning-funktion, einer gleitenden 3-Punkt-Mittelung

Befehlsform:

```
HAFU       DATA       DATB    IB1  IB2         XSIGN
____       ____ ____  ____  *   |*   |*   |*     .   |*     .   |*     . |
```

DATA	ist der Name der Datei, die geglättet werden soll(rohes Spektrum)
DATB	ist der Name der geglätteten Datei (modifiziertes Spektrum)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
XSIGN	ist entweder 1. oder -1. (siehe Bemerkung)

Beispiel:

```
HAFU       ROSP       MOSP      1    4         1.
____       ____ ____  ____  *   |*   |*   |*     .   |*     .   |*     . |
```

Die ersten 4 Blöcke der Datei ROSP werden geglättet mit dem positiven Hanning-Algorithmus (siehe Bemerkung). Das Ergebnis wird unter dem Namen MOSP im Lager abgespeichert.

Bemerkung:

Gleichung der komplexen Glättung:

XSIGN = +1.:

```
MOSP (1) = 0.5 * (ROSP(1) + ROSP(2))
MOSP (K) = 0.25 * (2*ROSP(K) + ROSP(K-1) + ROSP(K+1))
       K = 2,3, .... , N-1
MOSP (N) = 0.5 * (ROSP(N-1) + ROSP(N))
```

Anwendung bei quadratischen Spektren

XSIGN = -1.:

```
MOSP (1) = 0.5 * (ROSP(1) - ROSP(2))
MOSP (K) = 0.25 * (2*ROSP(K) - ROSP(K-1) - ROSP(K+1))
       K = 2,3, .... , N-1
MOSP (N) = 0.5 * ( -ROSP(N-1) + ROSP(N))
```

Anwendung bei linearen Spektren

Näheres siehe Literaturhinweis bei Befehl FOUT
Minimale Punktzahl = 3

HOLE

```
**************
*            *
*    HOLE    *
*            *
**************
```

holt im SEDAP-Format geschriebene Dateien von einem Datenträger (Band oder Platte)

Befehlsform:

```
HOLE                DATA          IFIL
____  ____ ____ ____ *  |*  |*  |*    .  |*    .  |*    . |
HOLE        ALLE                  IFIL
____  ____ ____ ____ *  |*  |*  |*    .  |*    .  |*    . |
```

DATA ist der Name der Datei, welche geholt werden soll
IFIL ist die Filenummer und referiert auf den Datenfile mit der dazugehörigen DD - Karte im Go Step
ALLE ist eine Angabe, die bewirkt, daß alle Dateien, die sich auf dem File befinden, in das Lager geholt werden.

Beispiel:

```
HOLE                TEMP          22
____  ____ ____ ____ *  |*  |*  |*    .  |*    .  |*    . |
```

Hole die Datei TEMP vom Datenfile mit der dazugehörigen DD-Karte //G.FT22F001. (Siehe Kapitel IBM-Job-Control,Beispiel 1).

Bemerkung:

Fortsetzung siehe nächste Seite

HOLE - FORTSETZUNG

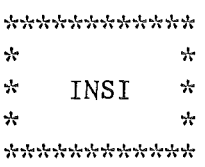
```
**************
*            *
*    HOLE    *
*            *
**************
```

Der HOLE-File enthält eine oder mehrere Dateien im Sinne von SEDAP. Dabei besteht solch eine Datei aus einem Kopf, der die Datei beschreibt, wie Name, Länge u.s.w. und den eigentlichen Daten, welche in Blöcken zu 512 Werten zusammengefaßt sind. Siehe auch Kapitel: SEDAP ANSCHLUSSPROGRAMM.

Der Dateikopf ist wie folgt aufgebaut:

Wort	Typ	Bedeutung	Bemerkung für ext. Programmerstellung
1	INT4	Länge des Vorsatzes	immer 128 Bytes
2	INT4	Lfd. Nr. des Datensatzes	Inhalt beliebig
3	INT4	Anzahl Werte im Satz	Angabe beliebig
4	INT4	Anzahl Werte letzter Block	Angabe notwendig
5	CHAR(80)	Beliebiger Text (80 Bytes)	Angabe notwendig
6	CHAR(4)	Name der Datei	Angabe beliebig
7	REAL4	Frequenz der Datei	Angabe notwendig
8	REAL4	Datum der Datei	Angabe notwendig
9	REAL4	Zeit der Datei	Angabe notwendig
10-13	REAL4	unbenutzt	

INSI



integriert eine Datei nach der Simpson Regel und speichert das Ergebnis in das Lager

```
INSI      DATA      DATB   IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche integriert werden soll
DATB	ist der Name der integrierten Datei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

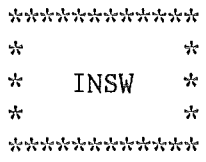
```
INSI      CH21      PR21      1    1    4
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Der erste Block der Datei CH21 wird integriert und das Ergebnis von 128 Werten wird unter dem Namen PR21 in das Lager gebracht.

Bemerkung:

Der Befehl benötigt zur Ausführung eine Mindestpunktzahl von 3 Punkten.

INSW



Integration mit Switch.
Integriert eine Datei nach der Trapezregel und speichert das Ergebnis in das Lager.
Das Integral des Signals kann auf einen vorgegebenen Wert mit Hilfe einer Steuerfunktion gesetzt werden. Das Zurücksetzen des Integrals erfolgt jedesmal, wenn die Steuerfunktion einen vom Benutzer definierten Schwellwert kreuzt. Eine typische Anwendung ist die Integration eines periodischen Signals (SINUS) oder eines pseudo-periodischen Signals (pulsartige Schockwellen), die leichter zu interpretieren ist, wenn das Integral periodisch auf einen vorgegebenen Wert zurückgesetzt wird.

Befehlsform:

```
INSW      DATA SWIT DATB   IB1  IB2   IR   TRIG      SETZ
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche integriert werden soll
SWIT	ist der Name der Datei, die die Steuerfunktion enthält
DATB	ist der Name der integrierten Datei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen
TRIG	ist der Schwellwert für die Steuerfunktion
SETZ	ist der Wert, auf den die Ergebnisdatei während der Integration jedesmal gesetzt wird, wenn der Schwellwert TRIG von der Steuerfunktion SWIT unter- oder überschritten wird

Beispiel:

```
INSW      SINE SINE HACK     1    2
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die ersten 2 Blöcke der Datei SINE werden integriert und das Ergebnis unter dem Namen HACK im Lager abgespeichert. Als Steuerfunktion wird ebenfalls die Datei SINE verwendet. Der Schwellwert der Steuerfunktion ist ebenso wie der Anfangswert der Integrationsabschnitte = 0.0, so daß unter der Annahme SINE sei eine Sinusfunktion die Integration nach jeder halben Periode wieder mit dem Wert 0.0 beginnt. Damit ergibt

Fortsetzung siehe nächste Seite

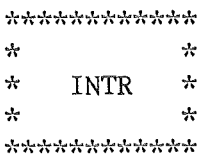
INSW - FORTSETZUNG

```
**********
*        *
*  INSW  *
*        *
**********
```

sich aus dem Signal X=SIN (ALPHA*T) nach der Integration ein zerhacktes Cosinussignal JX = SIGN*0.5*(1.-COS(ALPHA*T)/ALPHA)
mit SIGN = +1. für die erste Hälfte und
mit SIGN = -1. für die zweite Hälfte einer Periode.

Bemerkung:
Als Steuerfunktion sollte man keine Dateien verwenden, die die Schwelle nach jedem zweiten Punkt kreuzen.

INTR



integriert eine Datei nach der Trapezregel und speichert das Ergebnis in das Lager

Befehlsform:

```
INTR      DATA      DATB   IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche integriert werden soll
DATB	ist der Name der integrierten Datei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

```
INTR      CH15      IN15     1    4    4
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die ersten 4 Blöcke der Datei CH15 sollen integriert werden und das Ergebnis von einem Block wird in das Lager unter dem Namen IN15 gespeichert.

Bemerkung:

Der Befehl benötigt zur Ausführung eine Mindestpunktzahl von zwei Punkten.

KETT

```
**************
*            *
*    KETT    *
*            *
**************
```

verkettet zwei Dateien miteinander und bildet eine neue Datei

Befehlsform:

```
KETT       DATA DATB DATC
____       ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA ist der Name der ersten Datei
DATB ist der Name der zweiten Datei, welche an die erste angehängt werden soll
DATC ist der Name der neu entstandenen Datei

Beispiel:

```
KETT       AT01 AT02 NEU
____       ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

An die Datei AT01 wird die Datei AT02 angehängt. Die neu entstandene Datei heißt NEU.

Bemerkung:

Die neue Datei bekommt die Anfangszeit, das Datum und die Frequenz der ersten Datei. Ist der letzte Block der ersten Datei nicht vollkommen gefüllt, so wird die nächste Datei direkt an den letzten gültigen Wert der ersten Datei angehängt. Es ist weder eine Blockangabe noch eine Raffungsangabe möglich.

KOKO

```
**************
*            *
*    KOKO    *
*            *
**************
```

ermöglicht die Multiplikation einer komplexen Datei mit einer ebenfalls komplexen Datei, wobei jedoch eine der Dateien vor der Multiplikation konjugiert komplex gemacht wird.

Befehlsform:

```
KOKO      DCXA DCXB DCXC   IB1  IB2
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DCXA	ist der Name der Datei, deren Werte in konjugiert komplexe Werte vor der Multiplikation umgewandelt werden
DCXB	ist der Name der zweiten komplexen Datei
DCXC	ist der Name der Ergebnisdatei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

```
KOKO      SPC1 SPC2 XREC     1    4
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die vier ersten Blöcke der Dateien SPC1 und SPC2 werden miteinander multipliziert. Vorher werden die Werte der Datei SPC1 in konjugiert komplexe Werte umgewandelt. Die Ergebnisdatei von 4 Blöcken wird unter dem Namen XREC im Lager gespeichert.

Bemerkung:

Eine Angabe des Raffungsfaktors ist nicht erlaubt.
Ein SEDAP Block besteht bei diesem Befehl aus zwei Blockhälften zu je 256 Werten, wobei der eine Teil real und der andere imaginär ist.

LEDI

```
**************
*            *
*    LEDI    *
*            *
**************
```

berechnet die mittlere Leistungsdichte (Auto- oder Kreuzkorrelation) nach der Ensemble-Technik. Mögliche Varianten: Segmentierung des Signals, Überlappung der Segmente, Glättung der quadratischen Spektren mit Hanningfunktion und Berechnung einer aperiodischen Korrelation.

Befehlsform:

```
LEDI      SIGA SIGB SPEC   IB1  IB2 ISEG   XLAP      XGLAT    XAPER
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    .  |
```

SIGA	ist der Name der Eingabedatei (Zeitsignal)
SIGB	ist der Name der zweiten Eingabedatei (Zeitsignal)
SPEC	ist der Name der Ergebnisdatei (korreliertes Spektrum)
IB1	ist der erste Block des insgesamt zu transformierenden Dateibereiches
IB2	ist der letzte Block des insgesamt zu transformierenden Dateibereiches
ISEG	ist die Länge der Dateisegmente, in die die Eingangsdaten zerlegt werden und gleichzeitig auch die Länge der Ergebnisdatei (0 < ISEG <= 16)
XLAP	ist die Länge des Überlappungsbereiches der Segmente (0 < XLAP < ISEG)
XGLAT	ist der Wiederholungsfaktor für die Anwendung der Hanningglättung (0 <= XGLAT <= 10)
XAPER	= 1. Ergibt eine vollständige aperiodische Korrelation. Sonst ergeben sich soviele aperiodische Werte wie Nullen den Dateisegmenten zugefügt werden.

Beispiel:

```
LEDI      SIGA SIGA APSD     1  100    4
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    .  |
```

Von den ersten 100 Blöcken der Datei SIGA wird die mittlere autokorrelierte Leistungsdichte berechnet. Die Transformation erfolgt in Abschnitten von 4 Blöcken, bis der ganze Eingangsrecord abgearbeitet ist. Eine Überlappung der Segmente, sowie eine Glättung der quadratischen Spektren findet nicht statt. Der lineare Mittelwert der 25 quadratischen Spektren wird unter dem Namen APSD im Lager abgespeichert. Die Korrelation ist zyklisch.

Fortsetzung siehe nächste Seite

LEDI - FORTSETZUNG

```
**********
*        *
*  LEDI  *
*        *
**********
```

```
LEDI      TIMA TIMB CPSO     1   16    3    1.0       3.0       1.0
____      ____ ____ ____ *  |*  |*   |*    .   |*    .   |*    .  |
```

Von den ersten 16 Blöcken der Datei TIMA und TIMB wird die mittlere kreuzkorrelierte Leistungsdichte berechnet. Die Transformation erfolgt in Segmenten von 3 Blöcken, mit Überlappung von je einem Block. Die Spektren werden je dreimal geglättet. Die Spektren haben eine Länge von 4 Blöcken (512 Nullen werden jeweils zugefügt, siehe Befehl FOUT). Der lineare Mittelwert der (8) korrelierten Spektren wird unter dem Namen CPSD im Lager abgespeichert. Die Korrelation ist aperiodisch.

Bemerkung

Siehe Sonderbemerkungen für die Verarbeitung von synchronen Dateien im SEDAP KfK-Bericht. Ein Raffungsfaktor kann nicht angegeben werden. Vor der Transformation der Dateisegmente wird ihr linearer Mittelwert berechnet und von diesen abgezogen. Die Länge der Segmente muß eine Potenz von 2 sein, sonst werden die Segmente um eine Anzahl von Nullen verlängert. Im Fall der aperiodischen Korrelation wird die Länge der Segmente verdoppelt durch Anhängen von Null-Werten.
ISEG kann dann max. 8 werden.
Die Länge der Ergebnisdatei ist:

```
(512 * ISEG + Zahl der Nullen)/512
```

Durch das zufügen von NUllen werden die Werte um den Faktor

```
(512 * ISEG / (512 * ISEG + Zahl der Nullen)) ** 2
```

zu klein (Korrektur mit dem Befehl AX+B möglich).
Die Mittelung der erzeugten quadratischen Spektren erfolgt linear.
Das komplexe Spektrum wird für den Frequenzbereich F = FTAST/N bis FTAST /2 (Grund- bis Nyquistfrequenz) ins Lager übertragen.
Real/ und Imaginärteil des Spektrums kann mit SO02 getrennt werden. Im Namen der getrennten Datei muß dann statt $$ das Zeichen FT stehen.
Literaturhinweis: siehe FOUT

Fortsetzung siehe nächste Seite

LEDI - FORTSETZUNG

```
**************
*            *
*    LEDI    *
*            *
**************
```

LEDI als Zusammenfassung von Standard Operationen:

Das Kommando LEDI wurde definiert, um dem Benutzer weiteren Komfort zu bieten. Die gleichen Operationen, die im 2. Beispiel spezifiziert sind, hätten auch mit den vorhandenen allgemeinen Operatoren durchgeführt werden können. Es würde sich jedoch eine umfangreiche Kommandoliste ergeben, das Lager würde stärker belegt und außerdem würden die vielen Transferoperationen die Gesamtrechenzeit stark vergrößern. Um den Unterschied in der Programmierung zu zeigen, folgt das Schema einer Kommandoliste, die das im obigen 2. Beispiel berechnete Leistungsspektrum liefert (siehe unten).
Da zur Berechnung der aperiodischen Korrelation kein Operator zur Verfügung steht, werden zunächst, um die Zahl der notwendigen Nullwerte zu erhalten, Hyperrecords gebildet.
Das erste Kommando erzeugt einen Nullrecord ANUL, der jedoch die gleichen beschreibenden Parameter wie die Spektren hat. Zur Bildung der Hyperrecords werden die Segmente der Signale dem Nullrecord überlagert (HA1..., HA8, HB1...HB8). Es folgen die FFT der Hyperrecords, die Korrelation durch komplex konjugierte Multiplikation, die Dreifachglättung der quadratischen Spektren mit der HANNING Funktion und schließlich die Mittelwertbildung durch Addition und Normalisierung. Ein Faktor 2 berücksichtigt, daß sich nach der konjugiert komplexen Multiplikation nur der halbe Wert der Leistungsdichte ergibt. Ein weiterer Faktor =7.1 berücksichtigt die Verfälschung durch das Zufügen von Nullen. (Damit ergibt sich für die Korrektur von SUMS : SUMS*2*7.1/8 = SUMS* 1.77). Über dieses Befehlsprogramm werden zwar viele Zwischeninformationen greifbar, es werden jedoch zusätzliche 78 Befehlskarten benötigt und 624 Blöcke im Lager belegt.

Fortsetzung siehe nächste Seite

LEDI - FORTSETZUNG

```
**************
*            *
*    LEDI    *
*            *
**************
```

Beispiel einer Abschätzung der Leistungsdichte ohne LEDI Befehl

```
AX+B      TIMA        ZERO     1    8     0.        0.
AX+B      TIMA        TSAI     1    3
ADDI      ZERO TSA1   HRA1     1    8
FOUT      HRA1        RSA1     1    8
AX+B      TIMB        TSB1     1    3
ADDI      ZERO TSB1   HRB1     1    8
FOUT      HRB1        RSB1     1    8
KOKO      RSA1 RSB1   QSP1     1    8
HAFU      QSP1        MS11     1    8
HAFU      MS11        MS21     1    8
HAFU      MS21        MSP1     1    8
                                                  1
AX+B      TIMA        TSA2     3    5
ADDI      ZERO TSA2   HRA2     1    8
  .                                          .
  .                                          .
  .                                              2
  .                                          .
  .                                          .
  .                                          .
  .                                          .
  .                                              7
AX+B      TIMA        TSAB    14   16
  .                                          .
  .                                          .
  .                                          .
                                                 8
ADDI      MSP1 MSP2   ADDI     1    8
ADDI      MSP3 ADDI   ADD2     1    8
ADDI      MSP4 ADD2   ADD3     1    8
ADDI      MSP5 ADD3   ADD4     1    8
ADDI      MSP6 ADD4   ADD5     1    8
ADDI      MSP7 ADD5   ADD6     1    8
ADDI      MSP8 ADD6   SUMS     1    8
AX+B      SUMS        CPSD     1    8    1.77      0.
  .
  .
  .
STOP
```

LOGA

```
**************
*            *
*    LOGA    *
*            *
**************
```

logarithmiert die Werte einer Datei zu einer beliebigen Basis

Befehlsform:

```
LOGA       DATA       DATB    IB1   IB2    IR   BASIS
____       ____ ____  ____ *   |*   |*    |*     .    |*     .    |*     . |
```

DATA	ist der Name der Datei, welche logarithmiert werden soll.
DATB	ist der Name der logarithmierten Datei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen
BASIS	ist der Basiswert des Logarithmus

Beispiel:

```
LOGA       INDA       LOGA      1     4     2    10.
____       ____ ____  ____ *   |*   |*    |*     .    |*     .    |*     . |
```

Jeder zweite Wert der Datei INDA wird zur Basis 10.0 logarithmiert und in die Ergebnisdatei LOGA gespeichert.

Bemerkung:

Ist der Basiswert = 0.0, so wird zur Basis e = 2.718281 logarithmiert. Ist der Basiswert negativ, so wird zur Basis |BASIS| logarithmiert.

MUKO

erledigt die Multiplikation zweier komplexer Dateien und speichert das Ergebnis in das Lager

Befehlsform:

```
MUKO      DAXA DAXB DAXC   IB1  IB2
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DAXA	ist der Name der komplexen Datei, welche multipliziert werden soll
DAXB	ist der Name der komplexen Datei, mit welcher multipliziert werden soll
DAXC	ist die komplexe Ergebnisdatei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

```
MUKO      SPCI SPC2 XREC     5    6
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die Blöcke 5 und 6 der komplexen Datei SPCI werden mit den Blöcken 5 bis 6 der Datei SPC2 multipliziert. Die entstandene komplexe Ergebnisdatei XREC wird in das Lager gespeichert.

Bemerkung:

Bei diesem Befehl kann kein Raffungsfaktor angegeben werden.
Siehe Sonderbemerkungen für die Verarbeitung von synchronen Dateien im SEDAP KfK Bericht.

MULT

```
**************
*            *
*    MULT    *
*            *
**************
```

multipliziert zwei Dateien miteinander und speichert das Ergebnis in das Lager

Befehlsform:

```
MULT      DATA DATB DATC   IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    . |
```

DATA	ist der Name der Datei, welche multipliziert werden soll.
DATB	ist der Name der Datei mit welcher multipliziert werden soll
DATC	ist der Name der Ergebnisdatei nach der Multiplikation (C=A*B)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

```
MULT      DA01 DACC PROD     2    3    4
____      ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    . |
```

Die Blöcke 2 und 3 der Dateien DA01 und DACC werden miteinander multipliziert und das Ergebnis von 256 Werten (IR=4) unter dem Namen PROD im Lager abgespeichert.

Bemerkung:

Siehe Sonderbemerkungen für die Verarbeitung von synchronen Dateien im SEDAP KfK Bericht.
MULT kann keine Dateien mit komplexen Werten miteinander multiplizieren (siehe Befehle MUKO und KOKO).

MWEF

berechnet den Mittelwert einer Datei und speichert das Ergebnis für einen späteren Gebrauch durch MWES ab.

Befehlsform:

```
MWEF      DATA            IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei aus welcher der Mittelwert bestimmt werden soll
IB1	ist der erste Block des verlangen Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

```
MWEF      MITL             1    3
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Aus den ersten 3 Blöcken der Datei MITL wird der Mittelwert errechnet und abgespeichert. Auf der Ausgabeliste erscheint ebenfalls der Mittelwert.

MWES

```
**************
*            *
*    MWES    *
*            *
**************
```

subtrahiert von den Werten einer Datei einen durch den Befehl MWEF berechneten Mittelwert.

Befehlsform:

```
MWES      DATA      DATB   IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei von deren Werten der Mittelwert subtrahiert werden soll
DATB	ist der Name der Ergebnisdatei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen.

Beispiel:

```
MWEF      MITL              1    3
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |

MWES      MITL      ZMIT    1    3
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Beide Befehle, in dieser Reihenfolge eingegeben, bewirken folgendes: Von der Datei MITL wird der Mittelwert errechnet. Dann wird er von jedem Wert der Datei MITL abgezogen und die neu entstandene Datei von drei Blöcken wird unter dem Namen ZMIT im Lager abgespeichert.

Bemerkung:

Beide Befehle müssen, falls der oben gezeigte Effekt erzielt werden soll, nacheinander in der SEDAP Eingabe liegen. Die Angaben von DATA, IB1, IB2 und IR müßen dann übereinstimmen.

NGET

```
**********
*        *
*  NGET  *
*        *
**********
```

konvertiert experimentell erfaßte Daten und speichert sie in Blöcken zu je 512 Werten in das Lager

Befehlsform:

```
NGET                DATA  ITB1 ITB2 IFIL
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche geholt werden soll
ITB1	ist der erste Block der konvertiert werden soll
ITB2	ist der letzte Block der konvertiert werden soll
IFIL	ist die Filenummer und referiert auf eine DD-Karte die die Eingabedatei beschreibt. IFIL sollte zwischen 02 und 50 liegen, mit Ausnahme von 05,06 07,15,40.

Beispiel:

```
NGET                TA33     3   22   29
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Zwanzig Blöcke der durch die DD-Karte 'G.FT29...' referierten Datei sollen konvertiert werden und unter dem Namen TA33 im Lager abgespeichert werden. TA33 ist dann 40 Blöcke groß.

Bemerkung:

NGET erwartet folgendes Dateiformat:

- Datenblöcke mit 1024 Werten. Diese werden bei der Konvertierung in Blöcke mit 512 Werten geteilt.
- Die drei ersten Blöcke dienen zur Beschreibung des Experiments. Der Block 0 enthält Text mit Angaben über die Anzahl der Meßblöcke, Kanalzahl, Frequenz, Datum und Paßwort. Die Blöcke 1 und 2 enthalten Meßdaten, die zur überprüfung der Meßeinrichtung aufgenommen wurden.
- Falls ITB1 = 0 ist, wird der Inhalt von Block 0 gelistet, eine Konvertierung von Daten findet nicht statt.
- SEDAP entnimmt die Information über Frequenz, Datum und Zeit aus dem Labelblock.

NOPR

```
**************
*            *
*    NOPR    *
*            *
**************
```

unterdrückt den Seitenvorschub und die Kommentarauflistung aller nachfolgenden Befehle.

Befehlsform:

```
NOPR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Beispiel:

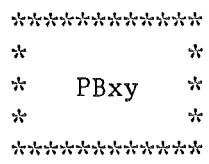
```
NOPR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Es wird nun bei jedem nachfolgenden Befehl der Seitenvorschub sowie die übliche Protokollierung unterdrückt.

Bemerkung:

Dieser Befehl wirkt bis zu dem nächsten PRNT-Befehl.
Durch NOPR wird nicht die Liste der Befehle PBHE, PBHF, PBVE, PBVF, BILD, ZUST und STOP unterdrückt. Das gleiche gilt für Fehlernachrichten. Die Standardangabe in SEDAP ist immer PRNT.

PBXY



druckt die Blöcke einer Datei auf dem Standard-Ausgabebefile.
PBxy ist ein Pseudonamen, denn die Buchstabenkombination xy kann durch folgende Angaben ersetzt werden:

x = H für eine horizontale Ausgabeliste
V für eine vertikale Ausgabeliste
y = E für das FORTRAN E-Format (E15.6)
F für das FORTRAN F-Format (F15.6)

Es sind also vier Befehle möglich: PBHE, PBHF, PBVE, PBVF

Befehlsform:

```
PBxy      DATA              IB1  IB2
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei, welche gedruckt weden soll
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes

Beispiel:

```
PBVF      TEMP                1    5
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die fünf ersten Blöcke der Datei TEMP sollen ausgedruckt werden. Das Ausgabeformat ist das FORTRAN F-Format und die Werteanordnung vertikal.

Bemerkung:

Die Angabe eines Raffungsfaktors ist nicht möglich.
Pro Seite wird ein Block (512) Werte ausgedruckt.

PLOT

```
**************
*            *
*    PLOT    *
*            *
**************
```

ermöglicht dem SEDAP-Benutzer die Plotausgabe

Befehlsform:

```
PLOT      DATA MODI        IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA ist der Name der Datei, welche geplottet werden soll

MODI ist ein Modifier und kann drei Bedeutungen haben: TEXT, GRID oder ALT*.
Wurde der Plotbefehl mit dem Modifier TEXT oder GRID angegeben, so wird eine neue Zeichnung eröffnet. Die Größe der Zeichnung kann durch unmittelbare Vorausdefinition (vor PLOT Befehl) mit Hilfe von DEFX und/oder DEFY vom Benutzer bestimmt werden. Fehlt die Vorausdefinition, wird die Zeichnung in der Standardgröße 20.0 * 15.0 cm ausgeführt. Die Angabe des TEXT Modifiers bewirkt das Lesen der nächsten SEDAP-Eingabekarte und verwendet davon die ersten 60 Zeichen für die Beschriftung des Plots. Es muß also in diesem Fall nach der PLOT- Befehlskarte eine Kommentarkarte folgen, sonst wird der nächste Befehl als Kommentar interpretiert. Diese Karte darf nicht mit dem Zeichen > in Spalte 1 beginnen.
Der Modifier GRID wird wie TEXT behandelt, es wird jedoch zusätzlich ein Gitter über die Zeichnung gezeichnet. Die Gitterabstände entsprechen denen der Skalenteilung, also alle 2.5 cm in beiden Richtungen. Fehlt die Angabe eines Modifiers, wird TEXT angenommen. Bei der Modifier Angabe ALT* wird keine neue Zeichnung eröffnet. Es sind maximal 30 Kurven in einem Plot möglich. Zur optischen Trennung der Kurven wird jede Kurve mit einem anderen Symbol gekennzeichnet und die Anzahl der Symbole steigt von PLOT-Befehl zu PLOT-Befehl. Ist die Ausdehnung irgendeiner Kurve größer als die des ersten PLOT (diese legt das Minimum und das Maximum fest), so wird diese Kurve abgeschnitten.

Fortsetzung siehe nächste Seite

PLOT - FORTSETZUNG

```
**********
*        *
*  PLOT  *
*        *
**********
```

IB2 ist der erste Block des verlangten Dateisegmentes
IB1 ist der letzte Block des verlangten Dateisegmentes
IR ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen.

Beispiel:

```
PLOT      TEMP              1    8
____      ____ ____ ____ *  |*   |*   |*    .   |*    .   |*    . |
```

Es werden die ersten 8 Blöcke der Datei TEMP geplottet. Da keine weitere Angabe getroffen wurde, wird eine neue Zeichnung eröffnet und keine Kommentarkarte für die Beschriftung eingelesen. Die Achse bekommt dann eine Standardbeschriftung.

Bemerkung:

Dieser Befehl kann maximal 20 Blöcke verarbeiten (1 Block = 512 Werte).

POTI

```
**************
*            *
*    POTI    *
*            *
**************
```

potenziert die Werte einer Datei mit einem angegebenen Exponent

Befehlsform:

```
POTI       DATA       DATB    IB1  IB2    IR     EXP
____       ____ ____  ____  *   |*   |*   |*     .    |*     .    |*     . |
```

DATA	ist der Name der ersten Datei, welche potenziert werden soll
DATB	ist der Name der Ergebnisdatei nach der Potenzierung
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen
EXP	ist der Exponent mit dem die Werte der Datei DATA potenziert werden sollen.

Beispiel:

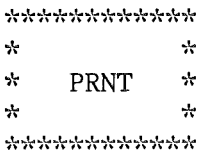
```
POTI       NORM       EXNO     1    9          2.5
____       ____ ____  ____  *   |*   |*   |*     .    |*     .    |*     .  |
```

Die Werte der Datei NORM, welche 9 Blöcke umfaßt, sollen mit dem Exponent 2.5 potenziert werden.

Bemerkung

Durchführung der Potenzierung: Ist der Exponent nicht ganzzahlig (z.B. 2.35) so wird von FORTRAN intern ein Verfahren zur Potenzierung angewandt, das keine negative Mantisse zuläßt. Es wird in einem solchen Fall eine FORTRAN Fehlermeldung ausgedruckt und dann bei mehrmaligem Auftritt die Ausführung abgebrochen. Ist der Exponent dagegen ganzzahlig (z.B. 3.0) so kann die Mantisse negativ sein. Benutzen Sie zum Quadrieren den Befehl MULT DATA DATA, das geht schneller.

PRNT



hebt den Befehl NOPR auf

Befehlsform:

PRNT
____ ____ ____ ____ * |* |* |* . |* . |* . |

Beispiel:

PRNT
____ ____ ____ ____ * |* |* |* . |* . |* . |

hebt den vorangegangenen NOPR-Befehl auf.

Bemerkung

Ist dem PRNT kein NOPR-Befehl vorangegangen, so hat der PRNT-Befehl keine Wirkung.

RENA

```
**************
*            *
*    RENA    *
*            *
**************
```

verändert den Namen einer im Lager gespeicherten Datei

Befehlsform:

```
RENA        ALTD        NEUD
____        ____ ____   ____ *   |*   |*   |*    .    |*    .    |*    . |
```

ALTD ist der Name der alten Datei dessen Name geändert werden soll
NEUD ist der Name der Datei nach der Änderung

Beispiel:

```
RENA        CH34        TEMP
____        ____ ____   ____ *   |*   |*   |*    .    |*    .    |*    . |
```

Die Datei CH34 wird in TEMP umbenannt und kann nun mit dem Namen TEMP aufgerufen werden

Bemerkung

Die Verwendung des Befehles RENA ist besonders nach der Sortierung von Dateien zu empfehlen.

SEDA

```
**********
*        *
*  SEDA  *
*        *
**********
```

bewirkt die Initialisierung von SEDAP

Befehlsform:

```
SEDA      DUMP MODI      ISIZE     IFILE
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DUMP bedeutet: Sollte während des Jobablaufes ein Fehler in der SEDAP-Eingabe auftreten und aus diesem Grund eine Weiterverarbeitung nicht mehr möglich sein, so wird, falls IFILE angegeben wurde, der aktuelle Lagerinhalt gerettet. Diese Werte können dann in einem späteren Step mit dem HOLE-Befehl wieder zugänglich gemacht werden.

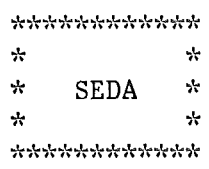
MODI ist ein Modifier und kann 3 Bedeutungen haben:
ALT*, NEU* oder TEMP.
ALT* bedeutet, SEDAP soll ein schon vorhandenes Lager verwenden. Es muß deswegen keine Initialisierung durchgeführt werden.
NEU* bedeutet, SEDAP soll ein neues Lager anlegen, initialisieren und nach Beendigung des Jobs das Lager nicht zerstören.
TEMP bedeutet, SEDAP soll ein Lager nur während der Jobausführung anlegen und danach wieder zerstören.
Fehlt der Modifier, so wird TEMP angenommen. In der JCL bewirkt diese Angabe folgendes: Im Falle ALT* muß der Dataset vorhanden sein. Ein vorangegangener SEDAP Lauf muß ihn außerdem initialisiert haben. Bei NEU* wird ein neu allokiertes Dataset initialisiert. Bei TEMP oder keiner Angabe wird vom System ein temporärer File angelegt und dieser bei Jobbeendigung gelöscht.

ISIZE ist die Lagergröße in Blöcken (a 512 Werten).

IFILE ist die File-Nummer des Dumpfiles.

Fortsetzung siehe nächste Seite

SEDA - FORTSETZUNG



Beispiel:

```
SEDA      DUMP            1000        33
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

SEDAP wird initialisiert und Platz für 1000 Blöcke bereitgestellt. Tritt die DUMP-Situation ein, muß durch eine DD-Karte mit dem Namen G.FT33.... Platz zum Retten der Dateien zur Verfügung gestellt sein (siehe Kapitel IBM-Job-Control-Karten, Beispiel DUMP).

Fehlt diese Angabe, so wird eine Lagergröße von 500 Blöcken angenommen. Eine größere Blockzahl muß außerdem bei dem Block-Parameter in der SEDAP-Prozedur berücksichtigt werden.
Z.B.

```
//   EXEC SEDAP,BLOCK=9000
```

Diese Angabe wirkt auf die Space-Allokierung in der //FT40F001 DD-Karte. (Siehe Prozedurliste im Kapitel IBM-JOB-CONTROL-KARTEN.)

SONN

```
**************
*            *
*    SOnn    *
*            *
**************
```

sortiert SEDAP-Dateien, das heißt es zerlegt sie in mehrere kleinere.
SOnn ist ein Pseudoname für folgende Sortierbefehle:

SO02, SO04, SO08, SO16, SO32 und SO64

Befehlsform:

```
SOnn      DATA      DA$$   IB1  IB2  LIM
____      ____ ____ ____ *  |*   |*   |*    .   |*    .   |*    . |
SO02      DATA      DAFT   IB1  IB2  LIM
____      ____ ____ ____ *  |*   |*   |*    .   |*    .   |*    . |
```

DATA ist der Name der Datei, welche sortiert werden soll

DA$$, DAFT
stehen für die Namen der Ergebnisdateien. Sie werden durch die ersten zwei angegebenen Buchstaben gebildet plus den Zahlen von 1 - nn oder LIM, z.B. für nn=16 von DA01 DA02 ... DA16. Im ersten Beispiel ist die variable Seite des Namens durch die beiden $ Zeichen erkenntlich gemacht. Dies ist normalerweise nicht notwendig, sollte aber den Benutzer daran erinnern, daß er in diesem Fall keinen Einfluß auf die beiden letzten Stellen des Namens hat. Bei verschiedenen Befehlen werden Dateien erzeugt mit Real- und Imaginärteil. Durch den Befehl SO02 kann man diese Dateien trennen. Man muß aber beachten, daß in diesem Fall statt der beiden $-Zeichen die Buchstabenkombination FT anzugeben ist. Dies bewirkt eine richtige Frequenzbehandlung der komplexen Dateien (die Frequenz wird in diesem Fall nicht durch 2 dividiert).

IB1 ist der erste Block des verlangten Dateisegmentes

IB2 ist der letzte Block des verlangten Dateisegmentes

LIM ist eine zusätzliche Angabe und bewirkt den vorzeitigen Abbruch der Sortierung bei dem Wert LIM. Aus diesem Grund sollte LIM zwischen 1 und nn liegen.

Fortsetzung siehe nächste Seite

SONN - FORTSETZUNG

```
**************
*            *
*    SOnn    *
*            *
**************
```

Beispiel:

```
SO16        DATA        CH$$       1   32   11
____        ____ ____ ____ *    |*   |*   |*     .    |*     .    |*     . |
```

Sortiere die Datei DATA mit dem Sortierfaktor 16. Speichere aber nicht alle 16 Kanäle, sondern nur 11 Kanäle. In das Lager werden nun die Dateien DA01,DA02 ... DA11 gespeichert. Jede Datei ist 2 Blöcke groß

Bemerkung:

Wurde LIM angegeben, so verursacht dies, daß nur LIM-Dateien nach der Sortierung im Lager gespeichert sind. Von dieser Möglichkeit sollte man unbedingt Gebrauch machen, wenn man Rohdaten, die durch ein Multiplexverfahren zusammengepackt wurden, trennen will. So werden z.B. beim SO64 Befehl in den wenigsten Fällen alle erzeugten Dateien von Wichtigkeit sein, außerdem spart man Rechenzeit und Platz im Lager. Mit LIM = 40 werden z.B. nur die Kanäle 1 bis 40 aus 64 vorhandenen Kanälen sortiert.

SUBT

```
**********
*        *
*  SUBT  *
*        *
**********
```

subtrahiert zwei Dateien voneinander und speichert die Ergebnisdatei in das Lager.

Befehlsform:

```
SUBT      DATA DATB DATC    IB1  IB2   IR
____      ____ ____ ____ *  |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Datei von welcher subtrahiert wird
DATB	ist der Name der Datei, welche subtrahiert wird
DATC	ist der Name der Ergebnisdatei nach der Subtraktion (C=A-B)
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

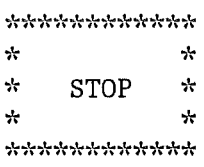
```
SUBT      CH10 CH20 DIFF     2    4    2
____      ____ ____ ____ *  |*   |*   |*    .   |*    .   |*    . |
```

Von den Werten der Datei CH10 werden die Werte der Datei CH20 abgezogen. Das Ergebnis von einem Block (IR=2), wird unter dem Namen DIFF im Lager gespeichert.

Bemerkung:

Siehe Sonderbemerkungen für die Verarbeitung von synchronen Dateien im SEDAP KfK-Bericht.
SUBT kann auch zum Subtrahieren komplexer Dateien verwendet werden.

STOP



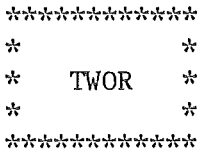
Befehlsform:

STOP
____ ____ ____ ____ * |* |* |* . |* . |* . |

Bemerkung:

Der STOP-Befehl sollte der letzte Befehl der SEDAP-Eingabe sein. Fehlt er, so wird das Programm abnormal beendet.
Karten, die nach der STOP-Karte folgen, werden mit der Eingabe aufgelistet, aber nicht verarbeitet.

TWOR



verursacht die Umwandlung von Thermospannungswerten in Grad Celsius. Die Umwandlung erfolgt nach der Spannungscharakteristik und Toleranz für Wolfram Renium Thermoelemente.

Befehlsform:

```
TWOR      DATA      DATC   IB1  IB2   IR
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA	ist der Name der Eingabedatei
DATC	ist der Name der Ergebnisdatei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen

Beispiel:

```
TWOR      CH15      TE15     1    3    3
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Die Werte der Datei CH15 werden von Millivolt in Grad Celsius umgewandelt, und die ein Block große Ergebnisdatei in das Lager mit dem Namen TE15 gespeichert.

Bemerkung:

Die Werte der Inputdatei sind in Millivolt anzugeben. Sie müssen positiv sein.

WERT

```
**************
*            *
*    WERT    *
*            *
**************
```

bietet die Möglichkeit Teile aus einer Datei herauszukopieren und diese weiterzuverarbeiten. Zwei Eingabeversionen sind möglich:

```
WERT      DATA      DATB   IP1  IP2
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |

                                                                  oder
WERT      DATA ZEIT DATB                      T1        T2     TFLOAT
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA ist der Name der Eingabedatei
DATB ist der Name der Ergebnisdatei
IP1 ist der erste Punkt innerhalb der Datei, ab welchem verarbeitet werden soll
IP2 ist der letzte Punkt der Datei, bis zu welchem verarbeitet werden soll
ZEIT bedeutet, daß die Grenzen nicht durch Punkte, sondern durch Zeiten angegeben werden.
T1 ist der erste Zeitpunkt (SEC) ab welchem verarbeitet werden soll
T2 ist der letzte Zeitpunkt bis zu welchem verarbeitet werden soll
TFLOAT ist ein Faktor, mit dem T1 und T2 in der Ergebnisdatei multipliziert werden, so daß sich ein Zeitbereich T1*TFLOAT bis T2*TFLOAT ergibt.
Standard-Wert ist 1

Beispiel:

```
WERT      TEMP      T200     1  200
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Bilde eine neue Datei mit dem Namen T200. Ihr Inhalt besteht aus den ersten 200 Werten der Datei TEMP

Fortsetzung siehe nächste Seite

WERT - FORTSETZUNG

```
                                                    **************
                                                    *            *
                                                    *    WERT    *
                                                    *            *
                                                    **************
```

```
WERT      TEMP ZEIT TCUT                    15.0      25.0        0.001
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    .   |
```

Bilde eine neue Datei mit dem Namen TCUT. Der Inhalt besteht aus den Werten der Datei TEMP, die zwischen 15.0 und 25.0 ms liegen (ms wegen TFLOAT = 0.001).

Bemerkung:

Die Angaben IP1 und IP2 müssen folgenden Bedingungen unterliegen:

IP1 < IP2 < = 99999 (I5 FORTRAN Format)

IP1 > 0

Ist TFLOAT = 0. oder wurde keine Angabe gemacht, so wird TFLOAT = 1. gesetzt. Die Benutzung des Befehls WERT setzt eine genaue Kenntnis des Lagerinhaltes in bezug auf Frequenz und Punktzahl voraus. Diese Werte können durch den Befehl BILD eingesehen werden.
Die Angabe eines Raffungsfaktors ist nicht möglich.

XTSD

```
**************
*            *
*    XTSD    *
*            *
**************
```

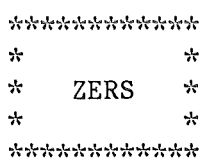
Dieser Befehl ermöglicht dem SEDAP-Benutzer auf zusätzliche Programme zuzugreifen, die er selbst geschrieben hat und deren Ausführung zu veranlassen. Da die Mannigfaltigkeit dieser 'EXTEND' Programme nicht bekannt ist, kann auch hier über den Aufbau dieser Eingabekarte nichts geschrieben werden.
Es sei jedoch bemerkt, daß sich der Benutzer dieses Befehles an die formatgebundene Erklärung der SEDAP-Befehle halten muß (siehe Tabelle im Kapitel: allgemeine Erklärung der Befehlsstruktur).
Das Schreiben eines 'EXTEND' Programmes setzt eine genaue Kenntnis des Programmes SEDAP voraus. Dieses wird im Rahmen dieses Handbuches nicht beschrieben. Es sei aber erwähnt, daß in dem SEDAP-KFK-Bericht diese Möglichkeit erklärt wird.
Ein oft auftretender Standardfall wird im Kapitel SEDAP EXTEND BEISPIEL dargestellt.

Befehlsform:

```
XTSD      DATA      DATB   IB1  IB2   IR   REAL1     REAL2     REAL3
____      ____ ____ ____ *   |*   |*   |*    .    |*    .    |*   .    |
```

DATA	ist der Name der Eingabedatei
DATB	ist der Name der Ergebnisdatei
IB1	ist der erste Block des verlangten Dateisegmentes
IB2	ist der letzte Block des verlangten Dateisegmentes
IR	ist der Raffungsfaktor, mit welchem die Inputwerte gerafft werden sollen
REAL1	Real Wert
REAL2	Real Wert
REAL3	Real Wert

ZERS



zerstört d.h. löscht eine oder alle gespeicherten Dateien im Lager.
Zwei Befehlsvariationen sind möglich:

```
ZERS      DATA
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |

ZERS           ALLE
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

DATA ist der Name der Datei, welche gelöscht werden soll
ALLE ist ein Modifier und verursacht das Löschen aller Dateien

ZUST

```
**************
*            *
*    ZUST    *
*            *
**************
```

listet alle gültigen Kommandos des z.Z. implementierten Systems auf und gibt Auskunft über die gültige Version des SEDAP Handbuches

Befehlsform:

ZUST

```
____      ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    . |
```

Bemerkung:
Nach der Auflistung aller z.Z. in SEDAP implementierten Befehle erscheint folgender Wortlaut:

Gültige Programmbeschreibung vom 1.06.1984

Es sind somit alle älteren Programmbeschreibungen ungültig. Es wird deshalb geraten, wenn ein Benutzer SEDAP nicht regelmäsig benutzt, nach gewissen Zeitabschnitten den Befehl ZUST aufzurufen, um damit festzustellen, ob er überhaupt noch eine gültige Programmbeschreibung hat.

FEHLERNACHRICHTEN

IERR = 1 Band Lesefehler. Die Information auf dem Magnetband ist zerstört.
Magnetband ist daraufhin zu prüfen.

IERR = 2 End of file auf Magnetband. Dies kann nur auftreten, wenn eine Blockzahl oder eine Datei gesucht wird, die nicht auf dem angegebenen File des Magnetbandes zu finden ist.

IERR = 3 Direct access Lesefehler. Die in SEDAP einprogrammierte direkte Zugriffsmethode hat beim Lesen eines Blockes eine Fehlermeldung gebracht.

IERR = 4 Die angegebene Datenmenge kann nicht aus dem Lager geholt werden, da die Datei aus einer Blockzahl besteht, die kleiner ist als die angegebene.

IERR = 5 Der angegebene Anfangswert ist größer als der Endwert.

IERR = 6 Mit diesem Befehl können nur bestimmte Mengen von Punkten verarbeitet werden.

IERR = 7 Die zu erzeugende Datei ist schon im Lager. Es ist daher unmöglich, eine zweite Datei mit gleichem Namen einzubringen.

IERR = 8 Die verlangte Datei ist nicht im Lager.

IERR = 9 Dieser Fehler tritt z.B. auf, wenn die durch den SEDA-Befehl begrenzte Lagergröße überschritten wurde.

IERR =10 SEDAP ist so konzipiert, daß die Kataloge zur Verwaltung der Dateien max 512 Worte beinhalten können. Aus diesem Grund können nur max 512 Dateinamen eingetragen werden. Wenn dies nicht ausreicht, kann der Benutzer mit Hilfe des ZERS-Befehls Platz im Lager und damit auch im Katalog schaffen.

IERR =11 Zur Initialisierung (erste Karte) und zur Dimensionierung vierte Karte) wurde ein ungültiger Befehl gegeben. An diesen Stellen kann nur der Befehl SEDA oder das Wort SEDAP stehen.

IERR =12 Der angegebene Befehl ist kein gültiger Befehl. Die Liste der gültigen Befehle ist in diesem Handbuch enthalten oder kann durch den Befehl ZUST ausgedruckt werden.

IERR =13 Der angegebene Modifier ist kein gültiger Modifier. Die Liste der gültigen Modifier kann in diesem Handbuch eingesehen werden.

IERR =14 Zur Ausführung dieses Befehls muß in der Datei eine Mindestpunktezahl > = 1 vorhanden sein. Die Begrenzungen sind bei folgenden Befehlen gegeben.

FIL1 > = 3	FIL2 > = 5	FIL3 > = 5	FIL4 > = 3
HAFU > = 3	DIFF > = 5	INSI > = 5	INTR > = 3

IERR =15 Die Frequenz ist < 0.0. Durch die Angabe keiner Frequenz in den Eingabebefehlen wird dem Frequenzwert = 0.0 zugewiesen.

IERR =16 Der angegebene Anfangswert ist entweder < 0 oder = 0. Der zweite Fall kann eintreten, wenn keine Angabe auf der Befehlskarte gemacht wurde.

IERR =17 Im Lager befindet sich noch keine erstellte Datei.

IERR =18 Der angegebene Raffungsfaktor liegt außerhalb des zugelassenen Bereichs von 1 bis 100.

IERR =19 Sonderfehler mit Code Nr.:

102 Der angegebene Überlappungswert der Dateisegmente im LEDI-Befehl ist nicht sinnvoll.

IERR =20 Der Labelblock eines Sedap kompatiblen Bandes ist falsch kodiert.

IERR =23 Zu viele Kurven in einem Plott. Es dürfen maximal 30 Kurven in einem Plott übereinander gezeichnet werden.

PROGRAMMBEISPIEL

Aus einem Versuch mit 16 angeschlossenen Temperaturmeßstellen, welche einen dynamischen Temperaturverlauf haben, soll die Meßstelle T16 nach folgenden Spezifikationen ausgetestet werden:

1. Es sind die auf dem Magnetband aufgenommenen Datenblöcke 3 - 130 zu konvertieren.
   (Karte 5)
2. Die Meßdaten den 16 Kanälen entsprechend zu sortieren.
   (Karte 6)
3. Eine Temperaturbezugspunktkorrektur durchzuführen.
   (Karte 7)
4. Die Meßwerte von MV in Grad C umzuwandeln.
   (Karte 8)
5. Die gemessene Kurve zu differenzieren.
   (Karte 9)
6. Die differenzierte Kurve mit der Zeitkonstante zu multiplizieren.
   (Karte 10)
7. Zur Ermittlung der korrigierten Kurve, gemäß der Funktion:
   TAU * X (T) + (X) T = Y (T)
   die entsprechenden Records zu addieren.
   (Karte 11)
8. Eine Y-Achse mit Y-MIN, Y-MAX und einer Längenangabe festzulegen (in cm).
   (Karte 12)
9. Eine X-Achse mit X-MIN, Y-MAX und einer Längenangabe festzulegen.
   (Karte 13)
10. Die Temperaturkurve mit Text zu zeichnen.
    (Karte 14+15)
11. Über der Temperaturkurve die korrigierte Kurve aufzutragen.
    (Karte 16)

Die Umsetzung dieser Auswerteangaben ergibt in der SEDAP-Sprache folgende Befehlsliste:

```
SEDAP    NSK
Auswertung eines NSK-Versuches.
Korrektur der gemessenen Temperatur.
SEDA                    5000
NGET                DATE    3  130   21
SO16     DATE       DA$$    1  256    6
AX+B     DA06       AX06    1   16          .005       .0
TWOR     AX06       CX06    1   16
DIFF     CX06       EX06    1   16
AX+B     EX06       FX06    1   16          0.0142
ADDI     FX06 CX06  IX06    1   16
DEFY                                       650.000   800.000     20.
DEFX                                         5.80      6.57      25.
PLOT     CX06 TEXT          1   16           5.80      6.57      25.
 NSK-Versuch Nr. 1/ vom 29.4.71/Mess.-St. T16 mit Temp.-Korrektur
PLOT     IX06 ALT*          1   16
STOP
```

Im folgenden sind zwei Programme gelistet. Das erste ist ein FORTRAN Programm zur Erstellung von Datensätzen, die SEDAP lesen kann. Das zweite Programm ist ein SEDAP-Eingabebeispiel zum Verarbeiten der im ersten Programm erzeugten Datei.

```
C
      PROGRAM SEDFOR
C
C
C     DIESES PROGRAMM ERZEUGT EINEN SEQUENTIELLEN FILE, DER ANSCHLIES-
C     SEND MIT DEM SEDAP HOLE - BEFEHL WIEDER GELESEN WERDEN KANN.
C     AUSSERHALB DER KFK IST DIES DIE EINZIGE MOEGLICHKEIT ZUR DATEN-
C     EINGABE FUER DAS SEDAP PROGRAMM.
C
C     DER FILE IST FOLGENDERMASSEN AUFGEBAUT :
C
C       |-------------------------------------------------------|
C       |    LABEL BLOCK MIT 128 WORTEN A 4 BYTE LAENGE         |
C       |-------------------------------------------------------|
C       |    DATEN BLOCK NR 1 MIT 512 WORTEN A 4 BYTE LAENGE    |
C       |-------------------------------------------------------|
C
C       |-------------------------------------------------------|
C       |    DATEN BLOCK NR N MIT 512 WORTEN A 4 BYTE LAENGE    |
C       |-------------------------------------------------------|
C
C     DIESE DATENSTRUKTUR (LABEL UND DATENBLOECKE) KANN MEHRMALS
C     WIEDERHOLT WERDEN.
C
C     DER LABEL BLOCK IST FOLGENDERMASSEN AUFGEBAUT :
C
C       |-------------------------------------------------------------|
C       | INTEGER*4     | LAENGE DES LABELS                           |
C       | INTEGER*4     | DATEI ZAEHLER                               |
C       | INTEGER*4     | ANZAHL DER ALLER WERTE IN DEN DATEN BLOECKEN |
C       | INTEGER*4     | ANZAHL DER WERTE IM LETZTEN DATEN BLOCK     |
C       | CHARACTER*80  | TEXT STRING FUER INFORMATIONEN              |
C       | CHARACTER*4   | NAME DER DATEI (FUER SEDAP VERARBEITUNG)    |
C       | REAL*4        | FREQUENZ DER DATEI                          |
C       | REAL*4        | DATUM DER DATEI         11.11.1983 = 1111.83 |
C       | REAL*4        | ZEIT DER DATEI  9 UHR 45 MIN 3 SEC = 9445.3 |
C       | REAL*4        | UNBENUTZT                                   |
C       | REAL*4        | UNBENUTZT                                   |
C       | REAL*4        | UNBENUTZT                                   |
C       | REAL*4        | UNBENUTZT                                   |
C  |-------------------------------------------------------------|
C
C
C    IM FOLGENDEN BEISPIEL WERDEN ZWEI DATEIEN AUF EINEN SEQUENTIELLEN
C    FILE GESCHRIEBEN. DER ERSTE FILE BESTEHT AUS VIER DATENBLOECKEN,
```

```
C   DER ZWEITE AUS DREI. BEI BEIDEN WIRD DER LABEL BLOCK VORANGESCHRIE-
C   BEN.
C
C
      INTEGER      ISATZ(4),IWRT,IBLK
      REAL         FSATZ(7),V(512)
      CHARACTER    DNAM*4,TSATZ*80,LSATZ(20)*4
      EQUIVALENCE  (TSATZ,LSATZ(1))
C
      KFILE = 2
      IWRT  = 512
C
C - ERZEUGE LABELBLOCK FUER DIE ERSTE DATEI
C
      IBLK     = 4
      ISATZ(1) = 128
      ISATZ(2) = 1
      ISATZ(3) = IWRT * IBLK
      ISATZ(4) = IWRT
      TSATZ    = 'BESCHREIBUNGSTEXT FUER ERSTE DATEI
      DNAM     = 'DATA'
      FSATZ(1) = 100.
      FSATZ(2) = 1410.83
      FSATZ(3) = 1053.36
      WRITE (KFILE) (ISATZ(I),I=1,4),(LSATZ(I),I=1,20),DNAM,
     &              (FSATZ(I),I=1,7)
C
C - ERZEUGE DATENBLOECKE FUER DIE ERSTE DATEI
C
      DO 20 J=1,IBLK
         DO 10 I=1,IWRT
            V(I) = J
   10    CONTINUE
         WRITE (KFILE) V
   20 CONTINUE
C
C - ERZEUGE LABELBLOCK FUER DIE ZWEITE DATEI
C
      IBLK     = 3
      ISATZ(1) = 128
      ISATZ(2) = 2
      ISATZ(3) = IWRT * IBLK
      ISATZ(4) = IWRT
      TSATZ    = 'BESCHREIBUNGSTEXT FUER ZWEITE DATEI
      DNAM     = 'DATB'
      FSATZ(1) = 100.
      FSATZ(2) = 1410.83
      FSATZ(3) = 1053.36
      WRITE (KFILE) (ISATZ(I),I=1,4),(LSATZ(I),I=1,20),DNAM,
     &              (FSATZ(I),I=1,7)
C
C - ERZEUGE DATENBLOECKE FUER DIE ZWEITE DATEI
C
      DO 40 J=1,IBLK
```

```
         DO 30 I=1,IWRT
            V(I) = J + 100.
   30    CONTINUE
         WRITE (KFILE) V
   40 CONTINUE
C
C
      STOP
      END
```

```
SEDAP     FORT  77
 DIESER TEST SOLL EIN SEQUENTIELLEN FILE EINLESEN, DER MIT EINEM
 EXTERNEN FORTRAN PROGRAMM ERZEUGT WURDE.
SEDA                        5000
HOLE                 DATA      1     4     2
HOLE                 DATB      1     3     2
BILD           ALLE
ZERS           ALLE
HOLE           ALLE                       2
BILD           ALLE
STOP
```

SEDAP EXTEND BEISPIEL

```
C
C   DIESES EXTSED PROGRAMM BEISPIEL SOLL DIE ANWENDUNG EINES SEDAP
C   XTSD BEFEHLES ZEIGEN. IM DATENARCHIV (LAGER) VON SEDAP BEFINDEN
C   SICH DATENBLOECKE AUS VORANGEGANGENEN RECHENSCHRITTEN. DIESE DATEN
C   WERDEN MIT SEDAP INTERNEN HILFSROUTINEN GEHOLT UND WIEDER IN DER
C   SEDAP INTERNEN FORM WEGGESCHRIEBEN.
C   UMGERECHNET WERDEN DIE WERTE GEMAESS DER GLEICHUNG :
C
C              Y(I) = X(I) / SQRT(REAL1-REAL2)
C
C   DAS ANWENDUNGSBEISPIEL STEHT HINTER DER SUBROUTINE.
C
C-----------------------------------------------------------------------
      SUBROUTINE EXTSED (NAM1,NAM2,NAM3,K1,K2,K3,K4,INT1,INT2,INT3,
     &                   REAL1,REAL2,REAL3)
C-----------------------------------------------------------------------
C
C  ERLAUETERUNG DER ARGUMENTE :
C
C        NAM1  = ERSTER DATEINAME AUF DER EINGABEKARTE (VON DATEI)
C                IM BEISPIEL : 'DATA'
C        NAM2  = ZWEITER DATEINAME AUF DER EINGABEKARTE (VON DATEI)
C                IM BEISPIEL NICHT VERWENDET
C        NAM3  = DRITTER DATEINAME AUF DER EINGABEKARTE (NACH DATEI)
C                IM BEISPIEL : 'DATB'
C        K1,K2 = WERDEN VOM HAUPTPROGRAMM GELIEFERT UND DIENEN ALS
C        K3,K4   ARGUMENTE FUER DIE DATENTRANFERROUTINEN. IN DER
C                SUBROUTINE EXTSED SOLLTEN DIESE WERTE ALSO NICHT
C                VERAENDERT WERDEN.
C        INT1  = ERSTER INTEGERWERT AUF DER EINGABEKARTE.
C                BEDEUTET ANFANGSBLOCKNUMMER DER ZU VERARBEITENDEN
C                DATEI. IM BEISPIEL : 1
C        INT2  = ZWEITER INTEGERWERT AUF DER EINGABEKARTE.
C                BEDEUTET ENDBLOCKNUMMER DER ZU VERARBEITENDEN
C                DATEI. IM BEISPIEL : 2
C        INT3  = DRITTER INTEGERWERT AUF DER EINGABEKARTE.
C                BEDEUTET RAFFUNGSFAKTOR DER ZU VERARBEITENDEN
C                DATEI. IM BEISPIEL NICHT ANGEGEBEN.
C        REAL1 = ERSTER REALWERT AUF DER EINGABEKARTE.
C                IM BEISPIEL : 1.97
C        REAL2 = ZWEITER REALWERT AUF DER EINGABEKARTE.
C                IM BEISPIEL : 0.35
C        REAL3 = DRITTER REALWERT AUF DER EINGABEKARTE.
C                IM BEISPIEL NICHT VERWENDET.
C
C
*INCLUDE SEDCOM
C
C
```

```
C
      INTEGER          IMESS,ISTAT,ISTAK,MAX,KXYZ,KSHIFT,KFUNC,
     &                 LOEF,KOF,KUF,KY,KSHI,
     &                 K1,K2,K3,K4,INT1,INT2,INT3
      REAL             REAL1,REAL2,REAL3
      CHARACTER*4      NAM1,NAM2,NAM3
C
C
      IMESS  = 1
      ISTAT  = 0
      ISTAK  = 0
      MAX    = MAXBLK
      KXYZ   = 2
      KSHIFT = 0
      KFUNC  = 1
      LOEF   = 512
      KOF    = 0
      KUF    = 0
      KY     = 3
      KSHI   = 0
C
C - UEBERPRUEFE OB REAL1 - REAL2 = 0.
C
      FAK = REAL1 - REAL2
      IF (FAK.EQ.0.) THEN
         WRITE (NP,1000)
         FAK = 1.
      ENDIF
C
C - HOLE INFORMATION UEBER DIE DATEI DIE VERARBEITET WERDEN SOLL
C
      CALL OPEIN (INT1,INT2,NAM1,K1,INT3,TIME,MAX,LKPT,FREQ,DATE)
      IF (IERR.GT.0) GO TO 99
C
C - UEBERPRUFE UND TRAGE INFORMATIONEN UEBER DIE DATEI DIE ERZEUGT
C   WERDEN SOLL IM DEPOT EIN.
C
      CALL OPAUS (LKPT,NAM3,K3,FREQ,DATE,TIME)
      IF (IERR.GT.0) GO TO 99
   10 CONTINUE
C
C - HOLE DIE WERTE DER DATEI DIE VERARBEITET WERDEN SOLL AUS DEM LAGER
C
      CALL ADDEIN (INT1,LKPT,LOEF,INT3,KSHIFT,IMESS,IM,KXYZ,ISTAT,IOF,
     &             IUF,KOF,KUF)
      IF (IERR.GT.0) GO TO 99
      KPT = IM
      DO 20 I=1,KPT
         Y(I) = X(I) / SQRT(FAK)
   20 CONTINUE
C
C - SCHREIBE VERAENDERTE DATEI WIEDER IN DAS LAGER
C
      CALL ADDAUS (KFUNC,ISTAK,GNAM,KPT,KY,KSHI,IMESS)
```

```
      IF (LKPT.GT.0) GO TO 10
   99 CONTINUE
      RETURN
C
 1000 FORMAT (1H ,' DIE DIFFERENZ DER ANGEGEBEN REAL WERTE = 0.'/,
     &             '  ES WURDE FAK = 1.0 ANGENOMMEN')
C
      END
//SEDAP       EXEC SEDAP
//L.SYSIN     DD   DSN=&&LKSET,UNIT=SYSSQ,DISP=(OLD,DELETE)
//G.SYSIN     DD   *
SEDAP      EXTEND
 DIESER TEST SOLL DEN SEDAP XTSD BEFEHL DEMONSTRIEREN

SEDA                        5000
> ERZEUGE KONSTANTE
DAGE            KONS DATA      1    2      500.                    2.
PBHF       DATA                1    2
XTSD       DATA       DATB     1    2        1.97       0.35
PBHF       DATB                1    2
STOP
//
```

GRAPHIK ZUR SEDAP-ORGANISATION

Die Beschreibung der folgenden Darstellung befindet sich im Kapitel SEDAP ORGANISATION